Regularization Parameter Optimization in NIPA using Simulated Annealing

TEUN HOVEN, University of Twente, The Netherlands

ABSTRACT

Understanding and predicting outbreaks of epidemics has become a major focus since COVID-19. Researchers have explored various methods, from basic curve fitting to complex machine learning techniques, to predict how the virus spreads. One promising method is the Network Inference-based Prediction Algorithm (NIPA), which uses the SIR-model and the least absolute shrinkage and selection operator to estimate how the infections spread over different regions. However, fine-tuning the regularization parameter of NIPA can be complicated because of the time-consuming process and sub-optimal result of k-fold Cross-Validation (CV). To overcome this, we suggest using Simulated Annealing (SA) to optimize NIPA's regularization parameter. Our study aims to combine SA with NIPA to make the process of choosing the optimal value for the parameter more effective. The results of the research show that the accuracy is improved and therefore indicate that SA is an acceptable alternative to CV, regardless of the computation time being higher. This research has found a method that can benefit epidemic modeling and prediction efforts.

Additional Key Words and Phrases: Hyper-parameter optimization, Regularization parameter optimization, Least Absolute Shrinkage and Selection Operator (LASSO), Network Inference-based Prediction Algorithm (NIPA), Simulated Annealing (SA)

1 INTRODUCTION

Since COVID-19 spread around the world, many research has been conducted in predicting the spread of the pandemic. Predicting how the spread of a virus will evolve is difficult, as it is comparable to weather forecasts and subject to fundamental limits [14]. Although it is challenging, it is not impossible. Many researchers have developed methods to try to predict the spread of COVID-19. This research ranges from simple approaches, such as fitting the number of infections to a sigmoid curve, to using statistical approaches, network-based approaches, machine learning algorithms, and parameter estimations in compartmental models such as the SIR-model [1]. In 2022, Achterberg et al. compared the precision of different network-based techniques to predict cases of COVID-19. The prediction algorithms that were compared in the paper were long short-term memory, Gompertz function, Hill function, logistic function and Network Inference-based Prediction Algorithm (NIPA). In the end, the NIPA proved to be the algorithm with the best accuracy. [1].

The well-performing NIPA algorithm uses the Susceptible, Infected and Recovered (SIR) model to predict the spread of COVID-19. The NIPA first infers a matrix that is used as the infection probability between regions when individuals come in contact. After the infection probability matrix is inferred, the NIPA will iterate over each time step and calculates the fraction of susceptible, infected and recovered individuals against the population. Every individual is put in one of the three compartments *Susceptible* (S), *Infected* (I) or *Recovered* (R), where the recovered individuals can not infect other susceptible individuals. For every city i at time k, we obtain the SIR *viral state* by $v_i(k) = (\mathcal{S}_i(k), \mathcal{I}_i(k), \mathcal{R}_i(k))^T$ [16], where $\mathcal{S}_i$ corresponds to the fraction of susceptible people in region i, $\mathcal{I}_i$ to the fraction of infectious and $\mathcal{R}_i$ to the fraction of recovered individuals.

The network that NIPA is estimating is the matrix B of infection probabilities from the SIR viral state observations $v_i(1), ..., v_i(n)$. The task of NIPA is to estimate each infection probability β_{ij} of the infected people of region j to susceptible people in region i. To infer the network of infections between regions, NIPA uses the Least Absolute Shrinkage and Selection Operator (LASSO). For inferring the network, we have to solve the LASSO for each row i to find the vector of weights, in NIPA's case the infection probabilities matrix B, that minimizes the quadratic error of the linear system:

$$\min_{\beta_i} \left\| y_i - X_i\beta_i \right\|_2^2 \text{ subject to } \sum_i |\beta_i| \leq c \tag{1}$$

where y_i are the responses, X_i are the predicted variables, β_i are the LASSO estimates, which the sum of these estimates are subjected to a constrained c [18]. This equation can be written in the orthonomal design as:

$$\min_{\beta_i} \left\| y_i - X_i\beta_i \right\|_2^2 + \rho \sum_i |\beta_i| \tag{2}$$

Where ρ is the regularization parameter and for every c of the constraint of Equation 1, there exists a corresponding value for ρ. Equation 2 can be divided into two parts, the Ordinary Least Square (OLS) estimate $\left\| y_i - X_i\beta_i \right\|_2^2$ and the penalty function $\rho \sum_i |\beta_i|$, for $\rho \geq 0$.

Due to this ℓ_1-norm penalty function, LASSO is able to shrink the OLS estimators (β_i) to zero. For this reason, LASSO can be regarded as a variable selection method with ρ as the shrinkage factor or the regularization parameter [12]. To find an optimal value for the regularization parameter, k-fold cross-validation or just Cross-Validation (CV) is often used. CV is a method to split the dataset into P amount of sets and for each set, divide it into a train and test set (often 80% and 20% respectively). The model is then trained on the train set and the evaluated on the test set. Although CV is the standard, the resulting value can result in unstable predictions and is computationally expensive to run [15].

In this paper, we propose using the Simulated Annealing (SA) [3] algorithm to find an optimal value for the regularization parameter without using the CV technique which has been used in the implementations of NIPA [1, 16]. LASSO in combination with SA does not

TScIT 41, July 5, 2024, Enschede, The Netherlands

have as much research as other regularization parameter optimization algorithms. However, in the research that has been done, the combination with SA has shown promising results against standard LASSO [20, 24]. For this reason, we are researching whether SA can be implemented for the *NIPA* as well. This results in the following research questions:

RQ1. *How can simulated annealing be incorporated into the network inference-based prediction algorithm to effectively search for an optimal value for the regularization parameter?*

RQ2. *How does NIPA with simulated annealing optimization, implemented in* ***RQ1****, compare to NIPA with k-fold cross-validation in prediction ability and computation time?*

We aim to contribute to the scientific community in three ways. First, we will create a framework to use SA with LASSO. Secondly, we will have implemented an algorithm to more accurately choose an optimal value for the regularization parameter. At last, due to the second contribution, there will be a more accurate prediction of COVID-19 cases with the NIPA.

