Ensuring Efficient Convergence to a Given Stationary Distribution

Kenneth Peluso

September 2017 - April 2018

Abstract

How may we find a transition matrix that guarantees the long-run convergence of a Markov Chain to a given stationary distribution? Solving for this (usually) undetermined system is non-trivial and presents unique computational challenges. Five different of methods of directly solving for a transition matrix are presented along with their limitations. Relaxations of the two core assumptions underlying these direct methods - the Identityless and Independence Assumptions - are considered. A method of generating a Mass Matrix - the transition matrix underlying hops between entire population states - is described while developing the notion of successively-bounded weak compositions. An algorithm for their exhaustive generation is also presented. Applications of some methods are provided with respect to optimizing firm profit via optimally distributing workers among wage brackets and optimizing measures of national wealth via manipulation of class distribution and immigration policy. A generalization of all applications is formulated.

1 Introduction

Given a set of finite bins, a finite population to be allocated among the bins, and an objective function that is optimized on some set of population distributions across the bins, conventional optimization practices fall short of pragmatic. Integer programming and other methods may yield an optimal population distribution, but how may we guarantee that the population converges to such an optimal distribution? This paper attempts to answer that question.

At least one optimal population distribution is assumed to exist and be calculable. Select an optimal population distribution or a linear combination of such distributions and normalize it to 1. We assert that this distribution, a discrete probability distribution, is a stationary distribution for some Markov Chain. This Markov Chain is assumed to provide a complete, probabilistic description of the evolution of this population. Our goal is to define that Markov Chain. In particular, we seek to find a transition matrix, the *Individual Matrix*, whose first eigenvector (the eigenvector corresponding to an eigenvalue of 1) is the asserted stationary distribution.

For $n > 2$, this is an underdetermined problem - there exists far more variables than constraints. These variables are the elements of the desired transition matrix. The few constraints available to us consist of a system of linear equations arising from the matrix multiplication between the transition matrix and its stationary distribution, in addition to the requirement that all columns of the transition matrix must sum to 1. Due to the underdetermined nature of the problem, many transition matrix solutions may exist, but finding them is nontrivial.

We begin by clearly defining the problem along with its assumptions. Methods of directly finding a solution given these assumptions are presented. Some methods admittedly perform better than others, but all are presented regardless. Those which require no more than 2 bins are detailed first, those lacking that requirement follow. A brief aside regarding the performance of each algorithm accompanies its description and pseudocode.

As the paper continues, we relax the 2 core assumptions while providing motivation for a tangentially related problem - that of finding the *Mass Matrix*, to be defined later. To generate this matrix, new definitions are required. In particular, we must generalize slightly beyond the restricted weak compositions of primary concern in Page 2012 [Pag12]. Algorithms comparable to those in Page 2012 are created to generate instances of this new definition, of *successively-bounded weak compositions*.

We continue by surveying some applications of our methods used to directly solve for the Individual Matrix. We discuss methods to ensure optimal convergence of...

- the distribution of citizens in a society among income brackets to maximize GDP or overall welfare.
- the distribution of employees in a firm among wage brackets to maximize the firm's profit.
- the distribution of citizens among economic classes to an ideal level of inequality in a nation.
- the genetic diversity of a nation to an ideal level to maximize income-per-capita.

For these applications, we reassert the two core assumptions – the Identityless and Independence Assumptions. The latter two applications build off of the work of Oded Galor and his colleagues, and connections are drawn between their previous work and this new material. A generalization of all possible applications concludes the application section.

We conclude with a summary of all that has been discussed and suggestions for future research.

2 Problem Definition

Our problem is to converge to some optimal distribution of a finite population among a finite set of unique bins, where individuals are allowed to hop between bins. Let our population size and the number of bins be some integers N and n, respectively. The optimal distribution of the N individuals, when normalized, forms a discrete, n-length probability distribution we will call π. For now, we assume that all individuals are non-unique - the *Identityless Assumption*. Therefore, all individuals hop between bins according to the same set of rules – the same probability distribution – and we can disregard the value of N. We will impose one more assumption, namely that any individual, regardless of their current bin-placement, cannot affect the movement of any other individual - the *Independence Assumption*.

