NCAR | National Center for
UCAR | Atmospheric Research

Optimizing Genetic Algorithm Parameters for Atmospheric Carbon Monoxide Modeling

Meera Duggal
William Daniels
Dorit Hammerling
Rebecca Buchholz

NCAR Technical Notes
NCAR/TN-566+STR

National Center for
Atmospheric Research
P. O. Box 3000
Boulder, Colorado
80307-3000
www.ucar.edu

NCAR/TN-566+STR
NCAR Technical Note

2021-06

Optimizing Genetic Algorithm Parameters for Atmospheric Carbon Monoxide Modeling

Meera Duggal
Department of Applied Mathematics and Statistics
Colorado School of Mines, Golden, CO

William Daniels
Department of Applied Mathematics and Statistics
Colorado School of Mines, Golden, CO

Dorit Hammerling
Department of Applied Mathematics and Statistics
Colorado School of Mines, Golden, CO

Rebecca Buchholz
Atmospheric Chemistry Observation & Modeling Laboratory
National Center for Atmospheric Research, Boulder, CO

NCAR Laboratory
NCAR Division

NATIONAL CENTER FOR ATMOSPHERIC RESEARCH
P. O. Box 3000
BOULDER, COLORADO 80307-3000
ISSN Print Edition 2153-2397
ISSN Electronic Edition 2153-2400

Optimizing Genetic Algorithm Parameters for Atmospheric Carbon Monoxide Modeling

Meera Duggal*[1], William Daniels[1], Dorit Hammerling [1], and Rebecca Buchholz[2]

[1]Department of Applied Mathematics and Statistics, Colorado School of Mines, Golden CO
[2]Atmospheric Chemistry Observation & Modeling Laboratory,
National Center for Atmospheric Research, Boulder CO

June 7, 2021

Abstract

A main source of atmospheric carbon monoxide (CO) variability in the Southern Hemisphere is large burn events, making CO a useful proxy for fires. Therefore, predictive CO models over fire regions can help countries prepare for unusually large fire seasons. Fires are related to the climate through fuel dryness and availability, both of which respond to variability in the climate. Climate indices are metrics that summarize climate variability through changes in sea surface temperature and wind. In previous work, we developed a multiple linear regression model that uses these climate indices to predict atmospheric CO and created the R package *regClimateChem* to perform variable selection. This package offers three different variable selection techniques: stepwise selection, a genetic algorithm, and an exhaustive search. The exhaustive search always finds the best possible model but is computationally expensive. Stepwise selection runs quickly and is scalable but often fails to find the best model. We implement a genetic algorithm as a potential compromise between computational expense and model accuracy. As a stochastic variable selection technique, the genetic algorithm has many parameters that affect the stopping criterion, frequency of the model modification techniques, and population size. Here we present a parameter optimization study for the genetic algorithm, seeking to balance computational expense and model quality. When considering models with four covariates, we find that the optimized genetic algorithm parameters result in a runtime reduction of 11.8% and only compromise 0.3% accuracy compared to the default settings. We then consider models with five covariates using a high-performance computing system. For models with five covariates, we find that the optimized population size becomes close to the total number of models, meaning it behaves similarly to the exhaustive method.

Keywords: carbon monoxide, climate chemistry, multiple linear regression, variable selection, genetic algorithm, optimization

*mduggal@mines.edu

Contents

1 Introduction

1.1 Motivation

In the Southern Hemisphere, large burn events are a main source of atmospheric carbon monoxide (CO) variability [1, 2]. As a result, atmospheric CO can be used as a proxy for fire intensity [3]. An early warning system for unusually intense fire seasons would have many public safety benefits. For example, it would give governments time to stock up on masks, recruit more volunteer fire fighters, and warn the public.

In a previous study, we used CO data collected through the Measurements Of Pollution In The Troposphere (MOPITT) instrument onboard the Terra satellite [4]. MOPITT has been gathering data since 2000 and uses the optimal estimation retrieval approach [5]. The data are available at https://asdc.larc.nasa.gov/project/MOPITT. In order to minimize systematic and random error we selected daytime, land-only retrievals from the thermal infrared version 7 product [3, 6]. Anomalies were created by subtracting a spatial and climatological average of monthly CO from 2001 to 2016 from monthly average values. See Buchholz et al. [3] for details. There are 7 different regions that are of interest in the Southern Hemisphere: Maritime Southeast Asia (SEA), North Australasia, South Australasia, Central Southern Africa, South Southern Africa, Central South America, and Southern South America [3]. Figure 1 shows the total column CO concentrations and the anomaly produced for a specific region, Maritime SEA. For this study we solely focus on the Maritime SEA region.

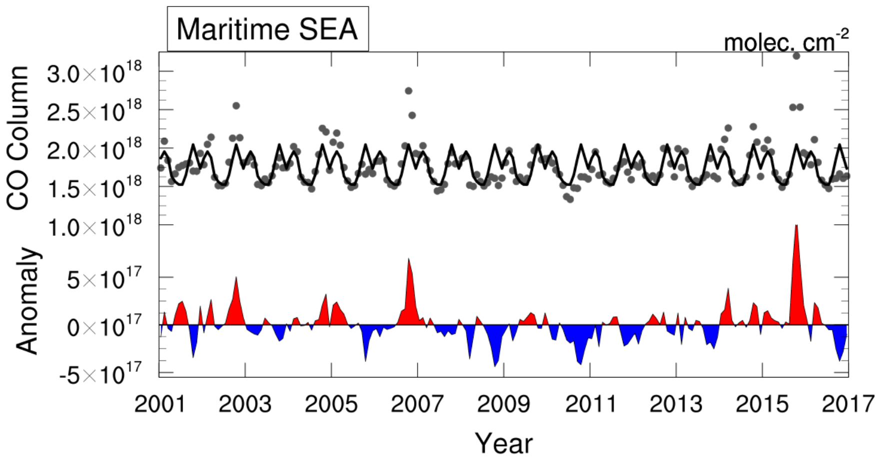


Figure 1: This Figure is from Buchholz et al. [3]. The plot on the top shows the monthly average of total column CO in the Maritime SEA region (grey dots) with the climatological seasonal cycle (black line) [3]. The plot on the bottom shows both negative and positive CO anomalies.

Large burns events are directly related to the availability of biomass, dryness of vegetation, and lack of water. These conditions are tied to natural variability in the climate. Climate indices are useful metrics that describe this natural variability through air temperature, sea surface temperature, air pressure, and other measurable fields. To connect atmospheric CO and climate variability, Buchholz et al. [3] uses four different climate indices — Niño 3.4, DMI, TSA, AAO — as predictors in a multiple linear regression model. Figure 2 shows the climate indices from 2001 to 2017, the time span used in this study. Our study aims to expand on the analysis performed in Buchholz et al. [3] by exploring multiple methods of variable selection and identifying the optimal parameter configuration for the genetic algorithm.

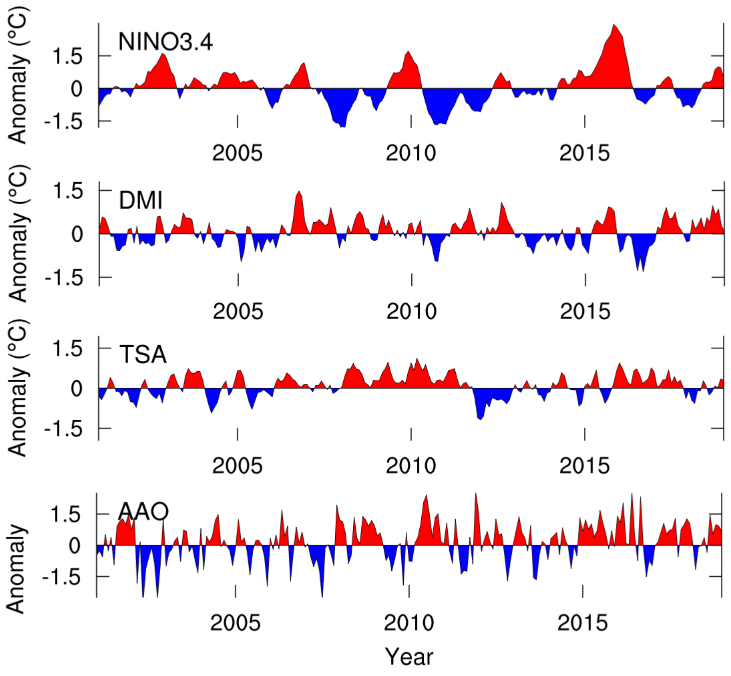


Figure 2: This figure is from Buchholz et al. [3]. Shown are the four different climate indices over the time period of our study. Positive anomalies are shown in red and negative anomalies are shown in blue.

1.2 Statistical Model

In a previous study, we developed a multiple linear regression model to model the total column CO seen by MOPITT [4]. These models can be used for predictive purposes. The response variable in our model is the carbon monoxide anomalies and the predictor variables are the climate indices. The climate indices are lagged from 1 to 8 months because our model is intended for prediction. Each possible combination of lag values (one lag for each index) is called a "lagset."

For a given region in the Southern Hemisphere, the multiple linear regression model can be described by equation (1).

$$CO(t) = \mu + \sum_{k} a_k \cdot \chi_k(t - \tau_k) + \sum_{i,j} b_{ij} \cdot \chi_i(t - \tau_i) \cdot \chi_j(t - \tau_j) \tag{1}$$

In equation (1), $CO(t)$ is the CO anomaly at time t in the given response region, χ are the climate indices, τ is the lag value for each index in months, μ is the constant mean displacement, and a_k and b_{ij} are coefficients. χ_k are the main effects and χ_i and χ_j are the interactions. The lag value, once chosen for an index, will remain the same for both the main and interaction terms.

1.3 *regClimateChem* R Package

In a previous study, we created the R package *regClimateChem* to perform variable selection for the model described in equation (1) [7]. The *regClimateChem* package provides three variable selection techniques: exhaustive, stepwise, and the genetic algortihm. The genetic and exhaustive search are implemented using the *glmulti* package, and stepwise selection is implemented via the *MASS* package [8] [9].

