Predicting the Optimal Computing Platform for Climate-Driven Ecological Forecasting Models

Scott Sherwin Farley

Master's Thesis

Advisor: John W. Williams

Introduction

Human-induced global environmental change, including climate warming, anthropogenic land use, and the spread of invasive species, threatens to severely alter biodiversity patterns worldwide in the coming century [@Thuiller:2007cn; @Thuiller:2008enb; @Root:2005ha; @Lowe:2011fl]. Habitat degradation and fragmentation, expansion of invasive species, and a loss of climatically suitable areas are expected to result in large-scale biotic reorganizations, including the extinction of over one-quarter of all species [@Thomas:2010ht]. Climate exerts significant control over species distributions, particularly over those of vascular plants [@woodward1987climate; @Salisbury:1926fn], implying that anthropogenic perturbations to atmospheric CO2 concentrations can have significant impact on species ranges [@Lowe:2011fl]. Assuming uniformitarianism [@Williams:2007iwa], statistical methods that quantify species responses to climatic gradients can then be used to forecast future biotic assemblages under different warming scenarios [@Thuiller:2008enb; @Guisan:2013hqa; @Maguire:2015eva; @Clark:2014gp; @Guisan:2005gz; @Guisan:2000tc]. Despite the potential for widespread, irreversible changes to the Earth's biosphere [@Barnosky:2012js], a growing volume of ecological data puts contemporary global change researchers in a position to forecast and mitigate impending catastrophic changes. Environmental monitoring efforts, such as the Long Term Ecological Research program (LTER, [@Hobbie:2003ej]), National Ecological Observatory Network (NEON, [@Schimel:2009tx]), community curated databases, like the Neotoma Paleoecological Database (http://neotomadb.org) and the Paleobiology Database (PBDB, http://paleobiodb.org), and modern biodiversity occurrence databases, such as the Global Biodiversity Information Facility (GBIF, http://www.gbif.org), form a network of informatics infrastructure for supporting global environmental change research. New cyberinfrastructures for ecoinformatics organize, store, and serve a tremendous amount of information to researchers attempting understand and forecast responses to perturbations in the earth system [@Brewer:2012bk; @Michener:2012ho]. Paleoenvironmental proxy data, including fossil pollen, macrofossils, and freshwater and marine diatoms, can be an important supplement to modern biodiversity data and are an important part of

these informatics efforts. When forecasting to future climate spaces, paleoecolgical proxies can enhance understanding climatic changes by providing information about bioclimatic gradients not found on Earth today [@Veloz:2012jw]. Understanding responses to past climates can improve prediction of distributions under climates with no modern analog that are likely to emerge in the near future[@Williams:2007iwa]. However, though collections of modern and historical biodiversity data have the potential for forecasting studies, their volume, uncertainty, and heterogeneity can make their uptake challenging for ecologists [@Hampton:2013ko]. Once a dataset has been assembled, statistical methods can be used to mine it for new insight, extract parameters, and simulate future scenarios, though these techniques can be computationally demanding, particularly on larger datasets [@Huang:2013iy]. Many contemporary Big Data applications, such as the popular microblogging service Twitter (http://twitter.com), require distributed, parallel, streaming methods to identify key analytical trends in real time [e.g., @Bifet:2011wa]. With the recent growth in monitoring and occurrence data, some ecological analysts apply similar, data-driven machine-learning techniques to ecological forecasting problems and have seen significant increases in predictive skill [@Elith:2006vt]. Climate-driven ecological forecasting models, variously called ecological niche models [@Peterson:2003gl], predictive habitat distribution models [@Guisan:2000tc], and species distribution models (SDMs)[@Guisan:2005gza], have seen extensive application in ecology, global change biology, evolutionary biogeography [@Thuiller:2008enb; @Araujo:2005jy], reserve selection [@Guisan:2013hqa], and invasive species management [@Ficetola:2007bn]. SDMs characterize a species' response to biospatial environmental gradients [@Franklin:2010tn], and use these responses to forecast potential future distributions under climate change scenarios. Recently, studies with an large number of occurrence records, along with computationally intensive modeling approaches, including nonparametric data-driven techniques [@Elith:2006vt] and Bayesian methods that rely on repeated sampling of full joint probability distributions [@Dawson:2016wa; @Clark:2014gp], have become commonplace.

With over 1 billion occurrence records in Neotoma and GBIF, traditional statistical methods for analyzing and forecasting ecological processes often cannot be applied without compromising analysis scope. Most leading SDM methods, though popular in the literature and highly skillful, are not scalable to very large datasets because they are not designed to take advantage of parallel processing or distributed computing. With so much data, ecological modelers are likely to need to adopt Big Data techniques, including distributed computing and ensemble parallelism, common in fields like bioinformatics [@Schatz:2010js], geonomics [@Anonymous:aiu_1fsc], climate analytics [@Schnase:2014dn], and private industry [@Mosco:2014cu]. As the volume of ecological data increases and the need for high resolution, accurate projections of biotic distributions becomes more pressing, reducing project scope [e.g., @Bolker:2009csa] can no longer be considered a valid option.

Cloud computing may offer a technological solution to some of the problems posed by the increasing Bigness of ecological data [@Hampton:2013ko; @Mich-

ener:2012ho]. Cloud computing is a architectural design pattern that provides "ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort" [@Mell:2012jj; @Hassan:2011uh; @Anonymous:mc8EfgMa]. With the rapid commercialization of cloud computing and the widespread availability of public cloud providers like Amazon Web Services (AWS) and the Google Cloud Compute Engine (GCE), scientists have a seemingly unlimited supply of computing resources at their disposal. The cloud has been touted by many of the largest players in Silicon Valley, and is credited with Obama's 2012 presidential election win, Netflix's ability to provide streaming entertainment to millions of consumers, and Amazon's massive success in online retailing ([@Mosco:2014cu]). In 2010, the U.S. Federal Government embraced the efficiency, speed, and economy of the cloud, requiring all federal agencies to adopt a "Cloud-First" policy when considering new information technology (IT) developments [@Kundra:2010vb]. Accordingly, the National Aeronautics and Space Administration (NASA) and the National Science Foundation (NSF) have both officially endorsed cloud technology [@Mosco:2014cu]. Moreover, researchers in many disciplines have posited that the cloud as the key to solving future computing and modeling challenges [e.g., @Yang:2011bd; @Hsu:2013jz].

Although the cloud offers a promising technological solution to the computational demands of ecological forecasting, there are few guiding principles on when the benefits, in reduced computing time, outweigh the financial costs of a cloud-based solution. The cloud introduces a novel expense model for computing: by partitioning large physical resources into smaller virtual machines (VMs), cloud providers can charge consumers for the use of computing resources, rather than their purchase [@Hassan:2011uh]. Under this new operational expense model, consumers are no longer tied to a single hardware configuration, rather they can scale their virtual instances up or down depending on computational demand [@Armbrust:2009wla]. Despite being more complex to set up and maintain, Cloud-based solutions can offer significant advantages over traditional desktop computing. While the exact costs of migrating to a cloud environment are difficult to estimate [but see @KeweiSun:2013dp], the computational time gains achieved by running models on high performance virtual instances can be measured empirically and combined with cost estimates to provide guidance on when a cloud based solution would be economically rational.

In this thesis, I develop a framework for predicting the optimal computing solution for a set of SDM activities in order to better understand limits to the application of climate-driven forecasting models and improve large scale modeling of biodiversity shifts in response to climate change. Treating workflow characteristics as model parameters, I posit a theoretical model for a scenario-specific optimal computing solution that minimizes computational time and financial cost. I gather data on the runtimes of four different classes of species distribution models under different parameterizations and on different computing hardware. Using this empirical dataset, I fit a nonparameteric learning model to this data to assess the drivers of SDM runtime and predict the execution

time of future modeling scenarios. My findings suggest that if SDMs, and biodiversity more generally, are to benefit from cloud computing, future effort should be directed towards developing models that more explicitly take advantage of parallelism and distributed computing frameworks.

Research questions

My research questions aim to develop a theoretically sound and empirically grounded understanding of the drivers of SDM execution time, and put this understanding towards predicting the computing solution with the lowest cost to the researcher. In particular, my research questions are:

1. With what skill can the runtime of climate-based SDMs be predicted?
2. Can an optimal computing solution for a given modeling scenario be predicted with confidence that this prediction is better than a null model that suggests no change in execution time between SDM scenarios.
3. What are the drivers of SDM execution time, and how sensitive is the runtime to changes in these drivers?

Background and Previous Work

Big Data in Ecology

The vast expansion of data the sciences has necessitated the development of revolutionary measures for data management, analysis, and accessibility [@Schaeffer:2008kl]. Worldwide data volume doubled nine times between 2006 and 2011, and successive doubling has continued into this decade [@Chen:2014fc]. With the large influx of massive geonomic sequences, long term environmental monitoring projects, phylogenetic histories, and biodiversity occurrence data, robust, expressive and quantitative methods are essential to the future of the biological sciences [@Schaeffer:2008kl]. As datasets become larger, significant challenges are encountered, including inability to move datasets across networks, necessity of high performance and high throughput computing techniques, increased metadata requirements for storage and data discovery, and the need to respond to new uses for the data [@Schnase:2014dn].