With this infrastructure of assumptions in place, we will attempt to ensure convergence to π by finding a Markov Chain, characterized by some $n \times n$ transition matrix $\mathbb{P}^{(I)}$, that converges to π. We call $\mathbb{P}^{(I)}$ the *individual transition matrix*. If one individual abides by a $\mathbb{P}^{(I)}$ that converges to π, then we know that the entire population converges to π by the Identityless and Independence Assumptions. We can now clearly state the problem to be solved:

$$\textbf{Find an } n \times n \textbf{ transition matrix } \mathbb{P}^{(I)} \textbf{ satisfying } \mathbb{P}^{(I)}\pi = \pi. \tag{1}$$

In other words, we seek a matrix $\mathbb{P}^{(I)}$ with stationary distribution π. Let us now delve deeper into the meaning of $\mathbb{P}^{(I)}$. Consider the following generalized $\mathbb{P}^{(I)}$:

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}$$

What do each of these p_{ij} values mean? Each is the probability of any single individual hopping from bin j to bin i. We assumed that every individual abides by this same $\mathbb{P}^{(I)}$ (courtesy of the Identityless Assumption)

and that the movements and placements of individuals do not affect those of other individuals (courtesy of the Independence Assumption). Therefore, under these two, core assumptions, we may completely describe the evolution of our population once we obtain $\mathbb{P}^{(I)}$. We must now solve for this $\mathbb{P}^{(I)}$.

Note that the given stationary distribution π is an argument for all algorithms to follow.

3 Finding $\mathbb{P}^{(I)}$ Directly when $n = 2$

In this section, we survey multiple means of directly calculating a $\mathbb{P}^{(I)}$ when $n = 2$, namely Binary Search, Random Search, and Binary-Random Search. Aside from working well when $n = 2$, all the following methods will become important in the Gibbs-Inspired Method, defined later, which can tolerate situations where $n > 2$.

We must first begin with a derivation, atop of which we will construct our 3 "$n = 2$"-case methods. Let us begin by setting $n = 2$. Then Equation (1) is a system of 2 linear equations with 2 unknowns, namely elements p_{11} and p_{22} of $\mathbb{P}^{(I)}$. These equations are:

$$p_{11}\pi_1 + (1 - p_{22})\pi_2 = \pi_1$$
$$(1 - p_{11})\pi_1 + p_{22}\pi_2 = \pi_2$$

Either of these equations can be used to arrive at the following equation:

$$p_{11} = 1 - (1 - p_{22})\frac{\pi_2}{\pi_1} \tag{2}$$

but we must only accept values of $p_{11} \geq 0$ from this line, so we must impose the following constraint:

$$p_{22} \geq 1 - \frac{\pi_1}{\pi_2} \tag{3}$$

We heavily rely on both Equation (2) and Constraint (3) in upcoming algorithms.

3.1 Binary Search (bS) for $n = 2$ Case

3.1.1 Description

Let $n = 2$ and let `ev()` be a function that inputs a matrix and outputs its eigenvector corresponding to an eigenvalue of 1. Let `avg()` take a list of numbers as input and output their average. We may now adapt the classic Binary Search algorithm for our purposes.

3.1.2 Algorithm

1. If $\pi_1 \geq \pi_2$:
   $b_1 \coloneqq 0$
   Else:
   $1 - \frac{\pi_1}{\pi_2}$
2. $b_2 \coloneqq 1$; $iters \coloneqq 2000$; $output \coloneqq \begin{bmatrix} x_{11} & 1-p \\ 1-x_{11} & p \end{bmatrix}$
   where $p = \texttt{avg}([b_1, b_2]), x_{11} \sim U[0, 1]$
3. While $output * \texttt{ev}(output) \neq \pi$ and $iters > 0$:

4. If $\texttt{ev}(output)_2 < \pi_2$:
5. $b_1 = \texttt{avg}([b_1, b_2])$
6. $output_{22}$ += $\texttt{avg}([b_1, b_2]) - b_1$
7. Else:
8. $b_2 = \texttt{avg}([b_1, b_2])$
9. $output_{22}$ −= $\texttt{avg}([b_1, b_2]) - b_1$
10. $output_{11} \coloneqq 1 - (1 - output_{22})\frac{\pi_2}{\pi_1}$
11. $iters$ −= 1
12. Return *output*

3.1.3 Performance

The worst case cost for implementing bS is $O(F(d)log_2(d))$, where $F(d)$ is the cost of calculating with arithmetic operations with d-digit precision. It should be noted that d in practice is determined by the threshold one may use to determine if 2 floating-point numbers are equal and by the user's machine and programming languages of choice.