We use the Bayesian Information Criterion (BIC) to compare the models selected by these three variable selection techniques. We compare models using the BIC because it imposes a relatively harsh penalty on the number of terms in the selected model and results in models that do not overfit the data. This results in models that are well suited for prediction. The equation for the BIC is

$$\text{BIC} = \ln(n)k - 2\ln(\hat{L}),$$

where n is the number of observations, k is the number of covariates, and $\hat{L}$ is the maximized value of the likelihood function. Lower BIC values correspond to better models. The first term in the BIC acts as a penalty for complexity and the second term measures how well the model fits the data. The BIC is well suited for prediction because of how harsh the complexity penalty is. For the BIC, $\ln(n)$ is used as a coefficient on the penalty term, where n is the number of observations. Comparatively, the Akaike Information Criterion (AIC) penalty uses a coefficient of 2 instead, making the penalty not as harsh.

Stepwise selection is an iterative process that begins with either the null model or the full model. With each iteration, the algorithm considers removing or adding a predictor to minimize the BIC. The algorithm selects the model that minimizes the BIC. The process stops once adding or taking out a term no longer decreases the BIC. In comparison, the exhaustive method computes the BIC value for all possible models to find the best one. The genetic algorithm uses different model modification techniques to find the best model, and will be explained in more detail in Section 2.

Daniels et al. [7] found that with only four predictor variables, the runtime of the genetic algorithm is very similar to that of the exhaustive search in certain scenarios. Our goal is to optimize the genetic algorithm such that the runtime is minimized, while still producing good models. The genetic algorithm implementation in the *glmulti* package has many parameters that can be adjusted to fit the needs of the particular use case. Here we present an optimization study to find the best *glmulti* parameters for the atmospheric CO application. By optimization, we refer to minimizing runtime while maximizing model performance.

2 Genetic Algorithm

2.1 Introduction

The genetic algorithm is implemented in *regClimateChem* via the *glmulti* package in R. We use the scenarios studied in Daniels et al. to better compare to previous work [7]. In this study, the genetic algorithm was found to be less scalable but more accurate, meaning it usually finds better models than stepwise selection. The genetic algorithm is stochastic and therefore uses probability to converge on a final model.

We now discuss the genetic algorithm in depth. A **population** of models is randomly initialized to begin an iterative process. The genetic algorithm modifies this population using three different techniques (described in detail later) to create a subsequent **generation** of this population. The number of models in the population is held constant through each successive generation. The algorithm continues to produce new generations until a stopping criterion is satisfied. Although the algorithm is probabilistic in nature, it is designed to improve the population of models with each successive generation. This is done by creating future generations based on the best models in the current generation. In our study, models are assessed using the BIC. Models in the current generation (called **parent models**) are used to create new models in the next generation (called **child models**). A general flow of the genetic algorithm is shown in Figure 3.

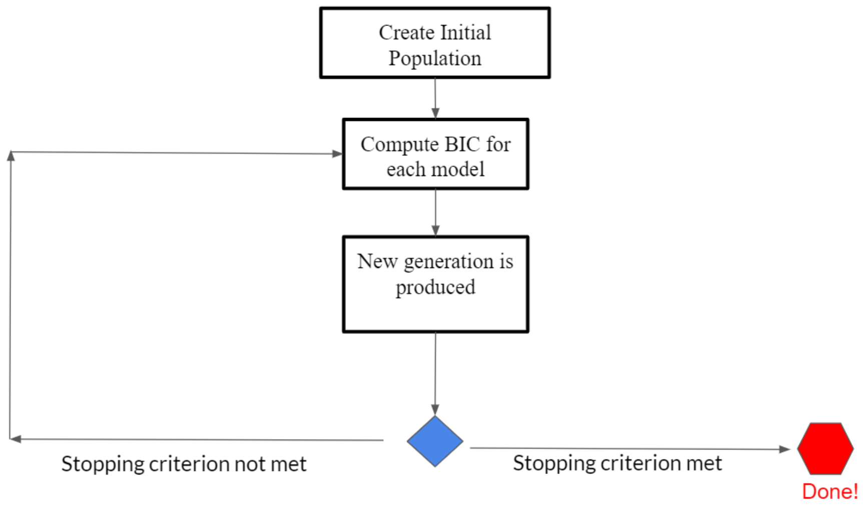

Create Initial Population
Compute BIC for each model
New generation is produced
Stopping criterion not met
Stopping criterion met
Done!

Figure 3: A general flow of the genetic algorithm.

The genetic algorithm uses a binary system to represent terms in the model. For each model, a term that is present is represented with a 1 and a term that is not present is represented with a 0. An example of this representation can be seen in Figure 4.

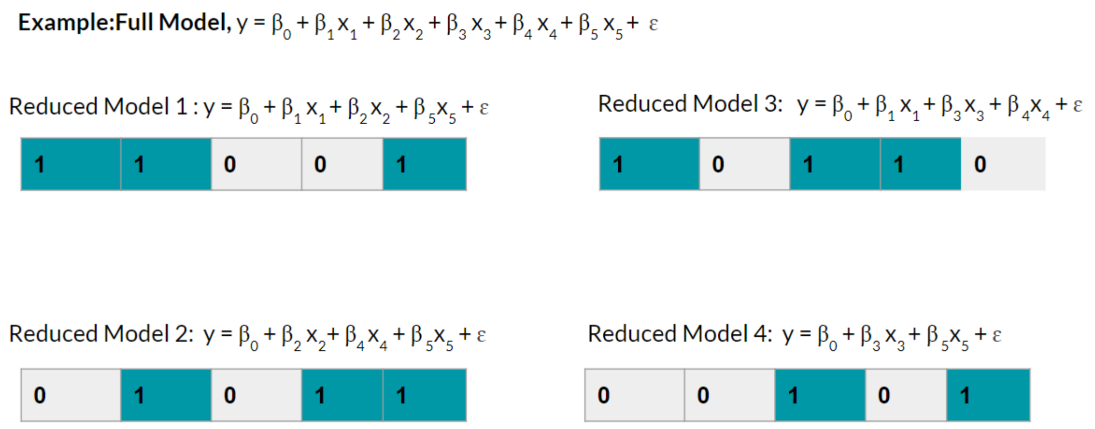

Example:Full Model, y = β0 + β1x1 + β2x2 + β3x3 + β4x4 + β5x5 + ε
Reduced Model 1 : y = β0 + β1x1 + β2x2 + β5x5 + ε
1 1 0 0 1
Reduced Model 3: y = β0 + β1x1 + β3x3 + β4x4 + ε
1 0 1 1 0
Reduced Model 2: y = β0 + β2x2 + β4x4 + β5x5 + ε
0 1 0 1 1
Reduced Model 4: y = β0 + β3x3 + β5x5 + ε
0 0 1 0 1

Figure 4: Example of the genetic algorithm binary model representation. A constant term (β_0) is always included. For brevity, a simplified model is demonstrated in this example with no interaction terms present.

2.2 Model Modification Techniques

In the genetic algorithm, the current generation of models is modified using three different techniques to create the next generation of models. These techniques are:

- Asexual Reproduction
- Sexual Reproduction
- Immigration

With asexual reproduction, each term in the model has the chance to mutate at a given probability called the mutation rate. When a term mutates, its binary representation switches. That is, if an included term mutates, it will no longer be included, and vice versa. A depiction of this process can be seen in Figure 5.

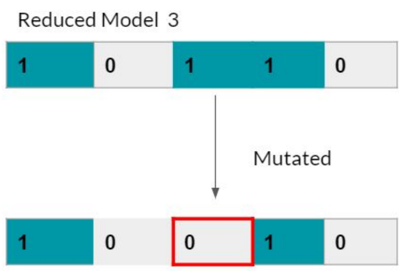


Figure 5: Example of asexual reproduction in the genetic algorithm

With sexual reproduction, two parent models are chosen to produce children models that will become a part of the next generation. The selection of parent models is biased according to the BIC. This means that "better" models are more likely to be selected as parent models. This makes future generations more likely to contain the best possible model. Each term in the child model has a 50% chance of coming from each parent model. An example of this modification method is shown in Figure 6.

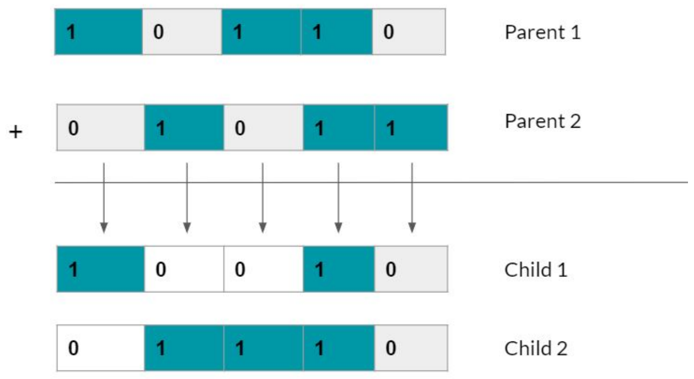


Figure 6: Example of sexual reproduction in the genetic algorithm. Terms that are included in both parents are always included in the child model, otherwise there is a 50% chance the term comes from parent 1 or 2.

The last model modification technique is immigration. Immigration introduces more variability into future generations compared to the other modification techniques. Immigration does not involve a parent model. When a model is produced via immigration, each term has an equal probability of being included or excluded. This completely randomizes the new model. This process allows for a wider range of models to be introduced and prevents the algorithm from getting "stuck" in local minima of the BIC.

2.3 Relative Frequency of Modification Techniques

The relative frequency of the three modification techniques is set by two *glmulti* parameters: `imm` and `sexrate`. `Imm` stands for the rate of immigration and `sexrate` is the rate of sexual reproduction.

The parameter `imm` represents the proportion of models that go through immigration relative to the number of models that go through asexual reproduction. The parameter `sexrate` represents the proportion of models that go through sexual reproduction relative to the number of models that go through asexual reproduction. The default rate for `imm` and `sexrate` are 0.3 and 0.1 respectively.

2.4 Stopping Criterion

The stopping criterion in the genetic algorithm is checked every 20 generations. The stopping criterion is made up of three different parameters: `deltaM`, `deltaB`, and `conseq`. The algorithm stops (i.e. converges) once seperate criteria for `deltaM`, `deltaB`, and `conseq` are met. `DeltaM` is the target change in the mean information criterion between the models in the current generation and the models in the previous generation. `DeltaB` is the target change in the best information criterion between models in the current generation and models in the previous generation. These criteria are met when the mean and best BIC of the current generation no longer decrease by `deltaM` and `deltaB`, respectfully. `Conseq` is the number of iterations that the genetic algorithm will run through once `deltaM` and `deltaB` are both satisfied. Larger values of `conseq` increase the probability that the best model is found when the algorithm converges.