Ecological occurrence data are records of presence, absence, or abundance or individuals of a species, clade or higher taxonomic grouping that are fundamental to biodiversity analyses, ecological hypothesis testing, and global change research. These data are increasingly being stored in large, dedicated, community-curated databases like Neotoma, GBIF, PBDB. Since the early 1990s, the internet and associated IT and an increased willingness to share primary data between scientists precipitated rapid influxes of digital occurrence records. While there are known problems with the quality and consistency of data records in large occurrence databases [@soberon2002issues;], they provide a low-friction way to consume

large amounts of data that would otherwise be prohibitively time consuming to derive from the literature or in the field [@Beck:2014ky; @Grimm:2013uu]. Entire new fields, namely 'Biodiversity Informatics' [@Soberon:2004jh], 'Ecoinformatics' [@Michener:2012ho], and 'Paleoecoinformatics' [@Brewer:2012bk] have been developed and delineated to address the growing challenges and opportunities presented by the management, exploration, analysis and interpretation of primary data regarding life, particularly at the species level, now centralized in biodiversity databases [@Soberon:2004jh].

The term Big Data is typically used to describe very large datasets, whose volume is often accompanied by lack of structure and a need for real-time analysis. Big Data, while posing significant management and analysis challenges, can provide new insights into difficult problems [@Chen:2014fc]. Though the precise definition of Big Data is loose, there are two prominent frameworks for discriminating Big Data from traditional data. One characterizes Big Data as "term used to describe data sets so large and complex that they become awkward to work with using standard statistical software" [@Snijders:2012ww]. This ambiguous delineation is echoed in the advertising and marketing literature that accompanies products, like cloud computing, that facilitate Big Data analysis. For example, Apache Hadoop, a popular distributed computing framework, has described Big Data as "datasets which could not be captured, managed, and processed by general computers within an acceptable scope" [@Chen:2014fc].

Under this framework, the Bigness of the data is specific to both the time of analysis and the entity attempting to analyze it. @Manyika:2015vk suggest that the volume of data required to be Big can change over time, and may grow with time or as technology advances. Furthermore, the criteria for what constitutes Big Data can vary between problem domains [@Chen:2014fc], the size of datasets common in a particularly industry, and the kinds of software tools that are commonly used in that industry [@Manyika:2015vk]. The Big Data label is most often applied to datasets between several terabytes and several petabytes (2^40 to 2^50 bytes). However, because of ecology's lack of experience with massive datasets and limited analysis software in the discipline, ecological occurrence data clearly falls under the banner of Big Data.

The recent development of complex relational databases that store spatiotemporal occurrence records and their metadata suggests that traditional methods of data handling were not sufficient for modern ecological analyses. While the datasets are not particularly large in storage volume, they are composed of millions of heterogenous records with complex linkages. Consider the complexity of the relationships between different data records, for example. Figure 1 shows the Neotoma relational table structure, and the complicated web of relationships between each entity. Further developments, like application programming interfaces and language specific bindings, supplement the tasks of accessing, filtering and working with the large occurrence datasets [@Goring:2015cr; @Anonymous:sFxqyl5u; @Anonymous:GSjiJRTH]. While occurrence data does not require the disk space of popular commercial applications like Twitter and

Youtube, it has recently demonstrated a need for new, custom built tools to store, analyze, and use large numbers of records.

A second important framework by which to assess Big Data is the 'Four V's Framework'. First introduced by IBM and used by large technological companies in the early 2000's to characterize their data, it is now a popular and flexible framework under which to describe Big Data. Under this framework, a dataset's Bigness is described by its Volume, Variety, Veracity, and Velocity. @Yang:2013gm describe this framework, suggesting that "volume refers to the size of the data; velocity indicates that big data are sensitive to time, variety means big data comprise various types of data with complicated relationships, and veracity indicates the trustworthiness of the data" [@Yang:2013gm p 276.].

Since the late 1990s, the scale of biodiversity information alone has become challenging to manage. Figures 2a and 2b track the growth in collections of Neotoma and GBIF through time. In 1990, only 2 of the records now stored in Neotoma were in digitized collections. Today, there are over 14,000 datasets containing [XXX] individual occurrence records, and associated spatial, temporal, and taxonomic metadata, corresponding to an average growth rate of 1.4 records [[[XXX occurrences]]] per day. Nearly all records in Neotoma are derived from sedimentary coring or macrofossil extraction efforts, data gathering techniques that require large expenditures of time and effort [@Davis:1963hk; @Glew:2002fv]. GBIF houses digital records of well over 600 million observations, recorded specimens (both fossil and living), and occurrences described in the scientific literature. Since its conception at the turn of the century, the facility's holdings have grown nearly 300%, from about 180 million records in 2001 to over 614 million records today. GBIF's reliance on literature and museum specimens allow its holdings to precede its origin in 2001, however, it is within the last 15 years that the data's volume has become clearly apparent.

The second characteristic of Big Data in the four V's framework is the Variety of the data, and its 'various types with complicated relationships' [@Yang:2013gm]. Biodiversity data is highly diverse with many very complicated relationships and interrelationships. As shown in Figure 3a, Neotoma's holdings feature 23 dataset categories, including X-ray fluorescence (XRF) and isotopic measurements, macro fossils of both vertebrates and plants, modern and fossil pollen records, and freshwater diatom and water chemistry series. Similarly, in GBIF, there are 9 distinct record types, including human observations, living and fossil specimens, literature review, and machine measurements (Figure 3b). Though the records coexist in large biodiversity database, they are distinctly different, derived using different protocols by different communities of researchers.

The data's spatial and temporal nature causes complex interrelationships between data entities. All of Neotoma's records and 87.6% of GBIF's records are georeferenced to specific places on the earth's surface. The spatial information in these databases is supplemented by other fields that describe the location of the observation, including depositional setting, lake area, and site altitude, to improve contextual interpretation of occurrence data. Managing data with

a spatial component is nearly always more challenging than managing data without it. Digital representations of spatial phenomena must grapple with data that is a discrete representation of a continuous physical feature, correlations between parameters, space and time, and processes differences at heterogenous spatiotemporal scales [@Yang:2011iy]. Furthermore, occurrence data represents the work of many dispersed individual researchers and research teams. The controlled vocabularies and organization of aggregating databases helps to efficiently assimilate large numbers of records, however, nearly every record was collected, analyzed, and published by a different scientist. While some scientists have contributed many datasets to occurrence databases, most have only contributed one or two. The median number of datasets contributed to Neotoma is only 2 and the third quantile value is just 7 datasets. Each researcher is apt to use different equipment, employ different lab procedures, and utilize different documentation practices, contributing to a highly variable dataset.

Biodiversity data also has high levels of uncertainty associated with it, comprising the third of the Four V's. Some of the sources of uncertainty in the data, like spatial or temporal positional uncertainty can be estimated [@Wing:2005wl] or modeled [@Blaauw:2010kg]. Other sources of uncertainty have yet to be quantified, for example inter-researcher identification differences, measurement errors, and data lost in the transition from field to lab to database. A recent paper by the Paleon working group used expert elicitation to quantify the differences between the dates assigned to European settlement horizon, a process they argue varies between sites, and depends on the "temporal density of pollen samples, time-averaging of sediments, the rapidity of forest clearance and landscape transformation, the pollen representation of dominant trees, which can dampen or amplify the ragweed signal, and expert knowledge of the region and the late-Holocene history of the site"[@Dawson:2016wa]. The findings of this exercise suggest that paleoenvironmental inference from proxy data is highly variable between researchers. Moreover, some information is undoubtedly lost in the process of going from a field site through a lab workflow to being aggregated in the dataset. Though some procedural information accompanies the data records in the database, not all process details can be incorporated into database metadata fields, and probably more importantly, contextual details essential to proper interpretation of the data often gets lost on aggregation.

Both Neotoma and GBIF show high levels of quantifiable uncertainty, and are likely to show high levels of unquantifiable uncertainty as well. Of a random sample of 10,000 records of the genus *Picea* from GBIF, over half did not report spatial coordinate uncertainty. Of the 4,519 records that did, the average uncertainty was 305 meters, and the maximum was 1,970 meters. Clearly, such high levels of uncertainty might be problematic for modeling efforts [@Beck:2014ky]. Neotoma records show a similar uncertainty in their temporal information. Neotoma records each have a minimum, maximum, and most likely age for each age control point (e.g., radiocarbon date). Out of a sample of 32,341 age controls in the database, only 5,722 reported any age uncertainty at all. The summary statistics for these age controls suggest that the median age control has a tem-

poral uncertainty of 260.0 years. The 25% percentile is an uncertainty of 137.5 years and the 75% 751.2 years, suggesting that dates are only identifiable down to ± 130 years of the actual date, on average. [NEOTOMA UNCERTAINTY THROUGH TIME]. Considering sediment mixing, laboratory precision, and other processes at work this is a relatively minor uncertainty, but it certainly contributes to occurrence data's lack of veracity.