Proof. Let x be the maximal number of times we can bisect a number, d. We are always bisecting our search space of d in half until we can no longer do so, so:

$$1 = d/2^x \Rightarrow log_2(d) = x$$

Furthermore, if any d-precision arithmetic operation costs $F(d)$ units, then the worst-cost cost incurred is $xF(d)$. Thus, the worst-case runtime of bS is $O(F(d)log_2(d))$. □

3.2 Random Search (rS) for $n = 2$ Case

3.2.1 Description

In bS, the next value of $output_{22}$ to be parsed and checked is the point that bisects the search space. This occurs in Steps 5, 6 and 7, 8 of the bS algorithm. In rS, this value is sampled from a uniform distribution spanning a fixed search space, [0,1]. This is found in Step 4 of the following rS algorithm.

3.2.2 Algorithm

1. If $\pi_1 \geq \pi_2$:
   $b_1 \coloneqq 0$
   Else:
   $1 - \frac{\pi_1}{\pi_2}$
2. $b_2 \coloneqq 1$; $iters \coloneqq 2000$; $output \coloneqq \begin{bmatrix} x_{11} & 1-p \\ 1-x_{11} & p \end{bmatrix}$
   where $p = \texttt{avg}([b_1, b_2]), x_{11} \sim U[0, 1]$
3. While $output * \texttt{ev}(output) \neq \pi$ and $iters > 0$:
4. $output_{22} = (b_2 - b_1) \cdot u + b_1$ where $u \sim U[0, 1]$
5. $output_{11} = 1 - (1 - output_{22})\frac{\pi_2}{\pi_1}$

6. $iters \mathrel{-}= 1$
7. Return *output*

3.2.3 Performance

The worst-case complexity for implementing rS is $O(F(d)10^d)$ with low probability, where $F(d)$ is the cost of calculating with arithmetic operations with d-digit precision.

Proof. Consider a complete Markov Chain of size $10^d \times 10^d$ with a transition matrix filled entirely of uniform distributions. That Markov Chain is analogous to iterating within rS, since each sample u in Step 4 of rS is independent and uniformly random across all 10-digit combinations (digits 0-9) of d-digits. The probability of hitting the correct value is always fixed at $\frac{1}{10^d}$. We can call this probability p and use it as a parameter for a geometric distribution. The expected amount of steps until the correct value is reached is thus the expected value of this geometric distribution, which is $\frac{1}{p} = 10^d$. For each of these steps, an arithmetic cost of $F(d)$ is incurred, but with a variance of $\frac{1-p}{p^2} \gg 0$. Thus the worst-case complexity is $O(F(d)10^d)$ with low probability. □

3.3 Binary-Random Search (brS) for $n = 2$ Case

3.3.1 Description

brS combines the efficiency of bS with the "open-mindedness" of rS. Instead of sampling uniformly from a fixed search space, as in rS, brS samples uniformly from the "remaining search space." The "remaining search space" is calculated by bisecting the previous iteration's "remaining search space," as in bS. Take note of Steps 4 through 9 to observe this in action:

3.3.2 Algorithm

1. If $\pi_1 \geq \pi_2$:
   $b_1 \coloneqq 0$
   Else:
   $1 - \frac{\pi_1}{\pi_2}$
2. $b_2 \coloneqq 1$; $iters \coloneqq 2000$; $output \coloneqq \begin{bmatrix} x_{11} & 1-p \\ 1-x_{11} & p \end{bmatrix}$
   where $p = \texttt{avg}([b_1, b_2]), x_{11} \sim U[0,1]$
3. While $output * \texttt{ev}(output) \neq \pi$ and $iters > 0$:
4. If $\texttt{ev}(output)_2 < \pi_2$:
5. $b_1 = \texttt{avg}([b_1, b_2])$
6. $output_{22} \mathrel{+}= (b_1 - b_2) \cdot u$ where $u \sim U[0,1]$
7. Else:
8. $b_2 = \texttt{avg}([b_1, b_2])$
9. $output_{22} \mathrel{-}= (b_1 - b_2) \cdot u$ where $u \sim U[0,1]$
10. $output_{11} = 1 - (1 - output_{22})\frac{\pi_2}{\pi_1}$
11. $iters \mathrel{-}= 1$
12. Return *output*

3.3.3 Performance

The worst-case complexity for implementing brS is $O(F(d)log(d))$ with low probability, where $F(d)$ is the cost of calculating with arithmetic operations with d-digit precision. The proof follows the same reasoning as that of bS, because, on average, each remaining search space is expected to be bisected.