3 Optimization Study

3.1 Overview

We vary *glmulti* parameters in order to find the parameter combination that is optimal in terms of both runtime and model accuracy. Runtime refers to the total clock time it takes for the genetic algorithm to converge. Model accuracy refers to the ability of the genetic algorithm to reproduce the best models found by the exhaustive search. To quantify model accuracy, we look at the proportion of models that are different between the genetic algorithm and the exhaustive search when considering all lagsets, which we call the proportion of differences. For the following study, each parameter value was varied independently, keeping the other parameters fixed at their default value.

3.2 *glmulti* Parameters

The parameters that we have chosen to consider in our optimization study are listed in Table 1:

Table 1: Each genetic algorithm parameter we vary is shown, along with the default and different values that are studied.

Parameter	**Default**	**Values Studied**
Population Size	100	$5, 20, 40, 60, 80, 100$
Mutation Rate	0.001	$10^{-5}, 0.001, 0.2$
Sexual Reproduction Rate	0.1	$0.001, 0.1, 0.7$
Immigration Rate	0.3	$0.001, 0.3, 0.7$
Consecutive Iterations	5	$1, 2, 3, 4, 5$

4 Four Covariate Case

We begin our optimization study with a four-covariate model, to better compare with models examined in Buchholz et al. [3]. To quantify uncertainty in our results, we ran the simulation 5 times at each parameter test value holding all other values fixed. In the following figures, the boxplots summarize the proportion of differences across the five trials at each parameter value. In Section 5, we consider a five covariate model to test if these results scale with model size. Note that the y-axis for the following plots differ, as the proportion of difference values span a large range.

4.1 Population Size

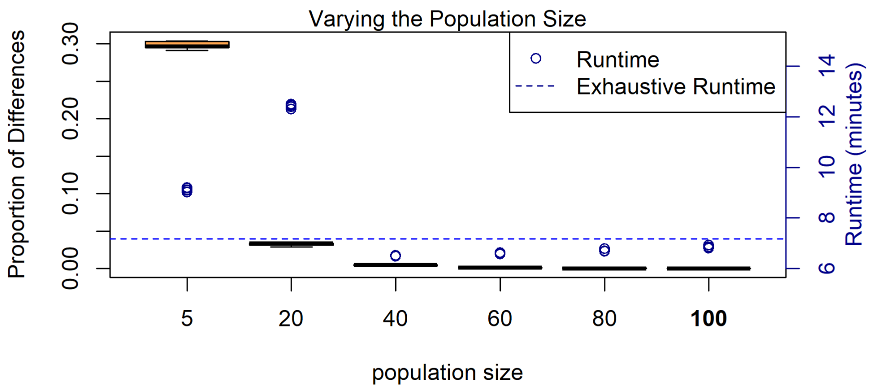


Figure 7: Results from varying population size. Default value of population size is boldfaced on the x-axis. Proportion of differences between the genetic algorithm and the exhaustive search are presented as box plots and correspond to the left vertical axis. Runtimes are plotted as blue circles and the exhaustive runtime is shown as a horizontal line which corresponds to the right vertical axis.

For the population size, we only studied values below the default of 100 because the total number of possible models in the four covariate case is 113. With a population size greater than 100, the genetic algorithm would essentially be performing an exhaustive search, as each population of models would contain nearly all of the possible models. A population size of 40 decreases the runtime compared to the exhaustive search, and runtimes increase for values larger than 40. The population size of 40 minimizes the runtime over the other values tested, and it also results in one of the smallest proportion of difference values. Therefore, we select 40 as the optimal population size for the four covariate case where the total number of possible models is 113.

4.2 Immigration Rate

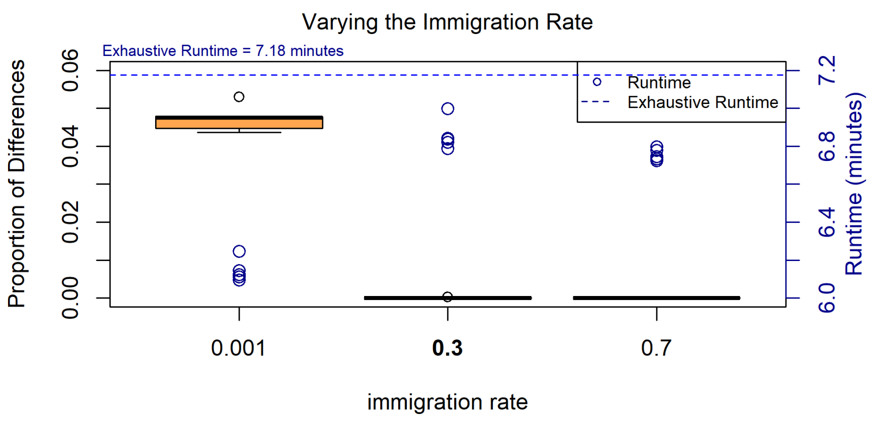


Figure 8: Results from varying the immigration rate. Default value of the immigration rate is boldfaced on the x-axis. Proportion of differences between the genetic algorithm and the exhaustive search are presented as box plots and correspond to the left vertical axis. Runtimes are plotted as blue circles and the exhaustive runtime is shown as a horizontal line which corresponds to the right vertical axis.

Figure 8 shows that the immigration rates greater than 0.3 result in longer runtimes compared to 0.001. Although the proportion of differences at an immigration rate of 0.001 might appear high, it is actually quite low considering the scale of the y-axis. For these reasons, we choose 0.001 as the optimal value for immigration rate.

4.3 Sexual Reproduction Rate

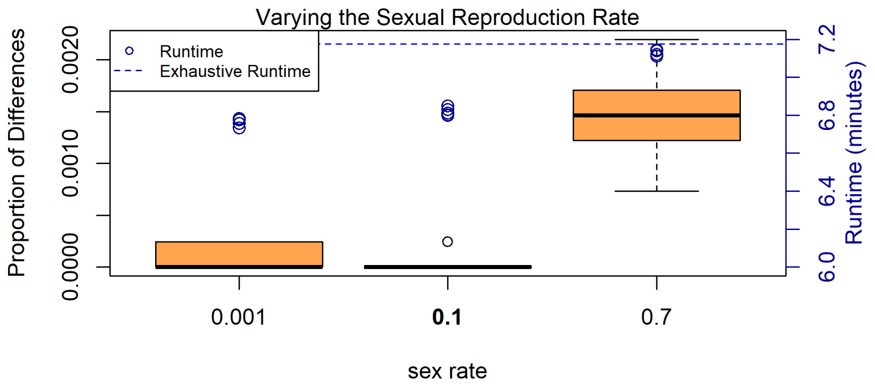


Figure 9: Results from varying the sexual reproduction rate. Default value of the sexual reproduction rate is boldfaced on the x-axis. Proportion of differences between the genetic algorithm and the exhaustive search are presented as box plots and correspond to the left vertical axis. Runtimes are plotted as blue circles and the exhaustive runtime is shown as a horizontal line which corresponds to the right vertical axis.

The default sexual reproduction rate of 0.1 minimizes the proportion of differences. Furthermore, there are only slight improvements in runtime when switching to a value of 0.001. Therefore, we select 0.1 as the optimal sexual reproduction rate parameter.

4.4 Mutation Rate

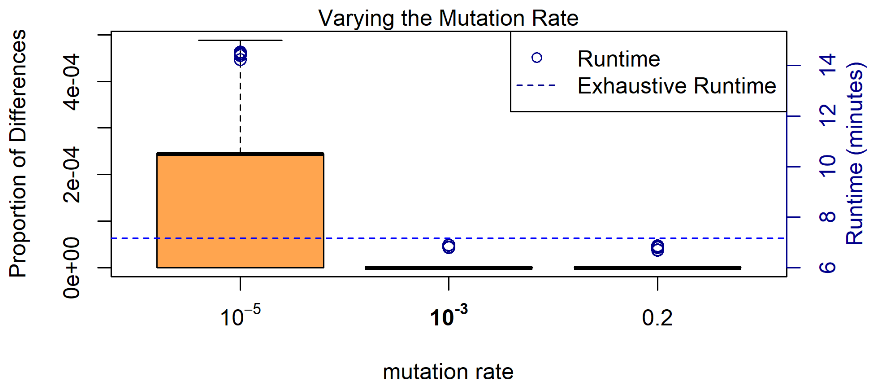


Figure 10: Results from varying the mutation rate. Default value of mutation rate is boldfaced on the x-axis. Proportion of differences between the genetic algorithm and the exhaustive search are presented as box plots and correspond to the left vertical axis. Runtimes are plotted as blue circles and the exhaustive runtime is shown as a horizontal line which corresponds to the right vertical axis.

A mutation rate below the default of 10^{-3} drastically increases runtime while leaving the proportion of differences largely unchanged because the y-axis is on such a small scale. Increasing the rate leaves the proportion of differences at essentially zero while slightly decreasing the runtime. So, we select 0.2 as the optimized mutation rate parameter.

4.5 Consecutive Value

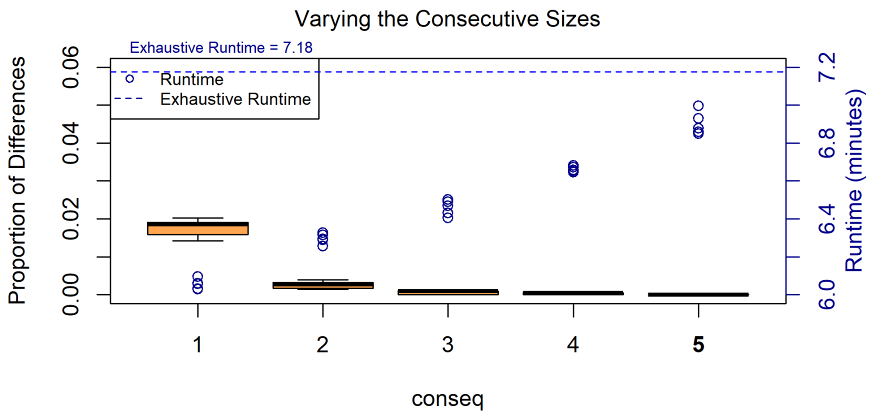


Figure 11: Results from varying the consecutive size. Default value of consecutive size is boldfaced on the x-axis. Proportion of differences between the genetic algorithm and the exhaustive search are presented as box plots and correspond to the left vertical axis. Runtimes are plotted as blue circles and the exhaustive runtime is shown as a horizontal line which corresponds to the right vertical axis.