The final piece of the Big Data framework is the dataset's velocity, which characterizes the dataset's sensitivity to time. High velocity data must be analyzed in real time as a stream to produce meaningful insights. Tweets, for example, are analyzed for trends as they are posted. User's are drawn to participation in up-to-the minute discussion, and significant effort has been put towards the development of sophisticated algorithms that can detect clusters and trends in real time [@Kogan:2014hh; @Bifet:2011wa]. The rate of increase in data volume in both Neotoma and GBIF is not fast enough to invalidate the results from previous analyses, suggesting that it's velocity is not high enough to warrant streaming techniques. Neotoma's growth rate of approximately 1.4 new datasets each day (1990-2016 average) and GBIF's daily growth rate of about 59,000 records (2000-2015 average) are small compared to the total number records in the database. Unlike in many private sector applications, there is little incentive to researchers to immediately analyze new biodiversity records, since all new findings will be reported in the academic paper cycle, typically several months to years. Moreover, automated analyses of distributional data have been warned against, due to the overall poor data quality [@soberon2002issues] and high levels of uncertainty.

While not time sensitive, ecological occurrence data requires advanced, sophisticated techniques to store and analyze, and demonstrates high volume, low veracity, and significant variety, and should therefore fall under the auspices of Big Data. Traditional statistical analysis techniques and storage methods for occurrence data may begin to suffer because they were not designed to handle Big Data. Both GBIF and Neotoma are experiencing sustained and increasing growth that has not diminished the early 1990s. To fully and accurately derive value from new data being added to distributional databases, novel and advanced techniques for modeling and analyzing this data are required.

Cloud Computing in the Sciences

In recent years, large technology companies have promoted cloud computing as a way of overcoming the computational challenges associated with Big Data. Like grid computing, the cloud model standardized requests for computer resource to leverage distributed networks of physical machines to create a computing utility. As @Foster:2008cl suggest,

> "Cloud Computing not only overlaps with Grid Computing, it is indeed evolved out of Grid Computing and relies on Grid Computing

> as its backbone and infrastructure support. The evolution has been a result of a shift in focus from an infrastructure that delivers storage and compute resources (such is the case in Grids) to one that is economy based aiming to deliver more abstract resources and services (such is the case in Clouds)."

Grid computing never achieved large-scale success in industry, though it did succeed in forming massive federated systems providing computing power and data to scientists that still exist today, such as the Earth System Grid [@Foster:2008cl]. Unlike the grid's collective and project based utility-style, the cloud provides a pay-as-you-go business model and large economies of scale [@Armbrust:2009wl; @Hassan:2011uh]. While some organizations and universities have developed 'private clouds', large collections of virtualized servers not made available to the general public, many researchers have recognized the potential for incorporating public clouds, utility computing provided as a service by a cloud provider, into their workflows. With this technology, scientists with little or no computational infrastructure can have access to scalable and cost-effective computational resources[@Hsu:2013jz].

Major scientific organizations in the United States, including the NSF and NASA, have made major pushes to promote cloud computing in their own operation. Spurred by the U.S. Office of Management and Budget's 2010 "25 Point Plan to Reform Federal Information Technology Management" [@Kundra:2010vb], federal agencies are now required to adopt a "Cloud First" policy when "contemplating IT purchases and evaluate secure, reliable, and cost-effective cloud computing alternatives when making new IT investments" [@Anonymous:sigZdwPI]. The federal plan also provisioned programs to help agencies adopt cloud technologies, reducing the effort needed to screen cloud providers for data security policies and enable rapid procurement of cloud services [@Kundra:2010vb]. NASA developed its own high performance open-source cloud stack, Nebula, before concluding that its own needs were better served by public cloud providers [@Anonymous:sigZdwPI], though private clouds are still a popular method for large academic organizations [@Huang:2013iy]. In 2013, the NSF announced a $20 million dollar solicitation for supporting "research infrastructure that enables the academic research community to develop and experiment with novel cloud architectures addressing emerging challenges, including real-time and high-confidence systems"[1]. The public cloud now provides several large open-access datasets for public consumption, including Landsat images, real-time NEXRAD radar, and the 1000 Geonomes project, and claims that many research institutions, including the NASA Jet Propulsion Laboratory, among others, use their products and services[2].

Cloud technology, both public and private, has been extensively lauded for its application to the traditional Big Data fields of bioinformatics [@Stein:2007jo;

[1] https://aws.amazon.com/government-education/research-and-technical-computing/. Accessed 18 September, 2016.

[2] http://www.nsf.gov/pubs/2013/nsf13602/nsf13602.htm

@Hsu:2013jz, @Issa:2013jp] and climate analytics[@Schnase:2015wp; @Schnase:2014dn; @Lu:2011ii]. Cloud based solutions for bioinformatics research relieve the large memory requirements often present in geonomics and drug-design data [@Hsu:2013jz]. Biomedical scientists have developed a variety of public-cloud based applications including low latency, streaming methods for data analysis [@Issa:2013jp] and biology-specific operating systems that support protein analytics out of the box [@Kajan:2013cb]. Climate analytics and reanalysis are also demonstrative of fields adopting cloud computing technology early in their history. @Schnase:2014dn describes the development of Climate Analytics as a Service, an effort to integrate data storage and high performance computing to perform data-proximal analytics[@Schnase:2014dn; @Schnase:2015wp].

Cloud services have also been used in the geosciences, and in ecological modeling problems specifically. @Yang:2011bd note that despite recent advances in computing, geoscientific problems are still limited by computational ability, including data volume bottlenecks, processing limitations, multi-user concurrency, and spatiotemporal velocity of data [@Yang:2011bd]. Many scholars suggest that the cloud provides a means of overcoming these challenges by leveraging distributed computational resources without increasing the carbon footprint or financial budget [@Yang:2011bd]. Several large spatial data infrastructure (SDI) projects are currently hosted on public clouds, though geospatial algorithms are more difficult to implement on public cloud infrastructure than on more traditional computing environments like grid computing [@Huang:2010us]. Numerical models, such as real-time dust storm forecasting, have seen significant performance increases when run on high performance cloud VMs [@Yang:2011iy]. Environmental models, when implemented in a consumption-oriented way, can also be run in the cloud [@Granell:2013ix]. @Candela:2013bl describe a novel 'hybrid' platform that supports SDM specifically, suggesting that a cloud-based approach can aid in data discovery and increase processing capabilities. OpenModeller [@deSouzaMunoz:2009bn], while not cloud-specific, offers a generic interface for running multiple SDM algorithms on a remote server.

Species Distribution Models

Species Distribution Models (SDMs) are a class of statistical models that quantify the relationships between a species and its environmental range determinants [@Svenning:2011jq]. While these models sometimes include mechanistic or process components, they most often refer to correlative models [@Elith:2009gj], using supervised statistical learning algorithms to approximate the functional relationship between species occurrence and its environmental or climatic covariates. Used in a variety of global change-related fields, the models have been shown to provide reliable estimates of climate-driven range shifts when compared to independent datasets [@Guisan:2006bz; @Guisan:2000tc]. With the widespread availability of statistical software and machine learning code libraries and lower barriers to access to environmental and occurrence data, the utilization

of these techniques has grown substantially in recent years [@Franklin:2010tn; @Svenning:2011jq], shown in Figure 4. Science and engineering as a whole had a 7% growth rate between 20013 and 2013 [@NationalScienceBoardUS:2016uv], while SDM literature far outpaced this with a 10.8% average growth rate over the same period [Don't know how or if to cite this].

SDMs work by approximating the species niche and then applying it to future scenarios. @Hutchinson:2016tg characterized a species' fundamental niche as an n-dimensional hypervolume that defines the environmental spaces where the intrinsic population growth rate of the species is positive [@Williams:2007iwa]. The realized niche describes the subset of environmental space that the species actually occupies at some point in time, and is smaller than the fundamental niche due to competing biotic interactions with other species. Most scholars argue that SDMs come close to approximating the species' realized niche [@Guisan:2000tc; @Soberon:2005vt; @Miller:2007br], though the inclusion of fossil data in the model fitting process can increase the likelihood that calibration captures the fundamental niche by exposing the model to states of the climate system not present on Earth today [@Veloz:2012jw].

SDMs rely on three important assumptions. As a fundamental justification for applying predictions across space and time, all SDMs assume niche conservatism, i.e., that the niche of species remains constant across all spaces and times [@Pearman:2008it]. @Thuiller:2008ena argues that SDM fitting with fossil data also lessens effect of assuming niche conservatism through time. Though speciation and evolution are not accounted for in the SDM paradigm, @Peterson:1999ff suggests that species typically demonstrate niche conservatism on multi-million year time scales. Second, SDMs rely on the assumption that species are at equilibrium with their environment, being present in all environmentally suitable areas while being absent from all unsuitable ones [@NoguesBravo:2009iva]. Given dispersal limitations and biotic interactions between species, this is rarely the case. @Svenning:2008gs showed that many European tree species are still limited by postglacial migrational lag. Finally, SDMs must account for extrapolation to novel and no-analog climates for which there is no modern or fossil data. Inductive learning is severely impacted when it is used to predict onto future cases that were not within the range of values provided in the training set. @Williams:2007iwa note the high likelihood of encountering novel and no-analog climates in the near future. Fitting the models with fossil data increases the likelihood that climatic assemblages will be included in the training data, however, given rapid and highly uncertain climate change, the problem of projecting models onto unseen climates is a major limit to their application.