3.4 $n = 2$ Direct Method Performance Review

In this section, we list the runtime complexities (the worst-case runtime for deterministic methods and expected runtime for probabilistic methods) for all "$n = 2$"-case direct methods. We then show their real-time performances for various parameter values.

3.4.1 Summary of Complexities

In practice, rS, bS, and brS all finish execution within a few tenths of a millisecond. All direct methods were tested on a computer with a single processor with values of $n \in 2, \ldots, 14$.

Method	Complexity
rS	$O(F(d)10^d)$
bS	$O(F(d)\log(d))$
brS	$O(F(d)\log(d))$

3.4.2 Real-Time Runtimes

All direct methods have a threshold parameter ϵ that determines whether or not our solutions is sufficiently correct. In particular, we run each algorithm until

$$\|P\pi - \pi\| < \epsilon$$

where P is the candidate value for $\mathbb{P}^{(I)}$ for a given π. In this section, we vary that ϵ parameter and note the differences in real-time runtimes that consequently surface between "$n = 2$"-case direct methods. Figure (1) plots the real-time runtimes for bS, rS, and brS for various epsilon values. For each epsilon, we average the real-time runtimes of each algorithm across 300 trials on the same "$n = 2 \times 1$"-size π. In Figure (2), we plot the same information with 100,000 trials. The anticipated indirect relationship is far more apparent in Figure (2). In both cases, rS performs relatively worse than bS and brS though not by much, and brS performs slightly better than bS.

4 Finding $\mathbb{P}^{(I)}$ Directly when $n \geq 2$

In this section, we survey 2 means of directly calculating a $\mathbb{P}^{(I)}$ when $n \geq 2$, namely GRS and GI.

4.1 Greedy Random Search (GRS)

4.1.1 Description

GRS may be considered a generalized brS method. Let `normalize`(P) take a matrix P and output the same matrix but with all columns normalized to 1. Let `randInt`($1 : n$) return a single iid samples from the