Increasing the consecutive value increases the runtime linearly. This is expected because the consecutive value determines how many additional generations the algorithm will produce once *deltaB* and *deltaM* conditions are satisfied. Each additional generation adds to the overall runtime of the algorithm in a linear manner. All consecutive values that are listed have a proportion of differences that is less than 2%. We choose 2 as the optimal value because the proportion of differences is almost zero and it has a lower runtime than the default value.

4.6 Discussion of Results

The optimized parameters in the four covariate case are as follows with the default in parenthesis.

- Population size: 40 (100)
- Mutation rate: 0.2 (0.001)
- Sexual reproduction rate: 0.1 (0.1)
- Immigration rate: 0.001 (0.3)
- Consecutive size: 2 (5)

In Figure 12, we directly compare the default *glmulti* parameter values to the optimized values.

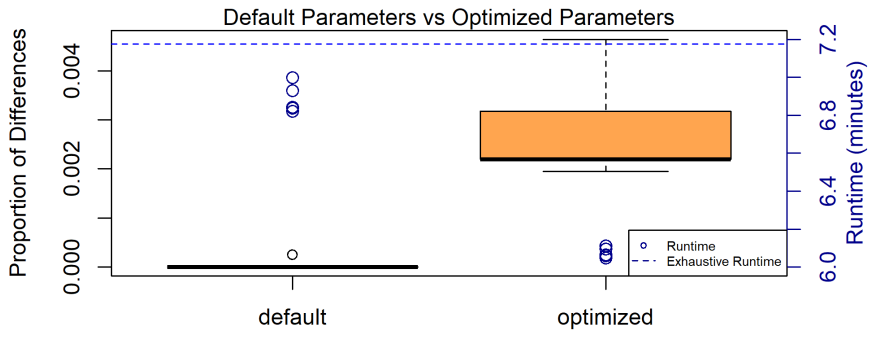


Figure 12: Comparison of default glmulti parameters to optimized parameters.

The optimized values decrease the runtime an average of 11.8% relative to the default values. The exhaustive runtime was 7.18 minutes and our optimized genetic algorithm is on average 6.06 minutes. The proportion of differences only decreases by an average of 0.28% between the default and optimized parameters. These results are for the four covariate case only. With only four covariates, the runtimes are already very short. Therefore, the time savings from this optimization are small. In Section 5, we examine the five covariate case, which will potentially have more room for optimization gains, as there are many more possible models.

5 Five Covariate Case

5.1 Varying a Single Parameter at a Time on a Personal Computer

To determine if our results scale to larger models, we consider a similar optimization study with five covariate models instead of four covariate models. The five covariate model includes the same climate indices as the four covariate model but with one interaction term. We began by varying the five *glmulti* parameters individually using a high, middle, and low value. This was done to select appropriate parameter ranges which we would study in more detail on a high-performance computing (HPC) system. This initial study was performed on a personal computer with an Intel Core i7 7th generation, 12 GB RAM, Windows 10, running 64 bit R, on R version 3.6.0. Due to longer runtimes, each different parameter value was only run once. Therefore, we use a bar graph to represent the single value rather than the box plots from the previous section. The proportion of differences is defined the same as in the four covariate case: the proportion of models that differ from the exhaustive method sorted by lagset. Note that the number of possible models in the five covariate case is 1,450, as opposed to 113 for the four covariate case. This is why we test larger population sizes, as described in the following section.

5.1.1 Population Size

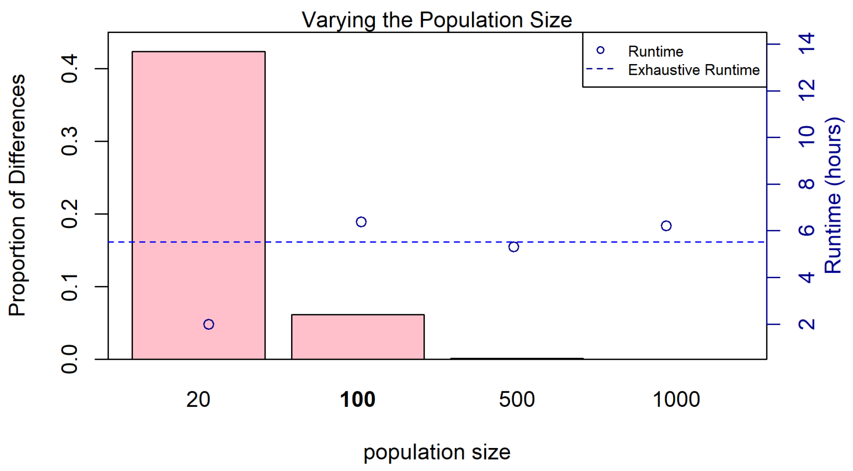


Figure 13: Proportion of differences (box plot corresponding the left vertical axis) and runtime (blue circles corresponding to right vertical axis) for the population sizes considered. glmulti default of 100 is bolded.

Using a population size of 20 with 5 covariates results in a very high proportion of differences. This could be because with such a small population size the algorithm will only be able to test a small percent of the total models at a time. The low runtime at a population size of 20 could be a result of the algorithm converging to local optima, which is again likely because of the relatively small number of models in each generation. We can see that the proportion of differences decreases to essentially zero as population size increases. This could be because the genetic algorithm converges to the exhaustive search as the population size approaches the total number of possible models.

5.1.2 Immigration Rate

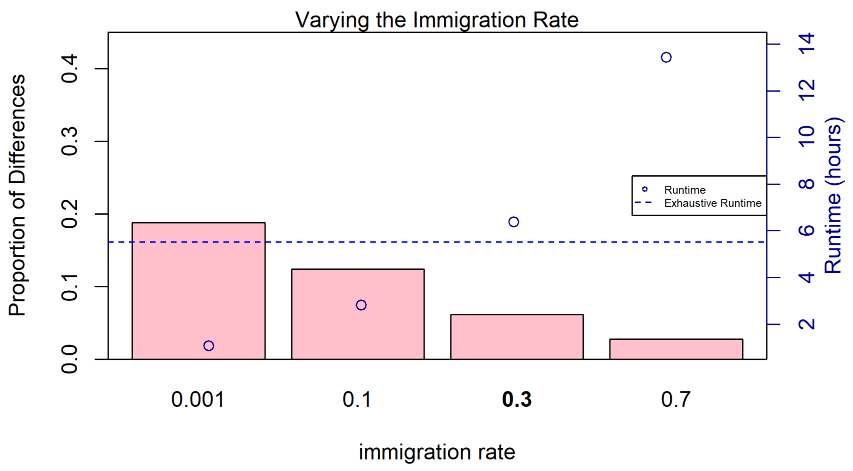


Figure 14: Proportion of differences (box plot corresponding the left vertical axis) and runtime (blue circles corresponding to right vertical axis) for the immigration rates considered. glmulti default of 0.3 is bolded.

The immigration rate is positively correlated to runtime. Immigration introduces the highest amount of variation in the population of models. Based on the runtime results, we can therefore hypothesize that increased variability in the population of models increases runtime. This could be because the increased variability in the models makes it harder to meet the *deltaM* and *deltaB* stopping criterion. Larger immigration rates (and hence more variability in the population of modes) also results in smaller proportion of differences. This could be because the increased variability ensures that the algorithm does not converge to a local optima.

5.1.3 Sexual Reproduction Rate

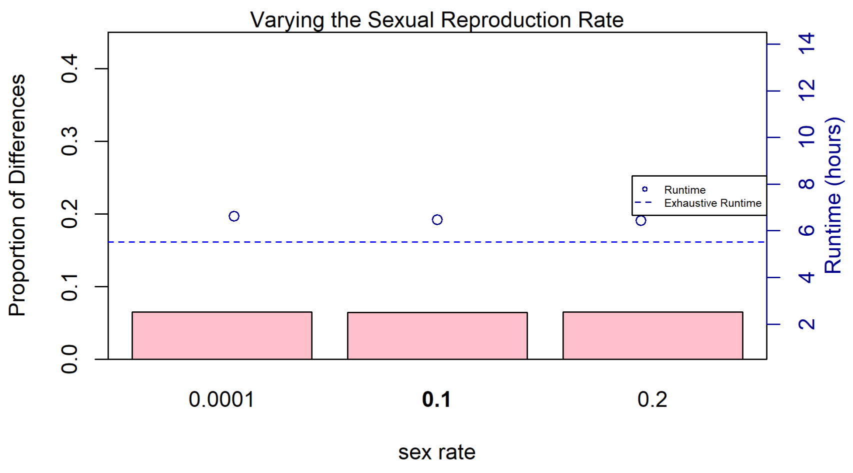


Figure 15: Proportion of differences (box plot corresponding the left vertical axis) and runtime (blue circles corresponding to right vertical axis) for the sexual reproduction rate considered. glmulti default of 0.1 is bolded.

Figure 15 shows that the proportion of differences and runtime is relatively constant across sexual reproduction rate values in the five covariate case. Therefore, we consider the sexual reproduction rate negligible. This is also seen in the four covariate case.