Despite their strong assumptions, SDMs have been used for a wide variety of paleo and contemporary studies of geographic and environmental distribution. SDMs are very often used to confirm ecological hypotheses, comparing hindcast projections with the fossil record. In this context, SDMs have been used to support hypotheses on the extinction of Eurasian megafauna [@Anonymous:2008jc], identifying late-Pleistocene glacial refugia [@Waltari:2007gc; @Keppel:2011ft;

@Flojgaard:2009ha], and to assess the effect of post-glacial distributional limitations and biodiversity changes [@Svenning:2008gs]. SDMs are often combined with genetic, phylogeographic, and other methods to develop a complete assessment of a species biogeographical history [@Fritz:2013er]. In an anthropogenic climate change context, SDMs have been used to assess the effectiveness of modern reserve planning [@Araujo:2004fd], predict the distribution of both endangered [@Thuiller:2005fm] and invasive species[@Ficetola:2007bn; @Vaclavik:2009gr], and ecosystems [@Hamann:2006jf], and evaluate the effectiveness of conservation planning for the future [@LOISELLE:2003gh].

A Taxonomy of Species Distribution Models

SDMs range from simple algorithms that characterize a 'climate envelope' for a species [@Guisan:2000tc] to multivariate bayesian techniques that use Markov Chain Monte Carlo simulations (MCMC) to develop probability distributions around projections and parameters. While all have the same fundamental goal of characterizing responses to climatic gradients, [@Franklin:2010tn] notes a conceptually meaningful way of grouping modeling algorithms into data-driven and model-driven algorithms. The data-driven/model-driven dichotomy is introduced in @Hastie:2009up and often employed when differentiating between 'statistical' and 'machine learning' algorithms. I add the burgeoning set of methods that employ stochastic, probability-based Bayesian methods to this taxonomy due to their recent uptake and high accuracy. No individual method or class of methods has consistently outperformed any other [@Veloz:2012jw; @Elith:2006vt; @ARAUJO:2007ep], though many scholars have attempted to assess interclass variation [@Araujo:2006bi; @Elith:2006vt], and variation between different parameterizations of the same model class [@Thuiller:2008enb; @Veloz:2012jw; @ARAUJO:2007ep].

Supervised learning techniques map inputs to outputs by using a set of training examples, $T = (x_i, y_i), i = 1, 2, ..., n$, where both values are known, then approximating the the real relationship between the two f, with a function ,$\widehat{f}$, that minimizes a loss function based on the difference between the real and predicted value, $y_i - \widehat{f}(i)$. Each training example may be composed of a p-dimensional vector of predictors, $\boldsymbol{x_i} = x_{i1}, x_{i2}, ..., x_{ip}, p = 1, 2, ..., p$. Models can either make *a priori* assumptions about the form of the input-output relationship or adapt to fit any given matrix of training examples. The former class of models, model-driven models, exhibit low bias but high variance and can make poor predictions if the assumptions are not upheld. The latter, data-driven models, do not rely on any stringent assumptions about the form of the relationship, but any particular subregion depends on only a handful on input points, making the models highly sensitive to small changes in the input data [@Hastie:2009up].

Model-driven learners fit parametric statistical models to a dataset, making assumptions about how inputs and outputs are related, including linearity and error distribution. These models were the first to see substantial use

in SDM applications and continue to be widely used because of their strong statistical foundations and ability to realistically model ecological relationships [@Austin:2002vy]. These models include simple boxcar algorithms, which build multidimensional bounding boxes around species presence in environmental space [@Guisan:2000tc], as well as more complex methods such as generalized linear models [@Guisan:2002dc; @Vincent:1983uw]. Other model-driven techniques include variants of linear and logistic regression for abundance and presence-absence outputs [@Franklin:2010tn].

The increase in available computing power has spurred the development and application of non-parametric, data-driven learning algorithms. These models, have, in some cases, been shown to significantly out perform their model-driven counterparts [@Elith:2006vt]. These models include genetic algorithms [@Elith:2006vt], classification and regression trees [@Elith:2008el], artificial neural networks [@Hastie:2009up], support vector machines [@DRAKE:2006cp], and maximum entropy techniques [@Elith:2010cea; @Anonymous:2008kla]. Since its introduction in 2006, MaxEnt, a maximum entropy algorithm and Java-based runtime environment has seen widespread use in SDM and has demonstrated its ability to perform consistently even on small sample sizes [@Phillips:2006ffa; @Elith:2010cea; @Anonymous:2008kla]. Trends in the SDM literature suggest that MaxEnt is the most popular SDM method in use today, and appearing in over 20% of all SDM studies published after 2008. Recent evaluations of MaxEnt, however, claim that its performance may be questionable when compared with other SDM algorithms [@Fitzpatrick:2013cb]. Data-driven models can be more computationally intensive than their model-driven counterparts because they usually take at least two passes over the input dataset to process the data and build the model. Furthermore, data-driven learners are often combined with techniques like bagging, which builds a collection of models based on subsets of the input data, and boosting, which combines many weakly predictive models into a single, highly predictive ensemble. A common application of boosting in SDM is in boosted regression trees [@Elith:2008el; @Elith:2006vt], where stochastic gradient descent is used to sequentially combine many small regression trees together into an ensemble that minimizes model deviance.

Models in the third category utilize Bayesian methods to develop the relationship between environmental predictors and species presence. Advantages of the Bayesian approach include the ability to include prior ecological knowledge in model formulation [@Ellison:2004fj] and the ability to estimate model uncertainty without the need for bootstrapping procedures [@Dormann:2012cj; @Elith:2009gja]. With improved computational infrastructure and better MCMC sampling algorithms, Bayesian methods have become increasingly popular in recent years, and have been used in several recent SDM studies [@Hegel:2010gu]. @Golding:2016bt introduce SDMs that incorporate Gaussian processes, a flexible method that demonstrates both high predictive accuracy and ecologically sound predictions. While most model- and data-driven SDMs use only abiotic environmental factors as their predictors, using a joint probability distribution of all entities in an ecosystem, joint SDMs can model both the climatic range limita-

tions of a species and its biotic interactions with other species [@Clark:2014gp]. Though it can be challenging for ecologists trained in the frequentist perspective to transition to a Bayesian approach [@Ellison:2004fj; @Hegel:2010gu], some software packages are in development for implementing Bayesian models out-of-the-box in languages like R [e.g., @Vieilledent:2012ty]. Though MCMC methods are computationally very expensive, numerical approximations and analytical solutions can, when available, significantly reduce computational burden [@Golding:2016bt].

A review of recent literature suggests that the majority of ecologists are using nonparametric models for SDM. Of 100 randomly sampled papers from the most recent 4,000 citations in Web of Science that met the query "(Species Distribution Model*) OR (Ecological Niche Model*) OR (Habitat Suitability Model*)", the overwhelming majority utilized techniques that were data-driven. Out of 203 modeling experiments described, 38 were model-driven, 131 were data-drive, and 1 was Bayesian. 33 additional experiments used unsupervised clustering analyses that are not capable of predicting future presence. Figure 5 shows the distribution of the 47 different models, and their aggregation into the categories described above. Of all algorithms, MaxEnt was the most popular (64 instances). Models in the model-driven category included generalized linear models (15), logistic regression (5) and multiple linear regression (2). Data-driven modeling applications included boosted regression trees (16), generalized additive models (11), genetic algorithms (11), random forests (8), artificial neural nets (6), and multivariate adaptive regression splines (4).

Because of the overwhelming propensity of scholars to employ methods in the second category, I focus my analyses on this class of algorithms. Many authors have alluded to the limitations imposed by computational complexity, though few have estimated those limits precisely. @Elith:2006vt recorded the execution time of the runs they used in their often-cited review of novel SDM techniques, noting execution times of up to several weeks for some modeling algorithms [@Elith:2006vt]. Their longest running model is the genetic algorithms for ruleset production, which they claim took six weeks to converge. Other popular learning models, including boosted regression trees (80 h), generalized additive models (17h), generalized linear models (17h), and MaxEnt (2.75 h), were all shown to be extremely computationally intensive. The authors suggest that performance could be improved if model building was split over multiple processing cores. While processor speeds have increased since the 2006 analysis, models are still often built sequentially, unable to leverage multiple processors.

In some cases, methodological papers advise against large modeling efforts due to computational limitations. A 2009 review paper in *Trends in Ecology and Evolution* suggests that, when fitting a generalized linear mixed model (GLMM), if a user encounters insufficient computer memory or time limitations, the user should reduce model complexity, perhaps using a subset of the original dataset [@Bolker:2009cs]. Many authors warn of the computational expense of running SDMs, noting that "considerable computational capacity is necessary

for the development of models even for a single species" [@Peterson:2003gl]. @Thuiller:2008enb cautions that "limits to the broad application of this approach may be posed ... by the computational challenges encountered in the statistical fitting of complex models." While better computing infrastructure may alleviate some of this problem, the the computation intensity of SDMs can cause challenges that are difficult to resolve without reducing model complexity.