The structure of the research paper will be as follows: In Section 2 we will discuss previous and related research on the use of SA for hyper-parameter optimization and in combination with LASSO. Section 3 will first describe the algorithms and definitions used in the research and secondly, we will describe the approach taken to get the results. The results of the research, will be presented in Section 4. In Section 5, we discuss the limitations of the research and what would be interesting to take into consideration for further research. At last, in Section 6 the conclusion of the research is given.

2 RELATED WORK

In this section we go over previous and related work of other researchers.

Finding optimal values for hyper-parameters by using SA has been researched for some time. In 2007, Lin et al. [13] proposed a SA method to determine parameters in a support vector machine. This method aims to search for the optimal value of the parameters to maximize the accuracy rate. This approach resulted in a higher classification accuracy rate than performing grid search when finding values for the parameters.

A proposed method called Evolutionary Simulating Annealing LASSO Logistic Regression (ESALOR) was proposed by Tutun et al. in 2016 [20]. This method uses a hybrid meta-heuristic optimization approach, where they prevent over-fitting of the coefficients of logistic regression by using regularization (LASSO) and to optimize these coefficients they use the evolutionary strategy of SA. It shows promising results, being more accurate in classifying readmission of diabetic patients than the other methods (support vector machines, artificial neural networks, naive Bayes algorithm and logistic regression). The researchers did not compare their method with other LASSO techniques, meaning that there is no proof if ESALOR is as accurate as LASSO without SA.

Another paper from 2016, from Zhang et al. [24] proposed a method in which they describe the dynamic shrinking of LASSO resembling an annealing process. Due to this relation, their method integrates a similar type of regularization optimization by using adaptive weights and this results in their solution path being different than the traditional LASSO solution path.

An interesting research about hyper-parameter optimization came from Bertrand et al. in 2020 [2]. It discusses how difficult setting the regularization parameter of LASSO-type estimators is. The paper then dives into the differentiation of the LASSO which allows the researchers to select the hyper-parameter through standard gradient-descent. This method can also scale to a high number of hyper-parameters.

With regards to approaching SA, Guilmeau et al. [7] reviewed different types of SA algorithms and also proposed a new technique which combines two previous SA methods, Fast Simulated Annealing (FSA) [17] and sequential Monte Carlo Simulated Annealing (SMC-SA) [25]. This combination should allow better state space exploration from FSA while remaining the meaningful exchange between particles from the SMC-SA.

3 DEFINITIONS AND METHODS

In this section, the definitions, algorithms and the approach of the research will be explained and discussed.

3.1 Algorithms & Definitions

3.1.1 NIPA. The NIPA is based on the SIR-model, where every individual is in one of three groups: *susceptible*, *infected* or *recovered*. The first group consists of individuals who have not yet been infected and will go from *susceptible* to *infected* when they come in contact with infectious individuals from the second group [16]. The third group consists of people that can not infect others, therefore it is often called *removed*. For every city i at any discrete time k we denote the 3×1 viral state by:

$$v_i(k) = \begin{pmatrix} \mathcal{S}_i(k) \\ \mathcal{I}_i(k) \\ \mathcal{R}_i(k) \end{pmatrix} \tag{3}$$

In the equation above, it holds that $\mathcal{S}_i(k) + \mathcal{I}_i(k) + \mathcal{R}_i(k) = 1$ due to the components of the equation corresponding to the fraction of susceptible, infected and recovered people, respectively. To predict the amount of infected individuals, the *viral state* is updated each time step k according to:

$$\mathcal{I}_i(k+1) = (1-\delta_i)\mathcal{I}_i(k) + (1 - \mathcal{I}_i(k) - \mathcal{R}_i(k)) \sum_{j=1}^{N} \beta_{ij} \mathcal{I}_j(k) \tag{4}$$

$$\mathcal{R}_i(k+1) = \mathcal{R}_i(k) + \delta_i \mathcal{I}_i(k) \tag{5}$$

$$\mathcal{S}_i(k+1) = 1 - \mathcal{I}_i(k) - \mathcal{R}_i(k) \tag{6}$$

In Equation 4, β_{ij} denotes the infection probability between regions i and j. The δ_i in Equations 4 and 5 denotes the curing probability of region i. These two probabilities are unknown and are based on the viral state $v_i(1), ...v_i(n)$, the estimations $\hat{\delta_i}$ and $\hat{\beta_{ij}}$ can be found by using CV and LASSO [16]. The infection probability β_{ij} specifies the probability of getting infected when infected individuals in region j come in contact with susceptible individuals in region i, where the infection probability matrix between regions is given by an $N \times N$ matrix

$$B = \begin{pmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N1} & \beta_{N2} & \dots & \beta_{NN} \end{pmatrix} \tag{7}$$

where each element represents a probability $0 \leq \beta_{ij} \leq 1$. This matrix is estimated by the LASSO [18]. The network inference approach is suitable for the compartmental epidemic models like the SIR-model. In the equations of the SIR-model (4, 5, 6), it appears that β_{ij} is linear, but $\mathcal{S}_i$, $\mathcal{I}_i$ and $\mathcal{R}_i$ do not. From these equation, the infection probabilities β_{ij} satisfy

$$V_i = F_i \begin{pmatrix} \beta_{i1} \\ \vdots \\ \beta_{iN} \end{pmatrix} \tag{8}$$

for all cities $i = 1, ..., N$. The matrix V_i $(n-1 \times 1)$ and F_i $(n-1 \times N)$ are given by

$$V_i = \begin{pmatrix} \mathcal{I}_i(2) - (1-\delta_i)\mathcal{I}_i(1) \\ \vdots \\ \mathcal{I}_i(n) - (1-\delta_i)\mathcal{I}_{n-1}(1) \end{pmatrix} \tag{9}$$

and

$$F_i = \begin{pmatrix} \mathcal{S}_i(1)\mathcal{I}_1(1) & \dots & \mathcal{S}_i(1)\mathcal{I}_N(1) \\ \vdots & \ddots & \vdots \\ \mathcal{S}_i(n-1)\mathcal{I}_1(n-1) & \dots & \mathcal{S}_i(n-1)\mathcal{I}_N(n-1) \end{pmatrix} \tag{10}$$