5.1.4 Mutation Rate

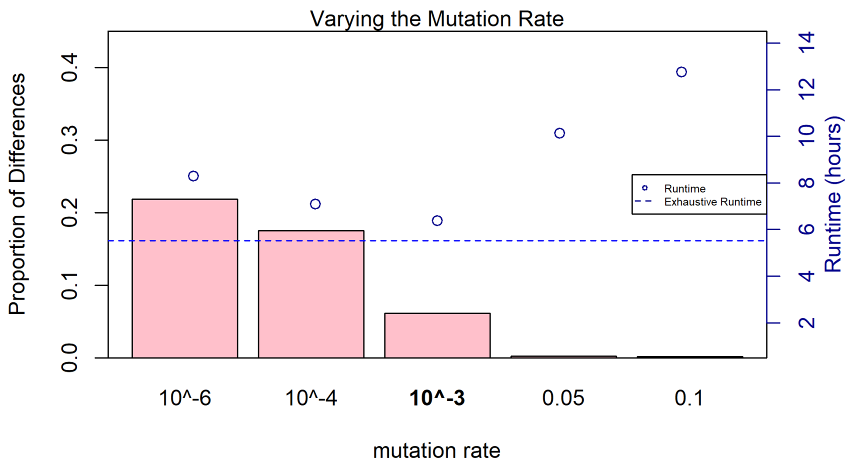


Figure 16: Proportion of differences (box plot corresponding the left vertical axis) and runtime (blue circles corresponding to right vertical axis) for the mutation rate considered. glmulti default of 10^{-3} is bolded.

As mutation rate increases, the proportion of differences decreases. The runtime experiences a minimum around the default vale of 10^{-3}. A good range to study further would be between 0.01 and 0.05.

5.1.5 Consecutive Value

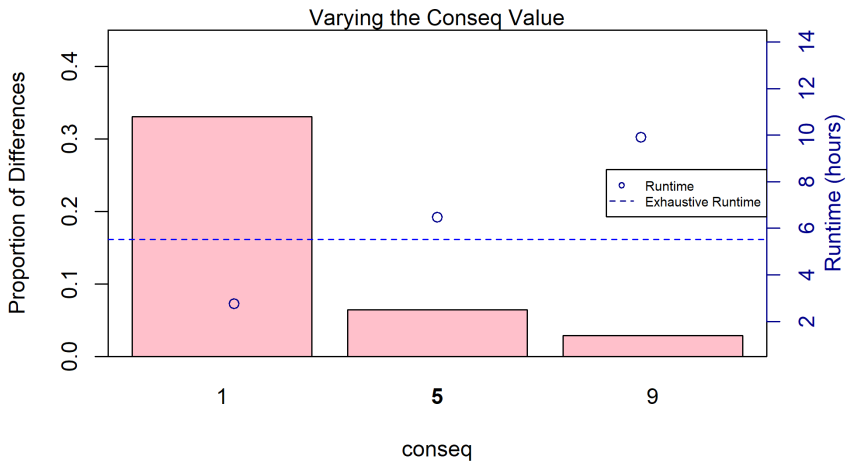


Figure 17: Proportion of differences (box plot corresponding the left vertical axis) and runtime (blue circles corresponding to right vertical axis) for the consecutive values considered. glmulti default of 5 is bolded.

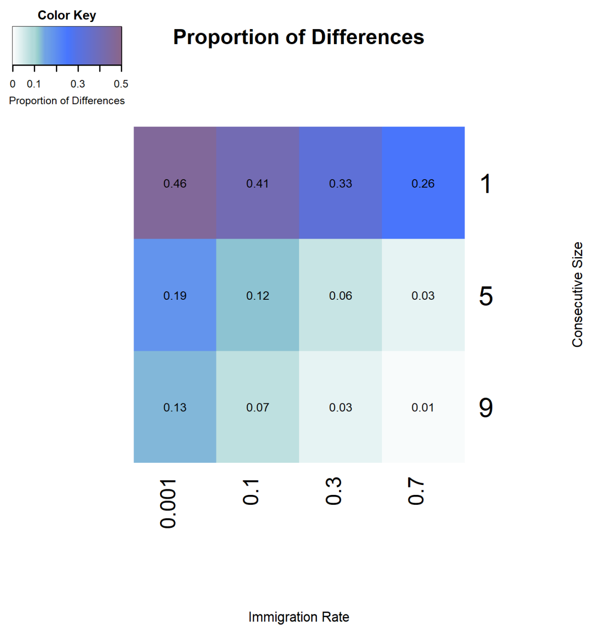
Just as in the four covariate case, there is an obvious trend in both proportion of differences and the runtime as the consecutive values increase. A higher consecutive value results in higher runtimes because the genetic algorithm will have to go through more iterations before converging. The default value is a suitable compromise.

5.2 Varying Two Parameters at a Time on a Personal Computer

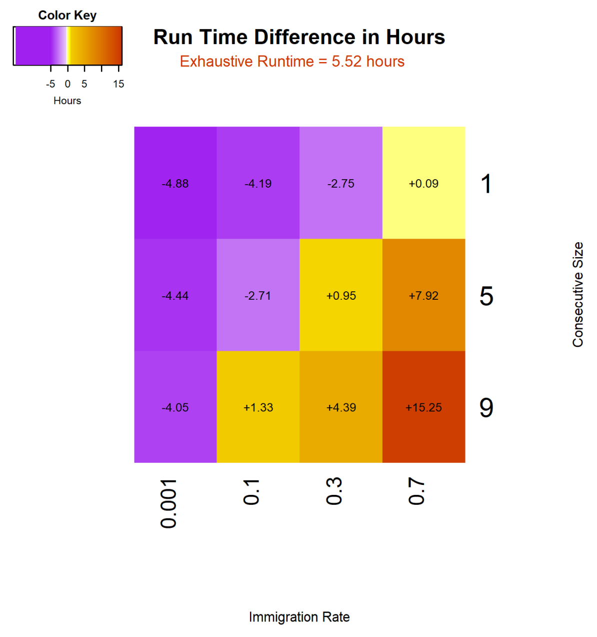
In the previous section, we varied each parameter individually (one degree varying) to determine the best range of parameter values to consider when varying two parameters at a time (pairwise interactions). In this section, we discuss the results of these pairwise interactions using the informed parameter ranges. This will reveal any potential interaction between the parameters. As previously seen, changes in the sexual reproduction rate produced negligible differences, so it will not be studied further.

5.2.1 Immigration Rates with Consecutive Values

We vary the immigration rate concurrently with the consecutive value to help us determine the optimal value for both parameters. The color scales in Figure 18 shows the different values for both the proportion of differences and the runtime difference. The runtime differences is defined as the difference between the genetic algorithm and the exhaustive search, with positive values indicating the genetic algorithm took longer to run than the exhaustive search. Each cell corresponds to the same cell in the other plot (Fig. 18a and 18b) to easily compare the two.



(a) *Proportion of Differences*



(b) *Runtime*

Figure 18: The default values for the genetic algorithm is an immigration value of 0.3 and a consecutive value of 5. The proportion of differences is the percentage of models that differ from the exhaustive models. The runtime difference is the runtime of the genetic algorithm subtracted from the runtime from the exhaustive method.

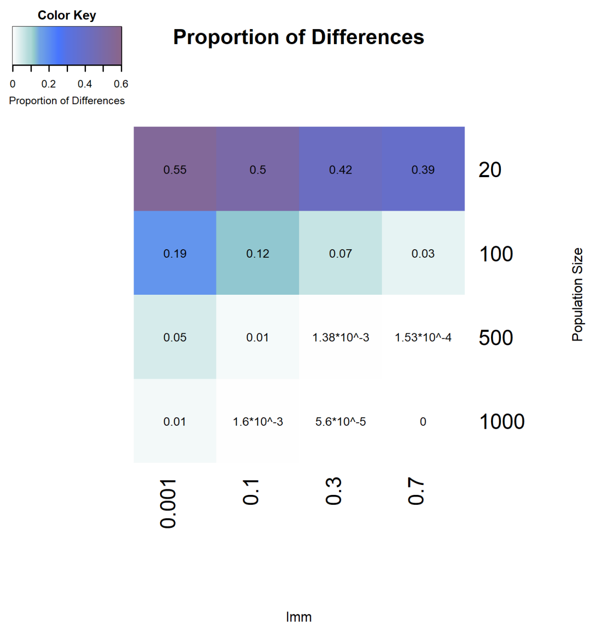
In this study using a personal computer, only the immigration rate and the consecutive value were changed. Every other parameter value was left at the default. As an example of how these heatmaps are useful, we can see that a consecutive value of 9 and a immigration value of 0.3 would be preferred over using a consecutive value of 5 and an immigration value of 0.7. This is because although the two sets of values result in the same proportion of differences, the runtime has a significant increase when using the latter values. The optimal region in this parameter space is between an immigration rate of 0.001 and 0.1 and a consecutive value of 9.

5.3 Varying Two Parameters at a time on an HPC System

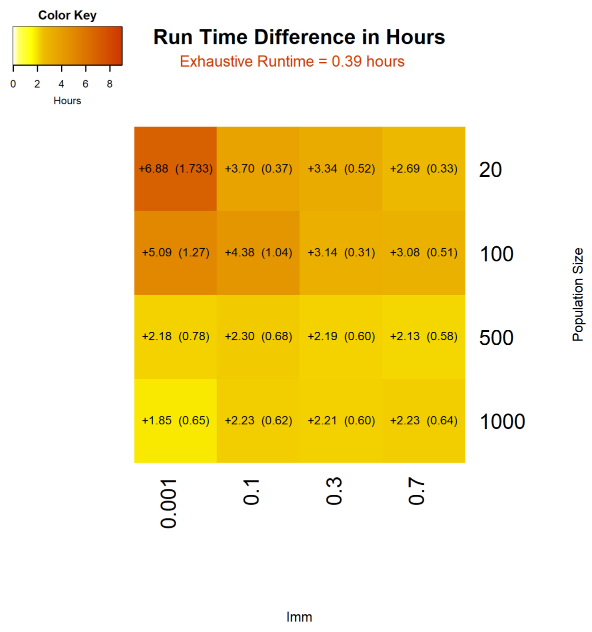
The remaining parameter combinations are run on the NCAR high-performance computing system (HPC) Cheyenne. Cheyenne has 145,152 processor cores, 4,032 computation nodes, and 313 TB of total system memory. HPC systems allow for faster computation, and therefore we again perform 5 trials for each parameter combination. It should be noted that the results from Figure 18 would not necessarily be the same if run on Cheyenne.

5.3.1 Population Size and Immigration Rate

First we vary population size and immigration rate values. Figure 19 shows heatmaps that display both proportion of differences and runtime differences. We will see that all the parameter combinations resulted in positive runtimes. This will be discussed more thoroughly in the Conclusion section.