Algorithm Execution Time: Drivers and Measurement

Analyzing the constantly evolving and multifaceted dimensions of computer performance has posed problems for analysts and scholars since the advent of modern computing [@Nordhaus:2001th]. While some figures, such as millions/billions of floating point operations per seconds (MFLOPS/GFLOPS), are used in the supercomputing literature, a computer's performance depends on the way in which it is used [@Lilja:TcjNvdug], rendering such universal metrics incomparable. While it is difficult to effectively and meaningfully quantify a computer's performance, both empirical and theoretical methods of estimating algorithm execution time exist.

Theoretically, it is possible to determine the upper, lower, and average run times using asymptotic complexity analysis. In this exercise, the order of growth of an algorithm's runtime is determined as its input is increased to infinity, so that only first order terms are relevant [@Knuth:1976if]. Asymptotic analysis is useful, because the algorithm that is more efficient asymptotically will typically be the best choice for all but very small inputs [@Cormen:2009uw]. The time complexity of an algorithm is most often applied when considering an algorithm's scalability [@Goldsmith:2007jd]. An estimate of worst case run time (Big-O), can usually be obtained by inspecting the structure of the algorithm and counting how many operations are required when the inputs is sufficiently large [@Cormen:2009uw]. For any given input, however, particularly on real-world programs, the actual runtime will vary [@Cormen:2009uw; @Goldsmith:2007jd].

Empirical complexity studies have attempted to bridge the gap between asymptotic characterization and real programs. These studies use observations of algorithm runtime under different sized parameters and inputs to build models that predict the run time of future model applications and parameterizations. These techniques seek a method "with the generality of a big-O bound by measuring and statistically modelling [sic] the performance ... across many workloads" [@Goldsmith:2007jd]. @Brewer:1995fh describes an initial attempt to develop a statistical model for the run and compile time of algorithms in a C library. While most contemporary empirical runtime models use data-driven pattern recognition, linear regression between input size and execution time has been shown to perform well in some cases [@Fink:1998vg]. Empirical complexity models have recently become an important subfield of artificial intelligence and have important applications to algorithm selection [@Hutter:2014cia]. Algorithms for solving very difficult (NP-Hard) combinatorial problems, can exhibit high

variance between different problem instances. When a collection of different algorithms are available, empirical runtime modeling can be used to select the model that will most efficiently reach a solution [@LeytonBrown:2003tn; @Hutter:2014du]. @Hutter:2014cia outlines a comprehensive analysis of strategies and methods uses for empirical runtime models in the context of algorithm portfolio optimization. Parameterized algorithms can be treated the same way as nonparametric algorithms, by including model parameters as input features in the execution time model, "notwithstanding the fact that they describe the algorithm rather than the problem instance, and hence are directly controllable by the experimenter" [@Hutter:2014cia]. Nonlinear, tree based methods for empirical performance modeling, including random forests, were shown to be superior to other methods because of their ability to group similar inputs together and fit local responses, so that some large outliers do not interfere with the predictions of other groups [@Hutter:2014cia; @Hutter:2014du].

Concurrently running programs, operating system tasks, and other processes may affect the execution time of a real computer program at any point in time. Changes in dynamic system state are stochastic and can cause unpredictable, non-linear and non-additive changes in program runtime [@Kalibera:2013kh; @Lilja:TcjNvdug]. Random variation in system state makes deterministic statistical modeling of hardware's influence on execution time difficult. These variations are a result of the way in which memory access patterns differ in space and time when small changes are made to the operating system state, timing device, or algorithm and its inputs [@Lilja:TcjNvdug], and few attempts have been made to model them explicitly. @Kalibera:2013kh suggest that models based on benchmarked runtime may provide an accurate estimate of an upper bound of execution time, though due to potentially large, nondeterminstic, system-induced variance in empirical results, it is important to perform the benchmarking experiment many times. @Dongarra:1987br suggest that a failure to properly characterize the workload, running benchmarks that are too simplistic, or running benchmarks in inconsistent environments can lead to meaningless results.

Despite the challenges, several empirical runtime studies have attempted to include computer hardware's effects on algorithm runtime and have achieved highly accurate results. @Wu:2011es contend that the execution time of a program depends on the complexity of the algorithm and its input data, the static hardware configuration of the resource (e.g., amount and type of RAM, CPU clock rate), and the dynamic system state (e.g., number of processes competing for resources). @Sadjadi:2008fe model execution time as a linear combination of contributions from elements of the computer's hardware characteristics, though multivariate nonlinear systems may be more appropriate to capture some of the complex changes possible in the empirical data [@Wu:2011es].

Previous attempts to model runtimes have relied on the subset of computing components thought to directly affect performance. @Sadjadi:2008fe include CPU rate (GHz) and number of CPU cores, though other studies have also included

memory amount and type, buffer size, and CPU cache size [@Wu:2011es]. While increasing the clock rate of a CPU is nearly guaranteed to improve execution time, since it can increase the number of operations able to be processed per unit time, the number of CPU cores can also affect execution time by allowing multiple programs to execute in parallel. To take advantage of multiple processors, algorithms must be specifically designed to partition their execution across multiple cores. Moreover, the addition of processor cores will only improve performance up to a point, after which all benefits of load sharing will be outweighed by cross-core communication [@Gustafson:1988dh]. Computer memory can also affect a program's runtime by reducing the number of times a computer must retrieve data from the physical storage device (e.g., hard disk) and can improve the the amount of concurrent work able to be done on a machine [@Wu:2011es].

Theoretical Problem Formulation

To conceptualize the optimal configuration for an SDM modeling workflow, it is useful to conceptualize the modeling process using the following framework. Here, the workflow is presented as a series of steps that advance a user towards her goal of obtaining scientific insight from an SDM model. The use of computing resources is essential towards this end, and the cost of these resources is proportional to their power. Computing prices are set by an external computing provider. As a rational consumer, the modeler will wish to minimize her costs, in both time and money, while maintaining the maximum accuracy under given budgetary constraints.

1. Consider a pool of computing resources, H. As posited by [@Wu:2011es], at any time t, the effective processing power of H is related to both the static and dynamic configuration of its hardware and software. Thus,

   $$H(t) = H_static + H_dynamic + \zeta$$

   in which $H(t)$ is the effective processing power, H_static represents the static, hardware capabilities of the machine that do not change with time and $H_dynamic$ represent the portions of the system that do vary with time. Execution times can vary non-deterministically with hardware due to stochastic changes in system state, so ζ represents process uncertainty and natural system variability that cannot be included in the model.

2. Consumers of computing services are part of a market driven by supply and demand, and face a costs set by computing providers dictated by the effective computing power provided. Particularly when operating in the cloud computing paradigm, computing providers can be seen as a utility provider [@Foster:2008cl], while in a traditional desktop computing model, the provider of the computers can be seen as a typical rational producer. Figure [X] demonstrates an instance of a Google Cloud Computing Engine's cost surface as a function of memory and CPUs. Notice that as the hardware capabilities of the virtual machines increase (i.e., more CPUs, more memory), the cost to a user of provisioning these resources is increased.

3. Every user of a modeling application has a particular set of goals for using it in the first place [@Norman:1984fg]. We can conceptualize, for a given model, a finite set of use cases for that model that fall within the bounds of existing or expected use [@Carroll:1999hh; @Rosson:2002vj]. For example, consider a hypothetical scenario that could apply to a typical species distribution model user (use case adapted from [@Smith:2013cs]):

Jessica Smith is a land manager at Yellowstone National Park, interested in understanding how Mountain Pine Beetle infestations may change under different anthropogenic climate change scenarios. Dr. Smith primarily wishes to characterize how the beetle range might change under the three different IPCC emissions pathways [@Moss:2010en], rather than differences between algorithms or characterization of modeling uncertainty.

From this brief scenario, we are able to infer that the model goal is to model a single taxonomic group (*Dendroctonus ponderosae*), in a single area of known size (Yellowstone National Park, ~3,500 miˆ2), under three climate scenarios. She requires only a single modeling algorithm, as she is not interested in assessing the differences in the outputs from multiple SDM models.

To generalize this example, I introduce U – a vector of characteristics that fully describe the user's goals in the scenario. The components of U include a number of experiments to undertake, as well as user traits, such as experience with the model and familiarity with the interface employed, motivation, skill, and accuracy required. U also contains a list of experiments desired by the user, which specifies the number and character of modeling runs the users wants to do. Each element in the experiments list contains enough information to specify the inputs for a single model run, such as modeling algorithm, spatial resolution, amount of data to be used as training data, availability of environmental covariates, and number of past or future time periods to project onto.