To infer the network based on the equations above, we use the LASSO (Equation 2)

$$\min_{\beta_{i1},\dots,\beta_{iN}} \left\| V_i - F_i \begin{pmatrix} \beta_{i1} \\ \vdots \\ \beta_{iN} \end{pmatrix} \right\|_2^2 + \rho_i \sum_{j=1, j\neq i}^{N} \beta_{ij} \tag{11}$$
$$\text{s.t. } 0 \leq \beta_{ij} \leq 1, j = 1, ..., N$$

where for each region i there exists an optimal regularization parameter ρ_i. With the LASSO, we can estimate the infection probabilities by using regression. The LASSO works by minimising the OLS with an ℓ_1-norm constraint given by the $\rho_i \sum_{j=1,j\neq i}^{N} \beta_{ij}$ part of Equation 11. The regularization parameter ρ_i determines how many values in the resulting $N \times N$ matrix go to zero.

3.1.2 *Cross-Validation.*

In previous research of the NIPA, the regularization parameter was selected by CV [16, 1]. The researchers have probably chosen this technique because CV is the most common technique for deciding the value of the regularization parameter [12]. The procedure of CV starts by defining a set of possible choices for the regularization parameter. For NIPA, a set Θ_i with 100 logarithmically equidistant values from a range based of V_i and F_i [16, 1]. The second step is dividing the data into P equal-sized parts (non-overlapping) with approximate s individuals in each part. Each part is then taken as the validation data denoted by x_v and $y_v (v \in \{1, ..., P\})$, the remaining parts $P-1$ we denote as training data x_t and y_t. Per value in Θ_i, we train the model on the training data and then try to predict y_v as $\hat{y_v}(\rho_i)$ with the validation data. The predictive ability is then evaluated and the optimal regularization parameter $\rho_{i,opt}$, which minimizes a certain metric of the prediction accuracy, is chosen to be the regularization parameter [12].

3.1.3 *Simulated Annealing.*

The SA algorithm is inspired by the physical process of metallurgy and uses terminology coming from the fields of physics. Atoms in metal experience large disorded movements when the metal is heated. When the metal is cooled down steadily, the movement weakens and the atoms stabilize around a certain position where the energy is minimal. This process of slow cooling the metal is called *annealing* [6]. To understand this intuitively, and therefore also the algorithm: at high temperature, the search (atoms) go through large random movements, resulting in exploration of a wide range of possible configurations, even with high energy. Due to these large random movements, high-energy configurations can be reached even though it might not be the preferred position. When the temperature lessens, the amount of movements also reduces, meaning that the search will prefer low-energy configurations and finally "freezes" into a low-energy minimum which can be a global minimum, although this is not guaranteed. Kirkpatrick et al. came up with the idea in 1983 to use this physics process to search for the global optimum and since then, SA, also called Classic Simulated Annealing (CSA), has been used in many optimization problems [9].

The CSA algorithm is surprisingly simple to implement. The pseudocode of the SA algorithm can be found in Algorithm 1. The algorithm utilizes an objective function of an optimization problem. For each iteration, the algorithm chooses a new location based on the visiting distribution, assumed to be Gaussian (local search) and computes the energy E at that location. If the energy is lower compared to the previous location, the move is always accepted. When the move results in a higher energy, a probability will be calculated that determines if the move will be accepted. This probability is given as follows:

$$P = 1 - e^{\frac{\Delta E}{T_k}} \tag{12}$$

where P is the acceptance probability, ΔE is the difference of energy between the new and the old location and T_k is the temperature at step k [4]. When T is high, it is more likely that a move to a location with a higher energy is accepted. Through the iterations the temperature T will decrease and eventually reach zero. The formula for the decreasing temperature can be linearly, although there has been much research in CSA with non-linear cooling schedules [4]. SA is a widely used global optimization method and much research has been done on SA which led to many versions of the algorithm.

The first modified SA was four years after its introduction in 1983 [17]. The proposal of Szu uses a Cauchy-Lorentz distribution (*semi-local search*) as the visiting distribution that moves the location often

Algorithm 1: Simulated Annealing

```
1 Initialization of $x_0$
2 for $k = 1, \ldots$ do
3     Generate a candidate $y_k \sim G(x_k, dy)$
4     Compute the acceptance probability $p_k = e^{\frac{\Delta E}{T_k}}$
5     Set $x_{k+1} = \begin{cases} y_k \text{ with probability } p_k \\ x_k \text{ with probability } 1 - p_k \end{cases}$
6 end for
```

in local search space but can occasionally jump to a point further away. With this addition, Szu and Hartly showed that the cooling schedule could be much faster and therefore this algorithm was called Fast Simulated Annealing (FSA). To decrease the temperature at a faster rate, FSA generalizes the accept-reject rule (see Equation 12) to any *acceptance function q* [7]:

$$P_k = q(p_k), \text{ where } pk := (\frac{\Delta E}{T_k}) \tag{13}$$

with this generalization, SA is allowed to decrease more slowly and the convergence will happen with a faster cooling schedule.

In 1996, Tsallis and Strariolo published their research on generalizing each part of the SA algorithm [19]. With the equations in their research, it was possible to obtain the CSA and FSA as well as a faster convergence with certain values for the Generalized Simulated Annealing (GSA) [19]. The GSA uses two artificial temperatures instead of the one temperature in CSA and FSA, where the shape of the distorted Cauchy-Lorentz distribution is controlled by q_v and the acceptance probability is controlled by q_a. The visiting distribution is as follows:

$$g_{q_v}(\Delta x(t)) \propto \frac{(T_{q_v}(t))^{-D/(3-q_v)}}{[1 + (q_v - 1)(\Delta x(t))^2/(T_{q_v}(t))^{2/(3-q_v)}]^{1/(q_v-1)+(D-1)/2}} \tag{14}$$

where D is the dimension of the variable space and T_{q_v} is the visiting temperature. The parameter q_v also controls the rate of cooling:

$$T_{q_v}(t) = T_{q_v}(1)\frac{2^{q_v-1} - 1}{(1+t)^{q_v-1} - 1} \tag{15}$$

q_a controls the acceptance probability which is a generalized Metropolis algorithm [22]:

$$P_{q_a} = \min\{1, (1 - (1 - q_a)\beta\Delta E)^{1/1-q_a}\} \tag{16}$$

where $\beta = \frac{1}{KT_{q_a}}$. To get CSA and FSA from this generalized form, we set $q_v = 1$ and $q_a = 1$ to get CSA and to get FSA, we set $q_v = 2$ and $q_a = 1$.