(a) Proportion of Differences

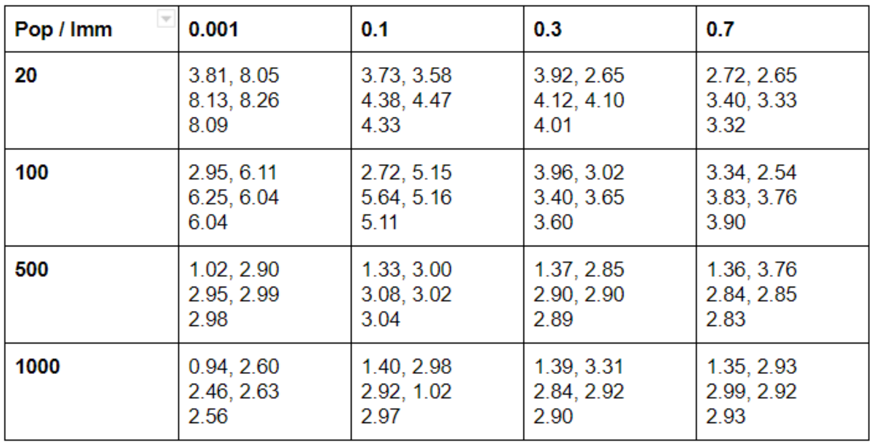


(b) Runtime

Figure 19: The default value for the genetic algorithm is an immigration value of 0.3 *and a population size of 100. The proportion of differences is shown on the left hand side and the difference in runtime from the exhaustive search is shown on the right hand side. Lighter colors correspond to better results.*

Here proportion of differences is defined as in previous sections. In the runtime heatmap, we plot the average runtime with the standard deviation in parenthesis. We do not present standard deviation for the proportion of differences, as there was very little variability (on the order of 10^{-3}). Due to the low number of trials and the high variability in runtimes, we list the individual runtimes in Table 2.

Table 2: Individual runtimes for 5 trials of each combination of population size and immigration rate parameter values. These are the raw runtimes of each trial, meaning that the exhaustive time was not subtracted from any of them. Population size corresponds to the y-axis and immigration rate corresponds to the x-axis.



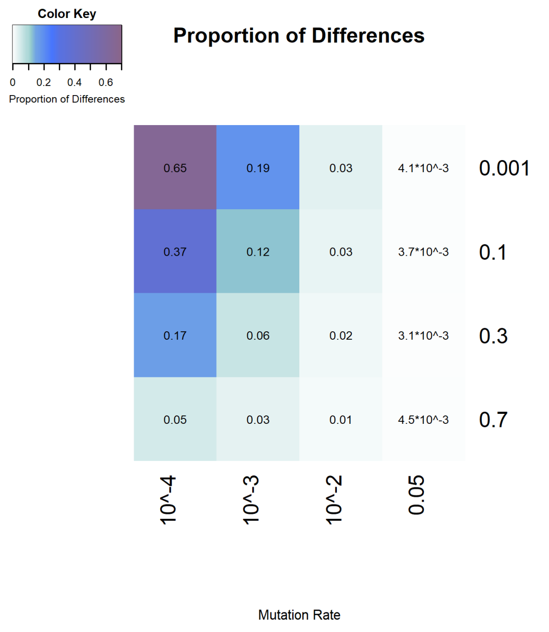
Pop / Imm	**0.001**	**0.1**	**0.3**	**0.7**
20	3.81, 8.05 8.13, 8.26 8.09	3.73, 3.58 4.38, 4.47 4.33	3.92, 2.65 4.12, 4.10 4.01	2.72, 2.65 3.40, 3.33 3.32
100	2.95, 6.11 6.25, 6.04 6.04	2.72, 5.15 5.64, 5.16 5.11	3.96, 3.02 3.40, 3.65 3.60	3.34, 2.54 3.83, 3.76 3.90
500	1.02, 2.90 2.95, 2.99 2.98	1.33, 3.00 3.08, 3.02 3.04	1.37, 2.85 2.90, 2.90 2.89	1.36, 3.76 2.84, 2.85 2.83
1000	0.94, 2.60 2.46, 2.63 2.56	1.40, 2.98 2.92, 1.02 2.97	1.39, 3.31 2.84, 2.92 2.90	1.35, 2.93 2.99, 2.92 2.93

For population size and immigration rate we can see that a higher population size and higher immigration rate reduces the proportion of differences. With the goal of minimizing the runtime and keeping a small proportion of differences, an optimal combination is a population size of

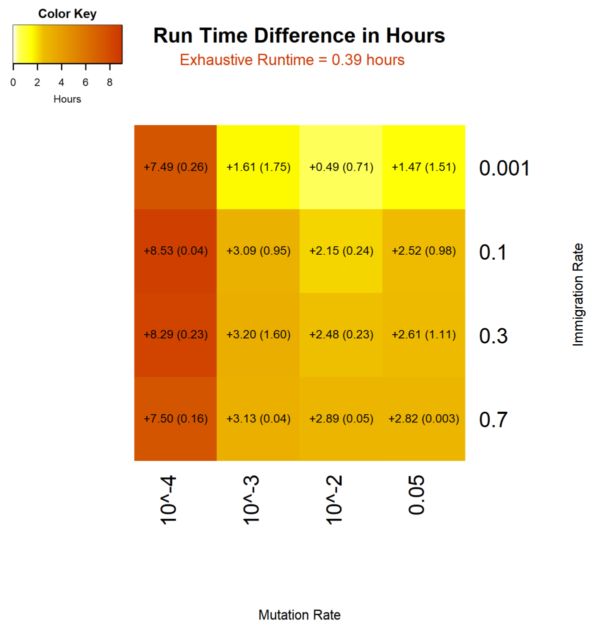
1000 and an immigration rate of 0.001. Even with this optimal parameter combination, it is still significantly slower than the exhaustive search. There also is low variation in the runtime between different trials for this parameter combination.

5.3.2 Immigration Rate and Mutation Rate

Figure 20 shows proportion of differences and runtime heatmaps for immigration rate and mutation rate values. The individual runtimes for these parameter combinations are given in Table 3.



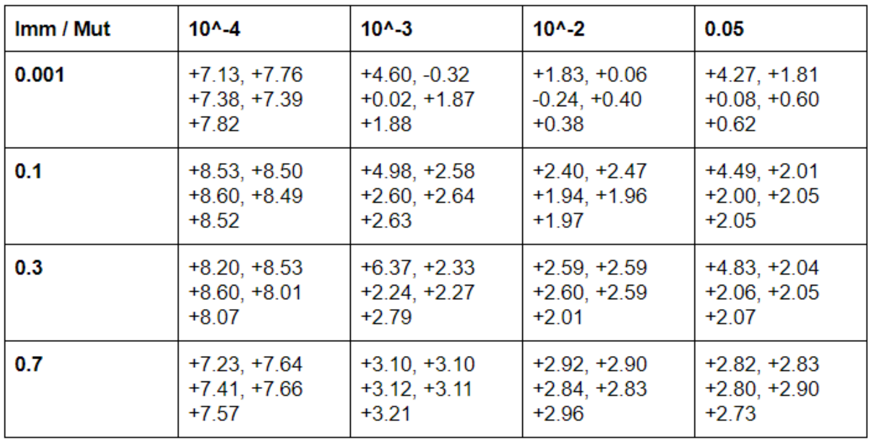
(a) Proportion of Differences



(b) Runtime

Figure 20: The default values for the genetic algorithm is an immigration value of 0.3 and a mutation rate of 10^{-3}. The proportion of differences is shown on the left hand side and the difference in runtime from the exhaustive search is shown on the right hand side. Lighter colors correspond to better results.

Table 3: Individual runtimes for 5 trials of each combination of immigration and mutation rate parameter values. These are the raw runtimes of each trial, meaning that the exhaustive time was not subtracted from any of them. Immigration rate corresponds to the y-axis and mutation rate corresponds to the x-axis.

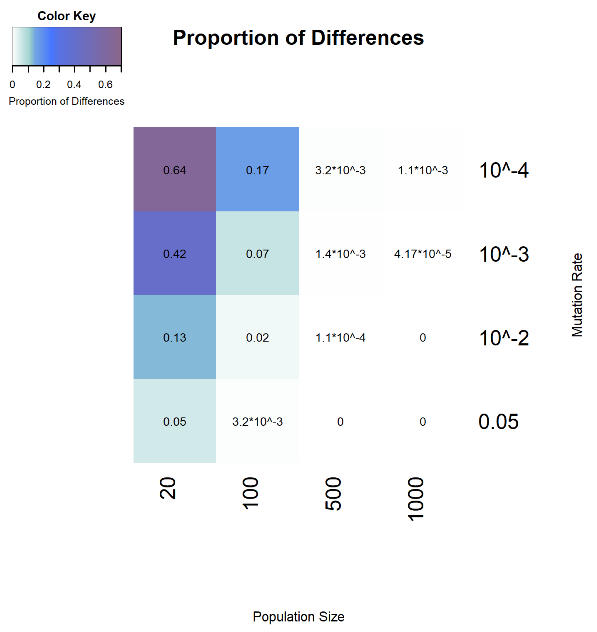


Imm / Mut	10^-4	10^-3	10^-2	0.05
0.001	+7.13, +7.76 +7.38, +7.39 +7.82	+4.60, -0.32 +0.02, +1.87 +1.88	+1.83, +0.06 -0.24, +0.40 +0.38	+4.27, +1.81 +0.08, +0.60 +0.62
0.1	+8.53, +8.50 +8.60, +8.49 +8.52	+4.98, +2.58 +2.60, +2.64 +2.63	+2.40, +2.47 +1.94, +1.96 +1.97	+4.49, +2.01 +2.00, +2.05 +2.05
0.3	+8.20, +8.53 +8.60, +8.01 +8.07	+6.37, +2.33 +2.24, +2.27 +2.79	+2.59, +2.59 +2.60, +2.59 +2.01	+4.83, +2.04 +2.06, +2.05 +2.07
0.7	+7.23, +7.64 +7.41, +7.66 +7.57	+3.10, +3.10 +3.12, +3.11 +3.21	+2.92, +2.90 +2.84, +2.83 +2.96	+2.82, +2.83 +2.80, +2.90 +2.73

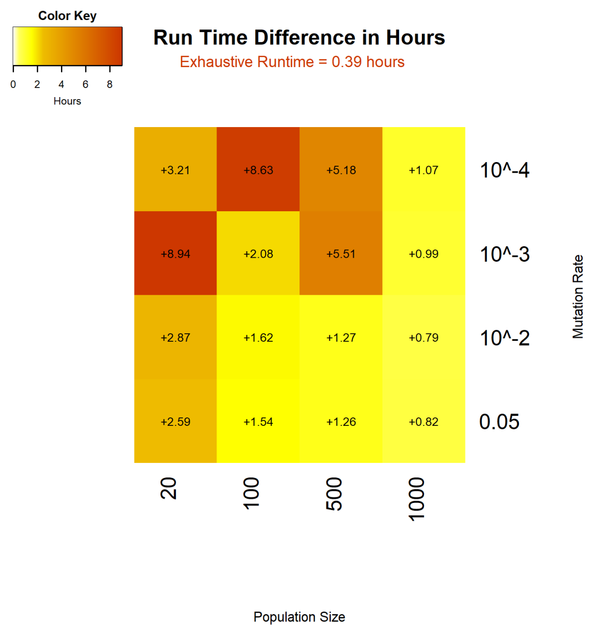
A mutation rate of 0.05 results in very low proportion of differences when combined with any immigration rate value. The lowest runtime parameter combination is an immigration rate of 0.001 and a mutation rate of 10^{-2}, and thus we chose these as the optimized values.