4. The time to compute a given algorithm with a given set of inputs is proportional to $H(t)$, the effective computing power. However, in addition to computing the model, the user must also undertake a number of other pre- and post-processing steps. The total time elapsed during a modeling experiment can be expressed as

$$T_{model} = T_{Input} + T_{Prep} + T_{Compute} + T_{Output} + T_{Interp}$$

In this formulation, T_{Input} represents the portion of time that is spent by user gathering the resources needed to model. In a species distribution modeling context, this term represents the time needed to find and download occurrence points and find and download predictor variables. T_{Input} can be thought of as a function of computing resources available to the user (how fast can data be downloaded?) and the experiment (what is the data?). T_{Prep} is the time required by the modeler to prepare the data for entry into an algorithm. In this case, T_{Prep} time might include data cleaning, projection, and conversion, as well as setting up and configuring computing platforms and environments, like R. This component can

vary widely between modelers and between model applications, based on data source and quality, user skill and motivation, and the interface and equipment user has on hand. [@Elith:2006vt] notes the potential impact of how experienced a user is with a model on the modeling time and results. T_{Output} includes the time it takes to return the output from the computation to the user, which may be non-trivial if the model is run on a set of remote resources and the output must be downloaded over a network to reach the client's machine. Finally, T_{Interp} represents the amount of time spent by the user evaluating model output and determining whether her goals were met during the modeling process.

5. Single experiments can be combined together to form workflows, so that a user's time-to-goal for a workflow of N modeling experiments can be expressed as

$$T_g = \sum_{i=1}^{N} T_{Model}(Experiment_i, H(t))$$

6. Combining equations from (2) and (5), we find the total time cost of a modeling experiment is the sum of total time of spent modeling and the total monetary cost is the cost of provisioning computing resources for this time. Thus, we derive a multivariate cost function for a modeling scenario that takes into account both time spent modeling and the cost of provisioning resources: $$C(U, H(t))= f(T_g(U, H(t)), C_{Compute}(H(t)))

7. Each user-based scenario, U, will have its own cost curve that's subject to both the particular characteristics of the workflow and the specific cost surface imposed by the computing provider. If we select a single element from the finite set of all possible U's, and call it $U*$, we obtain a unique cost function for this set of activities that depends only on the computing resources used to fit the model. The cost surface, C, is defined for the set of all real computing solutions, however, some may be suboptimal for that particular experiment. Indeed, the optimal solution for $U*$ is multidimensional minimum of C that maximizes model accuracy subject to any constraints imposed by the user. Such constraints may include limited budget, minimum required accuracy, or maximum number of data points.

Without any user constraints, we can determine the unconstrained optimal configuration using a three step process. First, we conceptualize all inputs into the cost equation as orthogonal axes in p-dimensional space, where p is the number of characteristics in U plus the number of hardware variables in H. Using the model for computing t_g and a regular sampling method, we calculate the time to compute each experiment within a finite bounded hypercube. Because the computing resources, $H(t)$, should have no impact on the accuracy obtained from the SDM model, we can estimate the accuracy of the model $U*$, the algorithm inputs, for every experiment contained within the hypercube. We then search those results for the combination of U elements that will result in the highest accuracy.

Second, we slice the hypercube on this set of U values that will result in the highest accuracy model. We are left with a hypercube of reduced dimensionality, who axes are the orthogonal components of H. For each of potential combinations of H, which may be limited by the computing provider and physical reality, we calculate the cost of that experiment in both time and money, using T_g.

Finally, we put each of the points calculated in the second step onto two new orthogonal axes, time and money. The ideal model experiment would lie at the origin of these two axes: no time, no money. However, it is economically and physically unrealistic to think that any model will actually lie at this ideal origin: no computing time is free and every modeling experiment can be broken down into simple instruction sets that must be executed by the computer. Therefore, no matter how fast the computer is, some time will still be expended while executing these commands. Thus, the optimal experiment configuration is the one that lies closest to this unobtainable optimal point. We find this point by calculating the distance between the origin and all candidate resource sets, and select the candidate with the smallest distance to the origin.

If a user constraint is placed on the optimization *a priori*, the same technique can be applied to subset the hypercube to contain only candidate solutions that meet the user's criteria. For example, if the user has a minimum accuracy under which the results of the model are not acceptable, the first step proceeds as described above. The hypercube is then searched for all potential configurations that might result in an accuracy greater than or equal to the user defined threshold. If no such experiment exists, then the user must relax her constraints or adjust her modeling protocol. If one or more solutions exist, the hypercube is subset to include only these candidates. Then, the second and third steps of optimizing to maximize accuracy while minimizing cost proceeds.

Similarly, a user may place a constraint on the total amount of money or time may be spent on a modeling workflow. In this case, the user would then wish to identify the configuration that maximizes the accuracy while not exceeding the monetary to time threshold. First, the cost of all experiments in the hypercube is estimated using T_g. The set of candidate solutions is truncated to include only those that fall within the user's predefined limits, if any exist. Finally, the reduced-space hypercube is searched for the configuration that yields the highest accuracy.

Methods

Approach

My methods employ a relatively simple methodology to develop an empirical dataset that can be used to predict the optimal configuration for a SDM workflow. First, using the Google Cloud Computing Engine, a large number of SDM simulations were run under systematically varied inputs and on systematically different hardware. In total, I collected data for four SDM classes and approximately

2600 configurations, resulting in nearly 27,000 model simulations. The data was the used to fit predictive models for accuracy and execution time for each model class, using the characteristics of the configuration as features in model training. The models were validated using several metrics of model skill and interrogated for insight into the drivers of the runtime and accuracy for each SDM class. Finally, the models were used in associated with a large regular grid of potential configurations to form the hypercube, which is subsequently used to predict the optimal configuration for a given SDM scenario.

Limitations and Assumptions

This methodological approach has several important limitations. While a real SDM workflow contains many important terms other than the time and money spent modeling (the terms in the T_g equation above), I focus in this thesis on only the computing time in my development on the optimal modeling configuration. The addition of the time to gather and prepare the data and interpret the results are excluded here because they are highly variable based on user skill and motivation and can depend on many factors that may be impossible to model. Furthermore, I exclude the T_{output} and T_{input} terms, to keep this project to a manageable scope. While excluded terms other than the computing time improve the analytical tractability of this problem, this exclusion prevents the prediction of the true optimal value. Future work could be pointed towards modeling these additional factors so that they could be effectively incorporated into the optimal prediction model.

A second limitation of the approach described here is that my analysis is limited to virtual computing instances hosted on Google Cloud Computing Engine (GCE), rather than real-world physical machines. This experimental design was developed to add validity to the benchmarks of computing time, by providing them with a consistent environment unaffected by other tasks or concurrent programs [@Dongarra:1987br]. However, because real-world machines do have many concurrent and interacting processes running at any one time, this may bias the results as a source of inference for real-world workflows. It is difficult to characterize whether a particular user will use a dedicated physical machine for developing her models or, instead, use a single machine for both modeling and other purposes, running concurrent programs while waiting for an SDM to converge. While the use of dedicated virtual instances may fast-bias the results, it simplifies the problem into one that can be more easily modeled. Moreover, by using GCE, I am unable to systematically vary the CPU clock rate as a hardware variable. The runtime of any algorithm is directly related to the number of cycles a processor can complete in a given amount of time. By using GCE instances, I am limited to only the CPUs provided by Google, which may be updated or changed as they wish. At the current time, GCE provides only one processor type, a state-of-the-art 2.6 GHz Intel Xeon E5 processor. While I am not able to alter the clock rate experimentally, my results will not be biased by using machines with different CPU rates, as might be the case if I used physical

machines instead of virtual instances.

I limit my work to the analysis of model classes in the data-driven tier of SDMs, and do not attempt to characterize the computational time of either parametric or Bayesian approaches. Systematic literature review suggests that this is a reasonable simplification to make, because a majority of SDM users utilize these machine learning methods. Using the same logic, I limit my analyses to the `R` implementations of these SDMs, which ensures that all code libraries are well documented and open source. While there are known limitations to the language design and speed of `R`, the platform is the most widely used for SDM analyses. Maxent, the most popular modeling algorithm in recent years, is excluded because (1) it is written in Java, with only R bindings linking it to the R platform and (2) it is not open source, it is distributed as a black-box algorithm for SDM.

Finally, my methodological approach was strongly limited by computational cost, both in financial and time contexts. Each SDM configuration was tested between five and ten times to ensure robust results. Experiments in my set ranged from less than five seconds to greater than five hours. In order to gather enough data to develop a robust predictive model, I limited the number of very long running models. Similarly, I limited my experimentation on virtual servers with very high vCPU counts or memory allocations. The cost of these instances was more than an order of magnitude of larger than smaller instances ($> \$1$/hr), so experimentation was shifted to less costly servers. More data collected in all areas, particularly on virtual instances with high memory and many CPUs may improve the robustness of my results, particularly for very data-intensive models or severs with advanced hardware capabilities.

Data Collection

SDM Data Preparation

I systematically collected data on the execution time and accuracy of four SDM algorithms that have shown competitive accuracy results in the literature: multivariate adaptive regression splines (MARS) [@Leathwick:2006bd], gradient boosted regression trees (GBM-BRT)[@Elith:2008el; @Friedman:2001db; @Natekin:2013ji], generalized additive models (GAM) [@Yee:1991jb; @Guisan:2002dc], and Random Forests [@Breiman:X1hY-WAY; @Elith:2009dl]. All of the SDMs were run in the R statistical environment [@Rcore] with the standard packages for fitting these models. GBM-BRT tree models were fit using the `dismo` package version 1.1-1 [@Hijmans:2012ej], GAMs using the `gam` package, version 1.12 [@gam], MARS using the `earth` package version 4.4.4 [@earth], and random forests with the package `randomForest` [@rf]. These standard packages were chosen for their popularity in the field, and are designed to represent normal use cases in species distribution modeling

workflows, however, it is likely that the results are highly sensitive to specific implementation details.