3.1.4 *Evaluation.*

To evaluate the performance of the prediction ability of NIPA with CV and NIPA with SA, we must define the evaluation metric. Measurement of performance for forecasting is not an easy task; researchers and forecasters have not yet agreed on a measurement technique that should be preferred [10]. Each method of evaluation has its own strengths and weaknesses, making the measurement technique different for each problem. Much research has been conducted on the accuracy of the prediction, as it is the most important criterion to select a prediction technique. Most forecast methods use point forecast which is why researchers are focused on identifying measures to show the accuracy of point forecasts [8].

Popular metrics for forecast evaluation is the mean square error (MSE), the mean absolute percentage error (MAPE) and the symmetric mean absolute percentage error (sMAPE). Where MSE penalizes large errors but is sensitive to outliers and can not handle multiple time series, MAPE handles these drawbacks but creates another. If the true values of the forecast go to zero, we get a large number and it is possible for MAPE to become undefined [10]. To counteract this, the sMAPE can be used. sMAPE is also commonly used by forecast researchers to quantify the predictions [8, 1]. After some discussion and consideration, we have chosen to evaluate our results according to the mean square error which is defined as:

$$e_{\text{MSE}}(t) = \frac{1}{N}\sum_{i=1}^{N}(Y_i(k) - \hat{Y}_i)^2 \tag{17}$$

where Y_i is the true data and $\hat{Y}_i$ is the predicted data.

3.2 Approach

3.2.1 *LASSO.*

To understand how to optimize the NIPA [16], we first have to understand NIPA by examining the working of the algorithm. The NIPA finds its base functionality in the LASSO [18] (Equation 2). With this algorithm we are minimising the sum of squared elements which is subjected to an ℓ_1-norm penalty. By using this constraint in the equation, it is possible for LASSO to shrink coefficients to zero, creating a sparse matrix of the coefficients and because of this, it can also be regarded as a variable selection method [12].

LASSO can achieve the shrinkage to zero because of the penalty function $\rho \sum_i |\beta_i|$ which results in a constraint with sharp edges (see Figure 1). When ρ is large, all the coefficients will go to zero and when ρ goes to zero, the constraint will be non-existent and the coefficients will equal the OLS estimation.

3.2.2 *NIPA.*

How LASSO is integrated into NIPA has been shortly introduced in Section 3.1.1. Where for each region i there is a specific regularization parameter ρ_i. As mentioned in the previous section, when this parameter is too high, it will result in all infection probabilities β_{ij} for regions i and j to be zero and when ρ_i is too small, the resulting matrix will not be a sparse representation of the infection probabilities between the regions. To confirm the correct regularization parameter is chosen without overfitting the model, Prasse et al. established a range where $10^{-4}(2||F_i^T V_i||_\infty) \leq \rho_i \leq 2||F_i^T V_i||_\infty$ [16]. Within this range, it is possible for all coefficients to go to zero and at the same time preventing the infection probabilities to be above zero for every β_{ij}.

3.2.3 *Simulated Annealing.*

Instead of CV, which is often used for determining the regularization parameter for LASSO [12], we propose using SA, introduced in Section 3.1.3, to find the optimal value for the regularization parameter ρ_i for every region i. The first part for answering **RQ1** is whether to implement SA ourselves or use a third-party library. Due to the scope and time-constraint of the research, we have chosen for the latter. The implementation of the SA algorithm has been done through the method dual_annealing

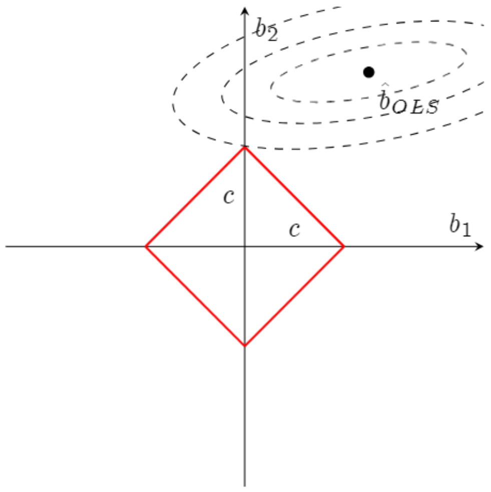

Fig. 1. The LASSO ℓ_1 constraint (red) in a two-dimensional parameter space (b_1 and b_2), where $\hat{b}_{OLS}$ is the (unconstrained) ordinary least square estimate and the contours show the estimates of b with equal variance in terms of squared error loss. c corresponds to the constraint in Equation 1.

of the SciPy library [21]. Dual annealing uses the generalization of CSA and FSA and combines it with a local search on each accepted location. As mentioned in Section 3.1.3, this implementation of SA uses a distorted Cauchy-Lorentz visiting distribution (see Equation 14), where parameter q_v also controls the temperature schedule as seen in Equation 15. For the acceptance probability, it utilizes Equation 16. The second part of answering the first research question and to make the implemented algorithm work as efficient as possible, is determining correct values for the parameters q_v, q_a, the initial temperature T_0 and the starting point x_0.

For the parameter q_v, that controls the visiting distribution, it is important that it is not too low ($q_v \leq 1$), because the visiting distribution will be confined to a local search space, as seen in the upper panel of Figure 2. If $1 < q_v \leq 2$ (FSA), we get a semi-local search where the search is more efficient than CSA but getting trapped at a local region can still occur (middle panel in Figure 2). To search the space homogeneously, we set $q_v = 2.62$ which creates the Tsallis-Stariolo form of the Cauchy-Lorentz distribution. With this form of the Cauchy-Lorentz, the search has the possibility of long jumps even at low temperatures. Due to this characteristic, it has a high probability of finding the global minimum instead of being trapped at a local minimum [22].