5.3.3 Population Size and Mutation Rate

Figure 21 shows the same heatmaps for population size and mutation rate. The individual runtimes are given in Table 4.



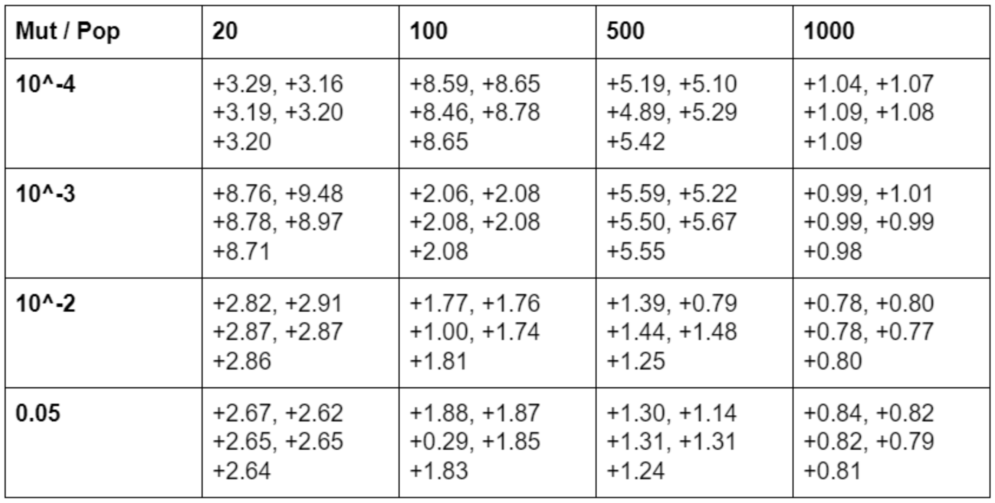
(a) *Proportion of Differences*



(b) *Runtime*

Figure 21: The default values for the genetic algorithm is a mutation rate of 10^{-3} and a population size of 100. The proportion of differences is shown on the left hand side and the difference in runtime from the exhaustive search is shown on the right hand side. Lighter colors correspond to better results.

Table 4: Individual runtimes for 5 trials of each combination of mutation rate and population size parameter values. These are the raw runtimes of each trial, meaning that the exhaustive time was not subtracted from any of them. Mutation rate corresponds to the y-axis and population size corresponds to the x-axis.

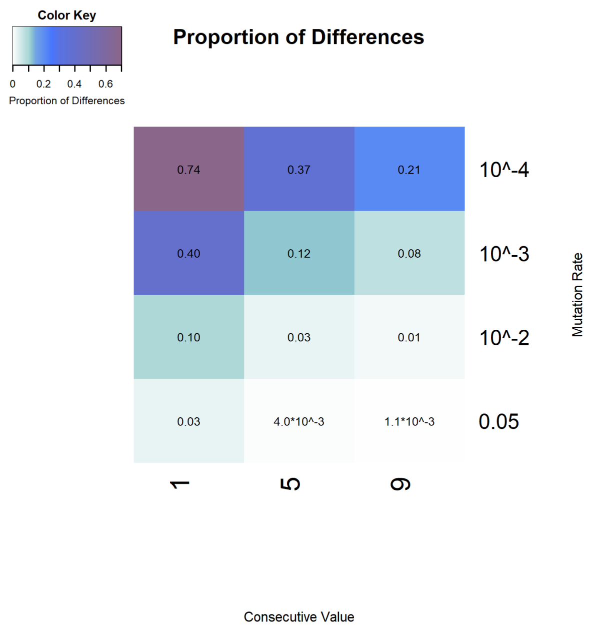


Mut / Pop	**20**	**100**	**500**	**1000**
10^-4	+3.29, +3.16 +3.19, +3.20 +3.20	+8.59, +8.65 +8.46, +8.78 +8.65	+5.19, +5.10 +4.89, +5.29 +5.42	+1.04, +1.07 +1.09, +1.08 +1.09
10^-3	+8.76, +9.48 +8.78, +8.97 +8.71	+2.06, +2.08 +2.08, +2.08 +2.08	+5.59, +5.22 +5.50, +5.67 +5.55	+0.99, +1.01 +0.99, +0.99 +0.98
10^-2	+2.82, +2.91 +2.87, +2.87 +2.86	+1.77, +1.76 +1.00, +1.74 +1.81	+1.39, +0.79 +1.44, +1.48 +1.25	+0.78, +0.80 +0.78, +0.77 +0.80
0.05	+2.67, +2.62 +2.65, +2.65 +2.64	+1.88, +1.87 +0.29, +1.85 +1.83	+1.30, +1.14 +1.31, +1.31 +1.24	+0.84, +0.82 +0.82, +0.79 +0.81

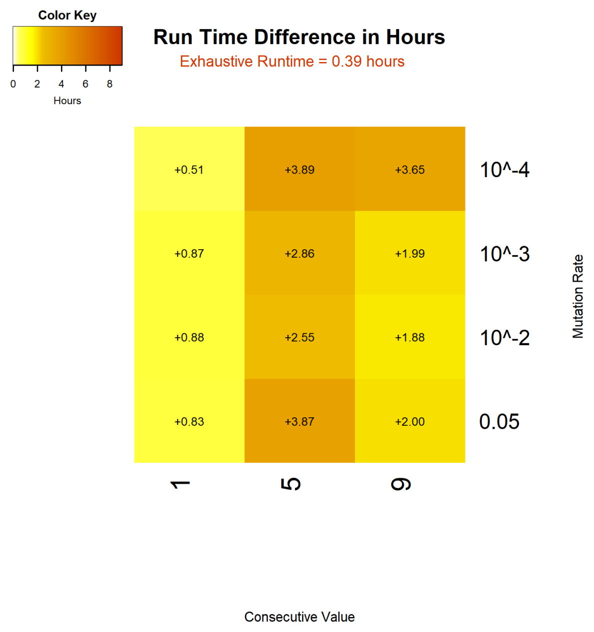
With a population size of 1000 there is both low runtime and proportion of differences, regardless of the mutation rate value. A population size of 1000 and a mutation rate of 10^{-2} results in an optimal runtime and proportion of differences.

5.3.4 Consecutive Values and Mutation Rate

Figure 22 shows the same heatmaps for consecutive values and mutation rate. The individual runtimes are given in Table 5.



(a) *Proportion of Differences*



(b) *Runtime*

Figure 22: The default value for the genetic algorithm is a consecutive value of 5 and a mutation rate of 10^{-3}. *The proportion of differences is shown on the left hand side and the difference in runtime from the exhaustive search is shown on the right hand side. Lighter colors correspond to better results.*

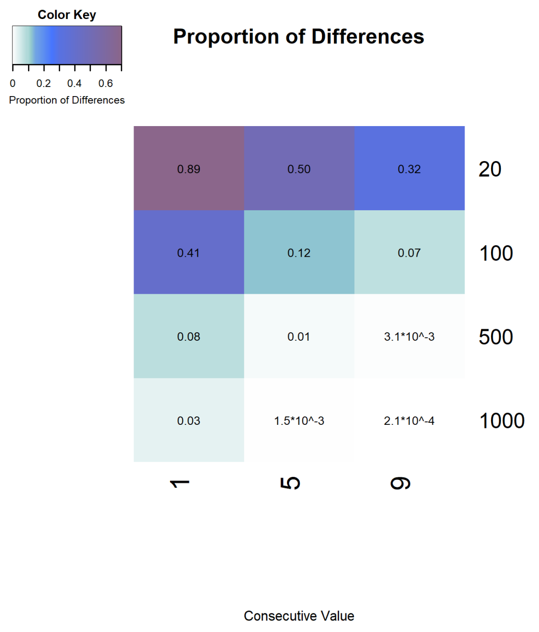
Table 5: Individual runtimes for 5 trials of each combination of mutation rate and consecutive parameter values. These are the raw runtimes of each trial, meaning that the exhaustive time was not subtracted from any of them. Mutation rate corresponds to the y-axis and consecutive values correspond to the x-axis.