Each SDM was fit using fossil pollen data obtained from the Neotoma Paleoecological Database in April 2016. All records for the genera *Picea* (spruce), *Quercus* (oak), *Tsuga* (Hemlock), and *Betula* (birch) were downloaded using the `neotoma R` package [@Goring:2015cr]. Global Quaternary occurrence records were filtered to only include those in the last 22 kya and located in North America. For each record, the latitude, longitude, age, and relative abundance of the taxon was retained and stored using a comma separated value format.

The SDM climatic covariates were downscaled and debiased Community Climate System Version 3 (CCSM3) model simulations for North America [@Lorenz:2016hu]. The post-processed model output was obtained in NetCDF format with a 0.5 degree spatial resolution and decadal temporal resolution for the last 22,000 years [@dryad_1597g_2]. Bioclimatic variables (BV) [@ODonnell:2012ww] were calculated for each timestep using the biovars function in the `dismo` R package [@Hijmans:2012ej]. For each fossil occurrence, BV values were extracted for that space-time location.

The occurrence-climate datasets were then filtered to include only the six least correlated BV predictors, a common practice when applying learning algorithms and SDMs. Collinearity among predictors can decrease model performance, can cause highly volatile model results, and "in truly extreme cases, prevent the numerical solution of a model" [@Obrien:2007ir]. The Variance Inflation Factor (VIF) was calculated and used to determine variable collinearity. VIF quantifies the expected amount of variance in a regression coefficient that is dues to collinearity in its predictors, lower bounded by 1 (no inflation) with no upper bound. Variance inflation was calculated using the `usdm` package. Based on the result of this analysis, I retained the six least intercorrelated variables, leaving a maximum correlation of 0.51. The variables I retained were BV2 (mean diurnal temperature range), BV7 (annual temperature range), BV8 (mean temperature of westtest quarter), BV15 (precipitation of warmest quarter), BV17 (precipitation of warmest quarter), and BV18 (precipitation of driest quarter).

Future climate layers for AD2100 were obtained from the CMIP project, HadCM3 climate model. These layers represent modeled climate variables under the UN IPCC RCP 8.5, a scenario that assumes high population, moderate economic growth, and a sustained dependence on fossil fuels [@Riahi:2011dk]. These layers were converted to bioclimatic variables and resampled to various resolutions for their use as output layers in different experiments.

Computing Infrastructure

I used the Google Cloud Compute Enginge (GCE) to complete all of my experiments. A popular Infrastructure-as-a-Service (IaaS) provider [@Hassan:2011uh], the platform provisions a wide array of virtual server instances, from the most

basic (1 CPU, 0.6 GB RAM) to exceptionally powerful (32 CPU, 208 GB RAM). The platform also provides a set of application programming interfaces (APIs) to promote workflow automation on the virtual instances, as well as a graphical user interface (GUI) for interactive resource provisioning.

Google's IaaS platform was chosen over other public cloud vendors because of its ability to create 'custom' instance types that adhear to user defined specifications. Other vendors (e.g., Amazon Web Services) provide a larger number of predefined instance types, some even more powerful than Google's top-end, but do not allow you to create an instance with an arbitrary number of processors and memory. By providing the ability to create custom types, Google's service fits well into my experimental design, and lets me avoid using software solutions to artificially alter hardware parameters.

I set up a distributed computing system to complete my experiments, featuring one centralized database node and multiple distributed computing nodes. Fault tolerance was important, because I utilized Google's less expensive 'preemptible' resources, which function like normal instances, but can be shutdown at any time if other customers require additional computational power. One Master Node hosts a MySQL database and a control script (written in python), and a pool of computing nodes that are fault tolerant and designed only for computing are provisioned and decommissioned as needed. The compute nodes, which run a Debian Linux operating system, are given only enough information to complete a given SDM, and the Master Node control script manages the progress of the project as a whole. A node.js script provides programmatic access to real-time database content and is used as the linkage between the master node and the worker nodes.

An outline of the system is described below and are illustrated in Figure [X].

1. First, a pool of computing nodes is assembled. The Master Node control script queries the central database for experiments that have yet to be completed or threw an error the last time they were run. The database responds, via the node API, with a JSON object that contains the hardware parameters required by the next experiment. The python script parses the response and uses the 'gcloud' tools associated with the GCE to create a pool of virtual instances that have the necessary amount of memory and number of CPUs.

2. Each node in the pool automatically begins running a startup script that begins the modeling process. First, a number of system-wide software packages, including R and Git are installed on the new instance. Git is used to clone the most recent version of the project repository (hosted as a private repository on GitHub) which contains all files necessary to compute an SDM. Once all packages have been installed, the timing script is initialized as a new R session. The R script queries the central database, identifying itself as a computing node with x cores and y memory. The database responds with the parameters needed to run a single experiment

on the given infrastructure. The script then loads the necessary variables and runs the SDM. When finished, it reports its results to the database and marks the experiment as completed. Experiments are continued until there are no more experiments that can be computed on an instance of x cores and y memory. If an instance is preempted by the system or otherwise crashes, a shutdown script will be executed, marking the in-progress experiment as interrupted – signaling other worker nodes that it should be attempted again later.

3. Because Google charges by the minute for the use of their virtual machines, instances must be torn down as soon as possible. As the computing nodes execute the experiments, the Master Node repeatedly polls the central database to determine the current position within the experiment table, determining the percentage completion of the current group of experiments. Once the group is complete, Master Node will use the gcloud tools to decomission the individual instances, and delete the instance pool and the instance template that was used to create each virtual worker node. After this, the Master Node is the only instance that remains online. At this point, Master Node returns to Step 1 to build a new pool of instances for a new hardware configuration.

SDM Model Protocol

The experimental parameters are communicated to the worker node by the central database. The computing node parses the database's response and starts a new experiment session. First, the set of occurrences corresponding with the species to be modeled is loaded from the disk. It is then randomly partitioned two disjoint subsets, a training set of N occurrences, and a testing set of 20% of the total number (*Picea* = 9935, *Quercus* = 8953, *Betula* = 10226, *Tsuga* = 7140). All examples were converted to binary presence-absence values using the Nieto-Lugilde [@NietoLugilde:2015bza] method for determining local presence from fossil pollen records. The training set is then sent to the specified SDM model where a learner is fit using p predictors. The model is then used to predict that taxon's range in 2100 AD under the RCP 8.5 scenario. The holdout testing set is used to evaluate the model's ability to discriminate presence-absence from the predictors. During the execution, separate times are recorded for model fitting, prediction onto the gridded surface, and accuracy calculation, as well as the total time, which is the sum of the three components. Furthermore, the Area Under the Receiver Operator Curve (AUC) accuracy metric, a popular method of evaluating logistic output (but see [@Lobo:2008du]), is computed and stored with the timing results. There is no database I/O inside of the timing script, so results should not be slow-biased by network connection or context switching. Learning parameters (learning rate, number of trees, tree complexity, etc) are held constant for all runs except a small subset which were designed specifically to determine sensitivity to these parameters. Timing was done within R using the `proc.time` function.

Due to budgetary constraints, and thus inability to cover all possible parameterizations, experiments were divided into several categories in which different variables were systematically altered to determine sensitivity. This compromise aims to capture as much within-parameter variance as possible while simultaneously capturing the influence of interactions between variables. One basic series of experiments was run for all SDMs using a small subset of algorithm inputs (training examples and memory) on a wide variety of VM types (core/memory combinations). Five additional, separate analyses were completed to determine the sensitivity to specific parameterizations. Because execution time can vary non-linearly when the hardware parameters are changed, I tested as many combinations of memory and CPUs as possible. On each computer, a standard set of 160 experiments were run for each sequential SDM (MARS, GBM-BRT, GAM), including four spatial extents, four training example sets, and 10 replicates of each. All experiments were done on the *Picea* data set.

Individual, target experiment sets were done to assess the contribution and sensitivity of individual or sets of parameterizations. While no theoretical difference would suggest that execution times should vary between different taxa, a set of 191 model runs were done to evaluate whether differences exist in practice. The model uses aspatial input sets, which suggests that geographic range, abundance, or taxon-specific patterns should not bias the results of the experiments. Using all four taxa on six different VM instance types, inter-taxonomic sensitivity was recorded. The number of training examples and spatial resolutions was held constant for these runs.