The value of the parameter that controls the acceptance probability, q_a, is not of big importance, but this value determines the initial value for the temperature. We have chosen for $q_a = -5$ as this is also proposed by Tsallis and Stariolo [19]. With this value we can determine a suitable value for the initial temperature. The following steps have been taken to calculate T_0 [5, 6]:

(1) Perform 100 steps of the algorithm.
(2) Compute the average difference in energy ($\hat{\Delta E}$).

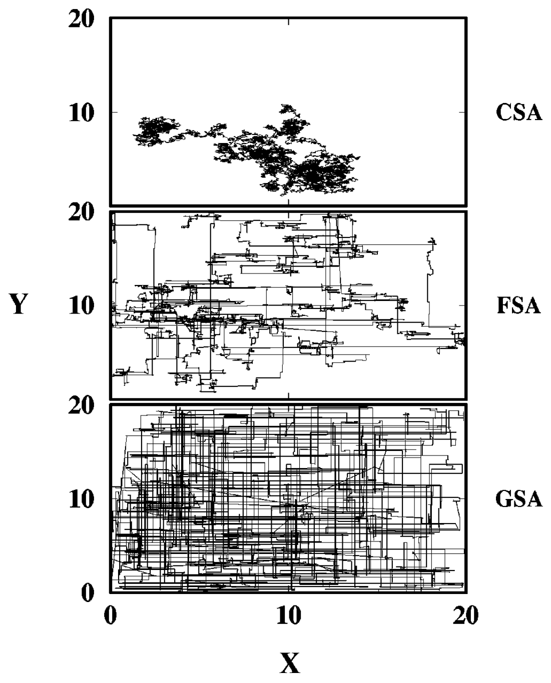

Fig. 2. Two-dimensional visiting distribution at a low temperature. Gaussian distribution for CSA ($q_v = 1$) in the upper panel, Cauchy-Lorentz distribution for FSA ($q_v = 2$) in the middle panel, and Tsallis-Stariolo form of Cauchy-Lorentz distribution for GSA ($q_v = 2.62$) in the lower panel. GSA visits the phase space homogeneously [22].

(3) Choosing an initial acceptance probability.
(4) Calculate the initial temperature T_0.

Where the initial acceptance probability has been chosen to be 0.8 as to allow the algorithm to make greater jumps at the start. If the initial acceptance probability is lower, the algorithm will not be able to make long jumps and when the probability is higher, it will accept too many bad moves [11]. We have chosen for this initial probability, because the starting point of the algorithm will be the lower bound of the range for the regularization parameter given in Section 3.2.2. This starting point has been picked due to the large probability that the optimal value is in close proximity and that the algorithm does not quickly go to the search space where all values of the infection probability matrix go to zero.

Finally, we calculate the initial temperature by inserting the values mentioned above into equation 16[1]. Due to the scope of the research, we assume that $K = 1$, preventing another optimization problem that would take up valuable time. The algorithm is then run over the NIPA model, the result is a local minimum, with a high probability of being the global minimum.

4 RESULTS

To get the results, we have implemented the NIPA algorithm with CV and the NIPA with SA with the parameters mentioned in Section 3.2.3.

[1] $T_{0,q_a} = 0.3693670547881299$

Regions	Cross-Validation	Dual Simulated Annealing
Wuhan	4.701e-11	1.130e-10
Huanggang	8.591e-09	4.408e-12
Jingzhou	3.537e-12	3.537e-12
Xiangyang	2.475e-12	2.475e-12
Xiaogan	1.266e-11	6.630e-12

Table 1. The regularization parameter that has been chosen by CV and by the DSA algorithm rounded to three decimal places. For the five regions in Figure 3c.

Regions	Cross-Validation	Dual Simulated Annealing
Wuhan	2.524e-01	2.322e-01
Huanggang	1.716e-01	6.363e-01
Jingzhou	8.990e-01	8.990e-01
Xiangyang	6.969e-01	6.969e-01
Xiaogan	6.363e-01	5.353e-01

Table 2. The curing probabilities that has the lowest MSE with the regularization parameter chosen by CV and by the DSA algorithm rounded to three decimal places. For the five regions in Figure 3c.

4.1 Hubei, China

The results will look at the difference in evaluation in the Chinese province of Hubei. The first case was on January 21 and the data lasts until February 14, since the local government changed their diagnosing policy causing an erratic increase in the number of reported cases on February 15 [1].

For the evaluation of both methods, we removed data points for a fixed amount of days m. The NIPA model with CV and with SA are then iterated over these m days. How the disease truly evolves (cumulative) can be seen in Figure 3 by the *blue* line. In the same figure, the predictive ability of NIPA with CV and NIPA with SA (*red* and *green* respectively) can be seen against each other. In Figure 3, it can be seen that both methods have approximately the same predictive results. To clearly see what the differences are between the two methods, we look at the evaluation metrics in Figure 4. These graphs show that for most predictions there is a slight improvement when NIPA is combined with the SA algorithm, except when the NIPA needs to predict two days ahead (Figure 4b). The chosen regularization values for the five regions in Figure 3 are in Table 1. In Table 2, it is shown which curing probability was optimal for the region with the chosen regularization parameter found in Table 1.

Besides the predictive ability of both methods, we also want to look at the computational difference. We run the algorithm multiple times with both optimization techniques. It took NIPA with CV, *1 minute and 7 seconds* to infer the infection probability matrix of Hubei, China. Using NIPA with SA, it took on average *28 minutes and 17 seconds* to infer the matrix. This big difference in computation time can be attributed to the fact that CV can be executed in parallel while SA has to wait for each iteration, to execute the next. This has a big impact on the performance. The many iterations SA needs to find the optimal value can be another reason why the computation time for SA is higher. Where CV only tries an x amount of values within the range, SA will search throughout the range.