Mut / Conseq	**1**	**5**	**9**
10^-4	+0.72, +0.71 +0.70, +0.71 -0.29	+3.96, +3.85, +3.85, +3.95 +3.85	+4.70, +4.65 +2.96, +2.96 +2.98
10^-3	+0.83, +0.85 +0.81, +0.82 +1.07	+2.86, +2.84 +2.81, +2.96 +2.82	+1.23, +1.22 +2.51, +2.46 +2.53
10^-2	+0.81, +0.85 +0.82, +0.80 +1.11	+2.53, +2.54 +2.49, +2.62 +2.55	+1.18, +1.18 +2.39, +2.30 +2.37
0.05	+0.75, +0.74 +0.77, +0.79 +1.11	+3.90, +3.85 +3.86, +3.89 +3.87	+1.32, +1.32 +2.45, +2.47 +2.44

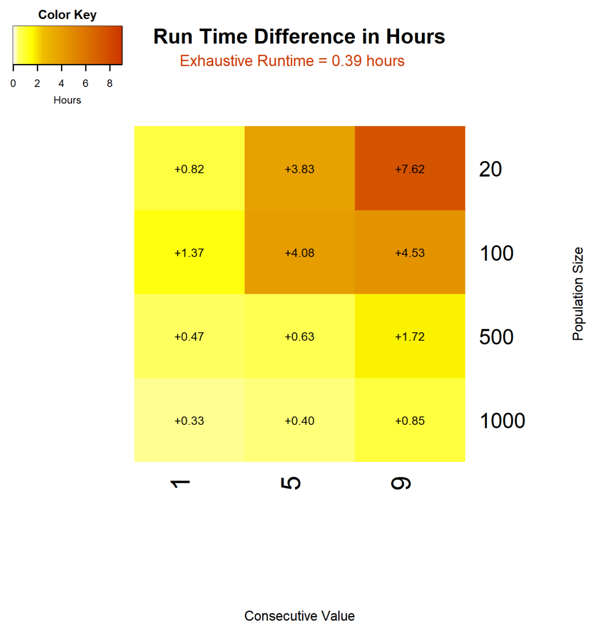
With a consecutive value of 1, the runtime decreases significantly compared to using a value of 5 or 9. As before we can see a higher mutation rate is associated with better proportion of differences. We select a consecutive value of 1 and a mutation rate of 10^{-2} as the optimal values.

5.3.5 Population Size and Consecutive Size

The sixth and final parameter combination that we tested was population size and consecutive value. Both heatmaps and a table of the different runtimes (Fig. 23 and Table 6) were produced for this parameter combination.



(a) Proportion of Differences



(b) Runtime

Figure 23: The default value for the genetic algorithm is a consecutive value of 5 and a population size of 100. The proportion of differences is shown on the left hand side and the difference in runtime from the exhaustive search is shown on the right hand side. Lighter colors correspond to better results.

Table 6: Individual runtimes for 5 trials of each combination of population size and consecutive parameter values. These are the raw runtimes of each trial, meaning that the exhaustive time was not subtracted from any of them. Population size corresponds to the y-axis and consecutive values correspond to the x-axis.

Pop / Conseq	**1**	**5**	**9**
20	-0.37, +0.96 +1.16, +1.18 +1.16	-0.33, +4.79 +4.96, +4.86 +4.86	+4.19, +8.37 +8.40, +8.52 +8.63
100	+1.32, +1.31 +1.42, +1.41 +1.38	+3.89, +3.81 +4.31, +4.20 +4.18	+5.49, +5.40 +3.83, +3.94 +3.96
500	+0.47, +0.47 +0.46, +0.47 +0.50	+0.64, +0.66 +0.61, +0.61 +0.65	+1.71, +1.72 +1.73, +1.73 +1.69
1000	+0.37, +0.45 -0.07, +0.47 +0.41	+0.52, +0.52 +0.0006, +0.51 +0.45	+1.09, +1.02 +0.05, +1.03 +1.05

There are low runtimes and low proportion of differences for any consecutive value when using a population size of 1000. With a population size of 1000, the proportion of differences does not increase much when using a consecutive value of 1 compared to a consecutive value of 5. Thus, a good parameter combination would be a population size of 1000 and a consecutive value of 1.

5.3.6 Discussion of Results

Based on the pairwise studies above, we have selected the following parameter values as optimal for the CO modeling application with five covariates.

Parameter	**Default**	**5 covariate values**	**4 covariate values**
Population Size	100	1000	40
Immigration Rate	0.3	0.001	0.001
Sexual Reproduction Rate	0.1	no significant influence-default selected	0.1
Mutation Rate	0.001	0.01	0.2
Consecutive Iterations	5	5	2

Table 7: The defualt for each parameter and optimized values for both the 5 covariate and 4 covariate case.

Table 7 shows the optimized values that resulted from the study on Cheyenne (HPC system at NCAR). It appears that on Cheyenne, the exhaustive method runs much faster than on a personal computer relative to the genetic algorithm. Note that the results in Table 7 are optimized on an HPC system and might be different for a personal computer.

We directly compare the optimized *glmulti* parameters to the default values with five covariate models Figure 24.

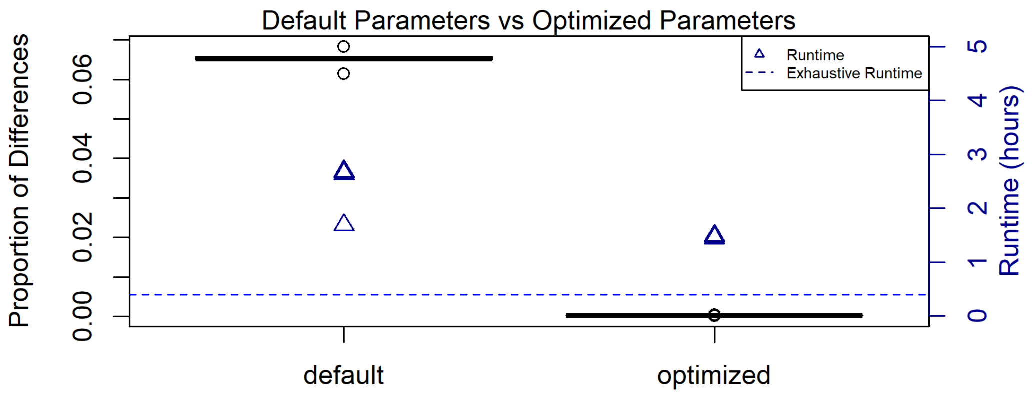


Figure 24: Comparison of default glmulti parameters to optimized parameters.

The optimized parameters decrease both runtime and proportion of differences. The runtime decreased by an average of 1 hour and 3.6 seconds, and the proportion of differences decreased by an average of 0.4%. It it interesting to note that the optimized genetic algorithm is still slower than the exhaustive search in an HPC environment. Again, these results might differ on a personal computer.

6 Conclusion

In the four covariate case, we successfully optimize the genetic algorithm so its runtime is lower than the exhaustive method. Notably, the optimized population size (40) is much smaller than the total number of possible models (113). The four covariate study is performed on a personal computer. Therefore, the genetic algorithm with optimized parameters can serve as an intermediate variable selection method between stepwise selection and the exhaustive search with four covariate models. This is useful for researchers who do not have access to computing resources, as the genetic algorithm runs faster than the exhaustive search without sacrificing significant model accuracy.

We perform our study for the five covariate case on the NCAR HPC system Cheyenne. As in the four covariate case, the optimized genetic algorithm is faster and has a lower proportion of differences than the default settings. That being said, the exhaustive method is faster than the optimized genetic algorithm in this scenario. In the five covariate case, the optimized population size (1000) was much closer to the total number of models (1450), which might indicate that the genetic algorithm is trending towards the exhaustive search. As the population size increases, the genetic algorithm becomes more similar to an exhaustive search. The exhaustive search seems to outperform the genetic algorithm in an HPC environment, so it makes sense that the optimized genetic algorithm parameters push the algorithm towards an exhaustive ‘search. While the runtime of the optimized genetic algorithm does not fall between the stepwise and exhaustive methods, these findings are still interesting and useful. Our results show that when using an HPC system, an exhaustive search will likely outperform a stochastic variable selection technique in applications with around one to two thousand possible models.

7 Acknowledgements

The NCAR MOPITT project is supported by the National Aeronautics and Space Administration (NASA) Earth Observing System (EOS) Program. MOPITT is also supported by the Canadian Space Agency (CSA), the Natural Sciences and Engineering Research Council (NSERC) and Environment Canada, and the contributions of COMDEV (the prime contractor) and ABB BOMEM. We would like to acknowledge high-performance computing support from Cheyenne (doi:10.5065D6RX99HX) provided by NCAR's Computational and Information Systems Laboratory, sponsored by the National Science Foundation. We also thank NCAR internal reviewer Benjamin Gaubert.

8 References

[1] Tracey Holloway, Hiram Levy II, and Prasad Kasibhatla. Global distribution of carbon monoxide. *Journal of Geophysical Research: Atmospheres*, 105(D10):12123–12147, 2000.

[2] D. P. Edwards, L. K. Emmons, J. C. Gille, A. Chu, J.-L. Attié, L. Giglio, S. W. Wood, J. Haywood, M. N. Deeter, S. T. Massie, D. C. Ziskin, and J. R. Drummond. Satellite-observed pollution from southern hemisphere biomass burning. *Journal of Geophysical Research: Atmospheres*, 111(D14), 2006.

[3] R. R. Buchholz, D. Hammerling, H. M. Worden, M. N. Deeter, L. K. Emmons, D. P. Edwards, and S. A. Monks. Links between carbon monoxide and climate indices for the southern hemisphere and tropical fire regions. *Journal of Geophysical Research: Atmospheres*, 123, 2018.

[4] James R. Drummond, Jiansheng Zou, Florian Nichitiu, Jayanta Kar, Robert Deschambaut, and John Hackett. A review of 9-year performance and operation of the MOPITT instrument. *Advances in Space Research*, 45(6):760–774, 2010.

[5] M. N. Deeter, L. K. Emmons, G. L. Francis, D. P. Edwards, J. C. Gille, J. X. Warner, B. Khattatov, D. Ziskin, J.-F. Lamarque, S.-P. Ho, V. Yudin, J.-L. Attié, D. Packman, J. Chen, D. Mao, and James R. Drummond. Operational carbon monoxide retrieval algorithm and selected results for the mopitt instrument. *Journal of Geophysical Research: Atmospheres*, 108(D14), 2003.

[6] M. N. Deeter, D. P. Edwards, G. L. Francis, J. C. Gille, S. Martínez-Alonso, H. M. Worden, and C. Sweeney. A climate-scale satellite record for carbon monoxide: the mopitt version 7 product. *Atmospheric Measurement Techniques*, 10(7):2533–2555, 2017.

[7] William Daniels, Dorit Hammerling, and Rebecca Buchholz. regclimatechem: An r package for data driven variable selection applied to atmospheric carbon monoxide. Technical report, (No. NCAR/TN-562+STR). doi: `10.5065/e8xj-3k89`.

[8] V. Calcagno and C. de Mazancourt. glmulti: An R package for easy automated model selection with (generalized) linear models. *Journal of Statistical Software*, 34(12):29, 2010.

[9] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.