Empirical Performance Models theory suggests that specific algorithm parameterizations will take longer to execute than others [@Cannon:2007ge]. SDMs have a large number of potential parameters with which to alter, though many ecologists use the defaults, or packages that make it difficult to change the default parameter values (e.g., `dismo` [@Hijmans:2012ej]). To assess the magnitude of changes in execution time due to different parameterizations, the GBM-BRT was tested on a set of 70 different combinations of tree complexity and learning rate. The learning rate parameter of the GBM-BRT model is a shrinkage parameter that reduces the impact of each additional fitted tree, motivated by the boosting paradigm of fitting a model with many small models rather than fewer large trees. If one of the greedy iterations does not improve model fit, the contribution of that iteration can be easily reversed in the subsequent iterations [@Natekin:2013ji]. The tree complexity parameter controls whether interactions between predictors are fitted. If tree complexity is 1, the tree will be an additive model with no interactions. A tree complexity of N will produce a model with N-way interactions between variables[@Elith:2008el]. These two variables together control the total number of trees needed to fit the model [@Elith:2008el].

To assess the relative performance of parallel methods over sequential models, RF SDMs were fit both sequentially and in parallel on instances up to 24 CPU cores. Spatial resolution, memory, and taxon were held constant while number of

ensemble members and number of training examples were systematically altered for each core. In total, 3576 random forests were fit using the randomForest function with the foreach package providing parallelization support. Sequential runs were fit using the same function but with number of cores set to only 1.

In addition to performance gains made by increasing the number of CPU cores and leveraging parallel methods, increasing instance memory should improve performance for very large datasets. Using simulated datasets, I attempted to assess the performance of the model when faced with more than 1 million input examples. R has little support for high memory tasks, and these tests routinely crashed the computer when trying to fit the SDM due to inability to allocate memory space.

Finally, I evaluated the effect of varying the number of predictors on the execution time of the algorithm. The literature on theoretical complexity of algorithms (e.g., [@Hastie:2009up]) often characterize the complexity of machine learning algorithms in terms of both number of training examples and number of features in each example. I systematically modified the number of training examples between 1000 and 11000 and the number of predictors on all SDM classes, to get an understanding of how the two parameters interact. Because both of these algorithms run serially, they can safely be run on a single processor without the need for estimating the effects of additional cores.

Modeling Execution Time and Accuracy

To model algorithm execution time, I fit a random forest regression model for each SDM class using 80% of the data collected for that SDM. Tree based models, particularly random forests, have been previously shown to be highly effective in empirical performance models [@Hutter:2014cia]. The model's response variable is the log-transform total time of execution. Log transforms are used because they do not allow for negative predictions, which are possible under non-transformed inputs but physically impossible for runtime, and have been shown to be more accurate when observed responses span large ranges [@Hutter:2014cia]. The random forests were fit in R using the `randomForest` package [@Breiman:X1hY-WAY]. Each random forest had 100 ensemble members and tried three variables at each splitting point. In the MARS, GAM, and RF SDM classes, the predictor set contained five features: number of training examples, number of cells in the raster grid used for prediction, number of CPU cores, gigabytes of RAM, and number of environmental covariates. In the GBM-BRT SDM case, two additional features were used: tree complexity and learning rate of the algorithm.

SDM accuracy was modeled in the same way as execution time, using a random forest of 100 ensemble members and the same predictor sets. The response variable in this case is the area under the receiver operator curve (AUC), a measure of classification skill that ranges between 0 and 1.

Model Evaluation

Models were evaluated using a randomly selected holdout set of 20% of total data points. Using this testing set, the predictive accuracy of both the execution time and accuracy models were evaluated using mean squared error (MSE) and the model's r^2 value between observed and predicted values. Visual assessment of fit was done by plotting predicted values against observed values, which, if perfectly predicted, should fall along the x=y line.

Model drivers were investigated using the `importance` method of the randomForest package, which provides insights into how each predictor variable was used in the model fitting process. The function reports the mean reduction in MSE by including that variable in the fitted model. Plots were made to visually assess the important contributors for both time and accuracy models.

Optimal Prediction

To predict the optimal configuration, the prediction models for each SDM class were used to generate predictions of execution time and accuracy for a large set of potential experiments. For every predictor variable, potential values were randomly sampled from small values (near zero) to very large, but still reasonable, value. Each candidate experiment in the hypercube was assigned a dollar-per-hour rate using the Google Cloud Engine pricing scheme. Using this scheme allows the prediction to be recomputed should a new scheme become available. For every candidate scenario in the set, the execution time and accuracy was predicted using those algorithm inputs on that hardware. Finally, the execution time was multiplied by the dollar-per-hour rate to obtain a monetary cost for that experiment.

The candidate experiment generation was done using the expand.grid function in R and the model prediction was completed on a 16-core, 60 GB RAM instance on GCE. Predictions were spread across all 16 cores using the parallel package and the mclapply functionality provided in that package. Once each experiment in the hypercube had been predicted, it was possible to predict the optimal configuration under a variety of optimality criteria. For several demonstrative optimality conditions, the results set was sliced and analyzed to yield the optimal configuration for the give user.

Orthogonal Axis	Minimum Value	Maximum Value	Step Size	Number of Steps
Training Examples	0	500000	10000	51
Cells	10000	1000000	100000	10
Number of Covariates	1	5	1	5
CPU Cores	1	24	1	24
Memory (GB)	2	22	2	12
Learning Rate *(GBM-BRT Only)*	0.001	0.11	0.005	22
Tree Complexity *(GBM-BRT Only)*	1	5	1	5

Results

Model Performance

The model results varied by SDM class, though overall showed considerable skill in predicting both time and accuracy. Both the MARS and GBM-BRT models showed clear ability to accurately predict the runtimes in the holdout testing set. The GBM-BRT model had an MSE of 0.076 ln(seconds)^2 and an r^2 value of 0.954. The MARS model slightly exceeded this, with an MSE of 0.062 ln(seconds)^2 and an r^2 of 0.961. The GAM and RF SDM prediction models show slightly less predictive skill. The RF model had an MSE of 0.667 ln(seconds)^2 and an r^2 of 0.536. Similarly, the GAM model showed an MSE of 0.0126 ln(seconds)^2 and an r^2 of 0.523. The GAMs took significantly less time fit that the SDMs (<10s). One possible reason for their high variance is that there are low level computing processes that affect processes at these time scales that have less impact when working on the longer time scales of the other models (minutes to hours). A second reason why these models may show less predictive skill is that these models have fewer data points on which to fit the model. Both the GAM and RF training sets had only 2636 and 2861 data points, respectively. Perhaps the addition of more training data would enhance the model fit. Nonetheless, all four models showed a significant ability to predict execution times, with all four MSEs far less than 1 log-second. The models explained a significant portion of the variance in the dataset, suggesting that stochastic interactions from the computer system play a relatively minor role in the SDM runtime. Figure [x] shows the relative accuracy of each of the four SDM classes. Note the relative scales, and that some models take several hours to compute, while others converge within several seconds.

Model	nTrain	nTest	MSE	r^2
GBM-BRT	9256	2314	0.07257	0.9558
GAM	2636	659	0.01319	0.5063
MARS	6632	1657	0.06155	0.9614
RF	2861	715	0.68057	0.5298

The accuracy of the SDM is also well modeled using random forests. The model evaluation metrics are quite similar, suggesting that the model is well suited to predicting accuracy using the given training features. The MSE is approximately 0.0002 AUC points for all models, and the r^2 variance explained ranges between 0.85 and 0.93. Statistically, the RF and MARS models were tied for the best performing model, with an r^2 of 0.935 and and MSE of 0.000184. However, the visual fit of the MARS model is significantly better than the RF model. The fact that all models show such similar skill metrics is surprising. A potential explanation for this is that there is a minor variable that is currently unaccounted for that could provide the missing ~14% of variance explanation. However, spanning all SDM classes it is difficult to figure out what this feature

might be. Figure [x] shows the relative skill of each of the accuracy models. Notice that while the same

Model	nTrain	nTest	MSE	r^2
GBM-BRT	9256	2314	0.000235	0.8748
GAM	2636	659	0.000276	0.8507
MARS	6632	1657	0.000184	0.9350
RF	2861	715	0.000184	0.9347

Model Drivers

The models show similar drivers over all classes, with the exception of the GAM models. In GBM-BRT, RF, and MARS models, the number of training examples used in model fitting are the single most important variable in determining model runtime, followed by the number of predictors used. In the GAM model, it appears that model fitting is trivial, and that the majority of the time was spent projecting the model onto the future raster grid, yielding number of cells in the output raster as the most important variable. In GBM-BRT, additional two additional algorithm input variables were included: tree complexity and learning rate. These parameters control the number of trees that the algorithm should fit, and so should in theory influence run time. However, these variables appear relatively minor in relation to number of training examples and predictors. In the GBM-BRT, MARS, and GAM models, the hardware variables show very little influence in the execution time. This is to be expected, since they are fit sequentially, there are not able to benefit from additional cores. An analysis of the runtime logs indicates that the models are CPU bound, and that they run out of CPU capacity before they are able to be significantly affected by memory limitations. The RF model is designed to run in parallel, and shows that number of cores can be a driver of model runtime when the algorithm is designed to take advantage of them.

The accuracy models showed a similar pattern of drivers amongst all model classes. All models were most strongly driven by the inclusion of additional training data. The next most important driver was the number of environmental covariates included. All other variables appear to play a relatively minor role in model accuracy, as would be expected by theory. Interestingly, in the GBM-BRT model, learning rate and tree complexity do not increase accuracy significantly.

Discussion