5 DISCUSSION AND FUTURE WORK

Due to a limited time frame for completing this research paper, parts of the research could be extended or more thoroughly researched. In this section, we are going to discuss what these parts are.

5.1 NIPA

As this research paper is a continuation of a previous bachelor thesis, the assumption was made at the beginning that the NIPA algorithm was already implemented and the research of this paper could focus on the optimization of the algorithm through SA. This was however not the case and due to this problem, the first weeks of the research has been spend in the development and implementation of the NIPA algorithm. There was an algorithm found online[2] and after some research into the code, we found that the code did not contain the constraints to which NIPA is subjected ($0 \leq \beta_{ij} \leq 1$ and $\sum_{j,j\neq i}^{N} \beta_{ij} \leq 1$). For these reasons, we have decided to first focus on our own NIPA algorithm and when comparing our results with other papers, we found a small difference. This can be attributed to the parsing of the data, where the other papers used Matlab for their implementation and we have used Python. For these reasons, the assumption was made that our implementation was correct.

5.2 Optimization Method

At the beginning of the research, we have delved into two different types of optimization for the LASSO algorithm, SA and Bayesian optimization. At first, the latter was found to more appropriate because of the way the LASSO estimates can be interpreted as a posterior mode. Because of this relation, a Bayesian LASSO has been proposed by Park and Casella [15]. In this approach, we could use the marginal maximum likelihood or hyperpriors for choosing the regularization parameter ρ_i. We could even use a Gaussian process (GP) to very closely approximate the optimal value for the Bayesian LASSO which is called BO-GP [23]. We have not chosen for this method as this would mean that we have to rewrite LASSO, NIPA and then implement the BO-GP algorithm and since this research paper has limited time and scope, we have decided to go for the SA algorithm.

5.3 Further Optimization

The NIPA algorithm consists of two variables which are unknown at the beginning of the model fitting. The regularization parameter ρ_i and the curing probability δ_i for each region i. SA is a good algorithm to optimize problems with a parameter dimension bigger than one, see Figure 2. This would also lessen the time it will take for inferring the matrix, this comes from the fact that we are now iterating and performing SA for every curing probability that is in the set of possible curing probabilities. We have chosen to focus on an one dimension parameter space (ρ_i) due to the time limits of the research.

[2]https://github.com/DVL-Sejong/NIPA

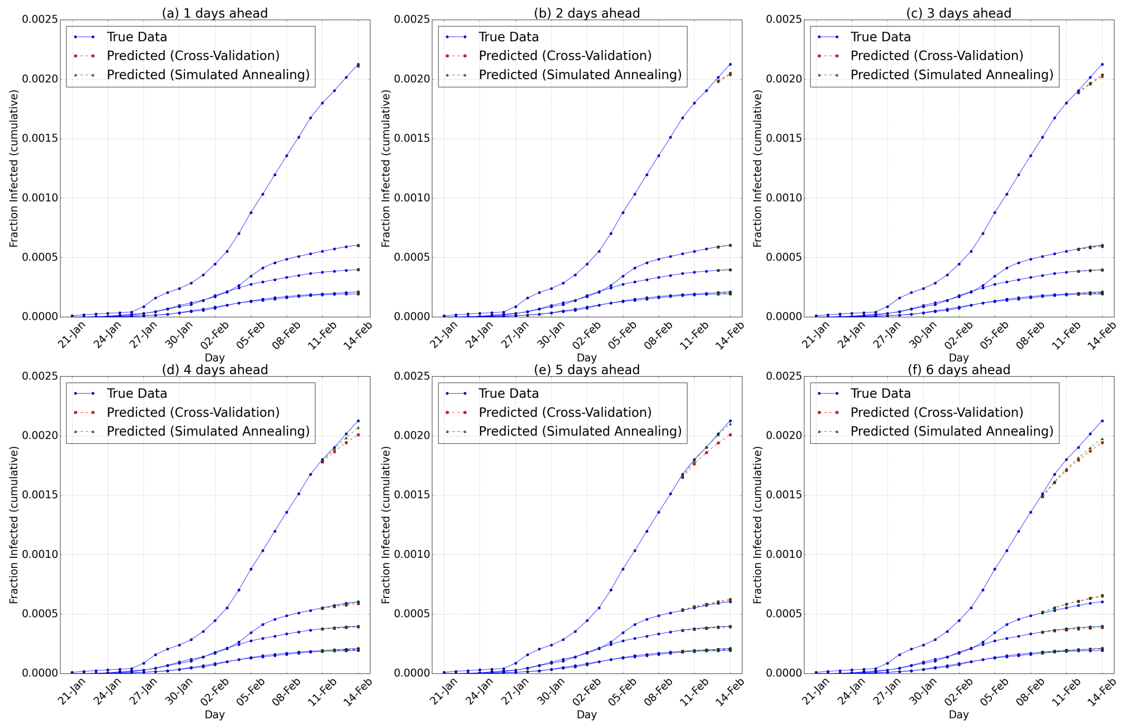

Fig. 3. The prediction of COVID-19 pandemic in Hubei by NIPA with CV and NIPA with SA. Here $N = 16$, but only five are shown. The amount of days predicted is in the title of the graph where $1 \leq m \leq 6$. Each point is the cumulative COVID-19 cases in Hubei.

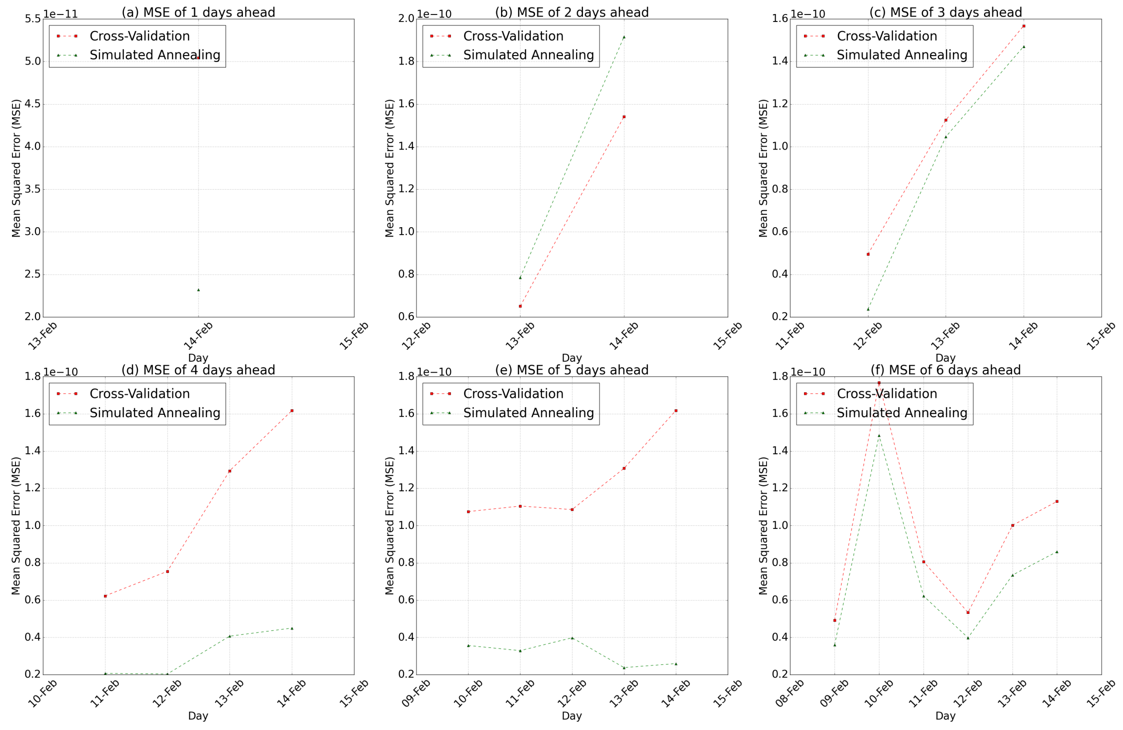

Fig. 4. The evaluation metric of the predictive ability of NIPA with CV and NIPA with SA. The amount of predicted days which is evaluated can be found in the title of the graph where $1 \leq m \leq 6$. Each point corresponds to the MSE between the predicted amount and the true amount of cases (lower is better).

5.4 Parameter Tuning

Like the NIPA, the SA algorithm has parameters that can be tuned for better and more efficient results. During this research we have tuned these parameters according to research and some trial-and-error. With more time, it would be possible to experiment more with different values and combinations of these values.

5.5 Static and Dynamic NIPA

In this research, we have chosen to implement SA only for the original NIPA. In previous research, by Achterberg et al., the formulation of NIPA was extended to include knowledge of the underlying contact network that was split into two algorithms, one where the prior knowledge is static (*NIPA static prior*) and one where it is dynamic (*NIPA dynamic prior*) [1].

5.6 Future Research

For future research, it would be interesting to see the results from more countries as well as seeing the SA algorithm be implemented for the NIPA static prior and the NIPA dynamic prior that were introduced by Achterberg et al. in their research.

As mentioned above in Section 5.3, it is possible to optimize the curing probability δ_i and to integrate this into the SA algorithm. With this addition it could be possible to even further optimize the NIPA.

6 CONCLUSION

In this research, we propose to use the NIPA prediction method with SA. Based on previous research, one region has been chosen to compare the results between NIPA with CV and NIPA with SA: Hubei, China. The results for this region provided valuable information about the use of NIPA with SA.

First, the incorporation of SA in the NIPA. As we can see in the results, we can safely assume that the implemented SA works accordingly. The values of the parameters chosen in Section 3.2.3 are efficient enough to search for an optimal value for the regularization parameter even though the computation time is slower compared to CV.

The results that are shown in Section 4, show that the NIPA with CV and the NIPA with SA show close predictive ability. Looking closer at the predictions, the NIPA with SA has shown better accuracy in predicting the cases of COVID-19. As can be seen in the evaluation metrics in Figure 4, where we see that the NIPA with SA has better evaluations for each of the results except for predicting two days ahead. These results are dwarfed by the difference in computation time, where the NIPA with CV is approximately 25× faster than the NIPA with SA.

We have shown that the SA algorithm improves the prediction ability against the CV technique which is used to choose a value for the regularization parameter for the NIPA. Although it has a better prediction accuracy, the computation time for the NIPA with SA is remarkably slow in contrast to the NIPA with CV. To determine which needs to be used, is a trade-off between accuracy and computation time.

ACKNOWLEDGEMENT

This research would not be possible without the helpful contributions and discussions with Alberto Garcia-Robledo and Mahboobeh Zangiabady. We thank them for their work on this research.

APPENDIXES

A LIST OF NOTATIONS

Notation	Equation(s)	Definition
c	1	The constraint of the LASSO
ρ, ρ_i	2, 11	Regularization parameter (of region i)
β, β_i	1, 2, 11	The OLS estimates of the LASSO, also known as the regression coefficients (of region i)
v_i	3	The viral state of the SIR-model of region i
N	4, 7, 11	The total amount of regions in the dataset
$\mathcal{I}_i$	3, 4, 9, 10	The fraction of infected individuals of region i in the SIR-model
$\mathcal{R}_i$	3, 4, 5	The fraction of recovered individuals of region i in the SIR-model
$\mathcal{S}_i$	3, 6, 9	The fraction of susceptible individuals of region i in the SIR-model
β_{ij}	4, 7, 8, 11	The infection probability between region j to region i
ΔE	12, 13, 16	The difference in energy (cost) between two locations in the SA algorithm
T_k	12, 13	The temperature of the SA algorithm at time step k
q_v	14, 15	The parameter that controls the Cauchy-Lorentz distribution and the rate of cooling in GSA
T_{q_v}	14, 15	The artificial temperature of the visiting distribution in GSA
q_a	16	The parameter that controls the acceptance probability in GSA
T_{q_a}	16	The artificial temperature of the acceptance probability in GSA

B AI USAGE

During the preparation of this work the author used ChatGPT in order to summarize the paper for writing the Abstract. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the content of the work.

REFERENCES

[1] Massimo A. Achterberg et al. “Comparing the accuracy of several network-based COVID-19 prediction algorithms”. en. In: *International Journal of Forecasting* 38.2 (Apr. 2022), pp. 489–504. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2020.10.001.

[2] Quentin Bertrand et al. “Implicit differentiation of Lasso-type models for hyperparameter optimization”. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 810–821.

[3] Dimitris Bertsimas and John Tsitsiklis. “Simulated Annealing”. In: *Statistical Science* 8.1 (Feb. 1993). ISSN: 0883-4237. DOI: 10.1214/ss/1177011077.

[4] Avrim Blum, Chen Dan, and Saeed Seddighin. “Learning Complexity of Simulated Annealing”. In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, Apr. 2021, pp. 1540–1548.

[5] Franco Busetti. “Simulated annealing overview”. In: May 2001.

[6] Bastien Chopard and Marco Tomassini. “Simulated Annealing”. In: *An Introduction to Metaheuristics for Optimization*. Series Title: Natural Computing Series. Cham: Springer International Publishing, 2018, pp. 59–79. ISBN: 978-3-319-93072-5 978-3-319-93073-2. DOI: 10.1007/978-3-319-93073-2_4.

[7] Thomas Guilmeau, Emilie Chouzenoux, and Victor Elvira. “Simulated Annealing: a Review and a New Scheme”. In: *2021 IEEE Statistical Signal Processing Workshop (SSP)*. Rio de Janeiro, Brazil: IEEE, July 2021, pp. 101–105. ISBN: 978-1-72815-767-2. DOI: 10.1109/SSP49050.2021.9513782.

[8] Rob J. Hyndman and Anne B. Koehler. “Another look at measures of forecast accuracy”. en. In: *International Journal of Forecasting* 22.4 (Oct. 2006), pp. 679–688. ISSN: 01692070. DOI: 10.1016/j.ijforecast.2006.03.001.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by Simulated Annealing”. en. In: *Science* 220.4598 (May 1983), pp. 671–680. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.220.4598.671.

[10] Diamantis Koutsandreas et al. “On the selection of forecasting accuracy measures”. en. In: *Journal of the Operational Research Society* 73.5 (May 2022), pp. 937–954. ISSN: 0160-5682, 1476-9360. DOI: 10.1080/01605682.2021.1892464.

[11] Sergio Ledesma, Gabriel Avia, and Raul Sanchez. “Practical Considerations for Simulated Annealing Implementation”. en. In: *Simulated Annealing*. Ed. by Cher Ming. InTech, Sept. 2008. ISBN: 978-953-7619-07-7. DOI: 10.5772/5560.

[12] Zitong Li and Mikko J. Sillanpää. “Overview of LASSO-related penalized regression methods for quantitative trait mapping and genomic selection”. en. In: *Theoretical and Applied Genetics* 125.3 (Aug. 2012), pp. 419–435. ISSN: 0040-5752, 1432-2242. DOI: 10.1007/s00122-012-1892-9.

[13] Shih-Wei Lin et al. “Parameter determination of support vector machine and feature selection using simulated annealing approach”. en. In: *Applied Soft Computing* 8.4 (Sept. 2008), pp. 1505–1512. ISSN: 15684946. DOI: 10.1016/j.asoc.2007.10.012.

[14] Kelly R. Moran et al. “Epidemic Forecasting is Messier Than Weather Forecasting: The Role of Human Behavior and Internet Data Streams in Epidemic Forecast”. en. In: *Journal of Infectious Diseases* 214.suppl 4 (Dec. 2016), S404–S408. ISSN: 0022-1899, 1537-6613. DOI: 10.1093/infdis/jiw375.

[15] Trevor Park and George Casella. “The Bayesian Lasso”. en. In: *Journal of the American Statistical Association* 103.482 (June 2008), pp. 681–686. ISSN: 0162-1459, 1537-274X. DOI: 10.1198/016214508000000337.

[16] Bastian Prasse et al. “Network-inference-based prediction of the COVID-19 epidemic outbreak in the Chinese province Hubei”. en. In: *Applied Network Science* 5.1 (Dec. 2020), p. 35. ISSN: 2364-8228. DOI: 10.1007/s41109-020-00274-2.

[17] Harold Szu and Ralph Hartley. “Fast simulated annealing”. en. In: *Physics Letters A* 122.3-4 (June 1987), pp. 157–162. ISSN: 03759601. DOI: 10.1016/0375-9601(87)90796-1.

[18] Robert Tibshirani. “Regression Shrinkage and Selection Via the Lasso”. en. In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58.1 (Jan. 1996), pp. 267–288. ISSN: 1369-7412, 1467-9868. DOI: 10.1111/j.2517-6161.1996.tb02080.x.

[19] Constantino Tsallis and Daniel A. Stariolo. “Generalized simulated annealing”. en. In: *Physica A: Statistical Mechanics and its Applications* 233.1-2 (Nov. 1996), pp. 395–406. ISSN: 03784371. DOI: 10.1016/S0378-4371(96)00271-3.

[20] Salih Tutun et al. “A Meta-heuristic LASSO Model for Diabetic Readmission Prediction”. In: Anaheim, California, May 2016.

[21] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[22] Y. Xiang and X. G. Gong. “Efficiency of generalized simulated annealing”. en. In: *Physical Review E* 62.3 (Sept. 2000), pp. 4473–4476. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.62.4473.

[23] Li Yang and Abdallah Shami. “On hyperparameter optimization of machine learning algorithms: Theory and practice”. en. In: *Neurocomputing* 415 (Nov. 2020), pp. 295–316. ISSN: 09252312. DOI: 10.1016/j.neucom.2020.07.061.

[24] Kai Zhang et al. “Annealed Sparsity via Adaptive and Dynamic Shrinking”. en. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, Aug. 2016, pp. 1325–1334. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939769.

[25] Enlu Zhou and Xi Chen. “Sequential Monte Carlo simulated annealing”. en. In: *Journal of Global Optimization* 55.1 (Jan. 2013), pp. 101–124. ISSN: 0925-5001, 1573-2916. DOI: 10.1007/s10898-011-9838-3.