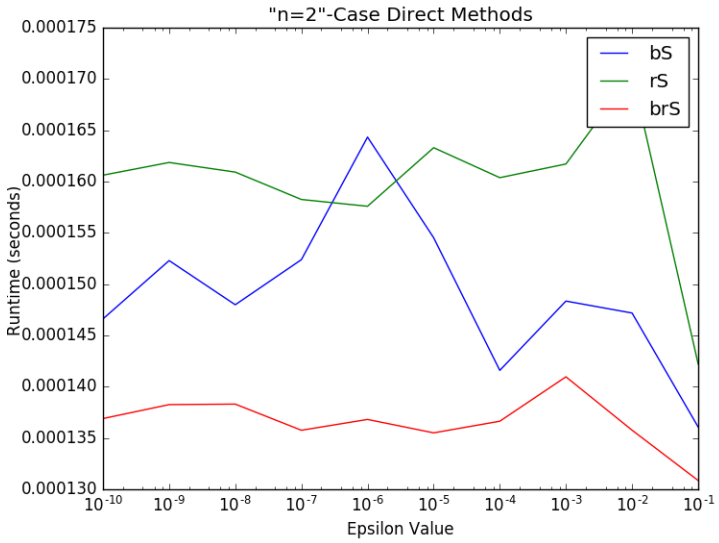


Figure 1: With 300 trials

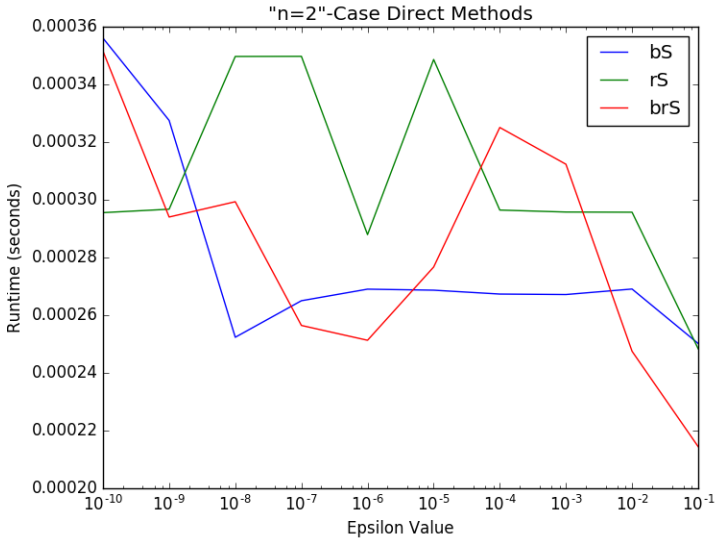


Figure 2: With 100,000 trials

discrete uniform distribution with support $\{1, \ldots, n\}$.

The intuition behind this algorithm is as follows. If v_1 is the first eigenvector of $\mathbb{P}^{(I)}$ and if $v_1[a] < \pi[a]$, then we ensure that the probability of staying at node a is greater than what it currently is. Likewise, if $v_1[a] > \pi[a]$, then we ensure that the probability of staying at node a is less than what it currently is. This logic was implemented, and, sadly, its effectiveness is limited. It rarely ever converges in a timely manner, hence why the number of iterations is limited.

Line 5 holds the most complex part of the GRS Algorithm. In this line, we first ensure that if $\forall a \in$

$\{1,\ldots,n\}; \mathbb{P}^{(I)}[a,a] < 0$ or $\mathbb{P}^{(I)}[a,a] > 1$, then we do not alter $\mathbb{P}^{(I)}$ – if we did alter $\mathbb{P}^{(I)}$ in that situation, then our algorithm may irreparably diverge toward a nonsensical $\mathbb{P}^{(I)}$ value. Otherwise, we double-check that $\forall a \in \{1,\ldots,n\}; 0 < \mathbb{P}^{(I)}[a,a] < 1$, then alter $\mathbb{P}^{(I)}[a,a]$ by a factor of $(1+c)$, which is either 1.45 or 0.55 depending on whether or not we intend to increase or decrease $\mathbb{P}^{(I)}[a,a]$, respectively. The value of 0.45 multiplied to c in Line 4 was selected because it appeared to cause the most efficient convergence to solutions after some non-rigorous experimentation conducted by the author. Perhaps future research may find into the optimal value for this factor.

4.1.2 Algorithm

When analyzing this algorithm, remember that logical operations can be treated as integers in some programming languages, including Python e.g. $(1 < 2) \to 1, (1 > 2) \to 0$.

1. **Initialize** $\forall$ columns $\vec{p} \in \mathbb{P}^{(I)}, \vec{p} \sim U[0,1]^n$,
   $\mathbb{P}^{(I)} \coloneqq \texttt{normalize}(\mathbb{P}^{(I)})$
   $threshold \coloneqq 0.001$
   $iterations \coloneqq n * 100000$
2. **Until** $|\mathbb{P}^{(I)}\pi - \pi| < threshold$ **OR** $iterations == 0$:
3. $a = \texttt{randInt}(1:n)$
4. $c = 0.45 * (2 * (ev[a] < pi[a]) - 1)$
5. $\mathbb{P}^{(I)}[a,a] = \mathbb{P}^{(I)}[a,a] * \Big(((1+c) * \mathbb{P}^{(I)}[a,a] > 1) + ((1+c) * \mathbb{P}^{(I)}[a,a] < threshold)\Big)$
   $+(1+c) * \mathbb{P}^{(I)}[a,a] * ((1+c) * \mathbb{P}^{(I)}[a,a] <= 1) * ((1+c) * \mathbb{P}^{(I)}[a,a] > 0)$
6. $\mathbb{P}^{(I)} = \texttt{normalize}(\mathbb{P}^{(I)})$
7. $iterations \mathrel{-}= 1$
8. **Return** $\mathbb{P}^{(I)}$

4.2 Gibbs-Inspired Method (GI)

4.2.1 Description

This algorithm derives its name from its inspiration, the Gibbs Sampler discussed in Geman & Geman [GG84]. The Gibbs Sampler persistently samples from different conditional distributions, each of which is conditioned on all but a single, randomly selected parameter. Similarly, GI persistently calculates the correctness of a matrix after "optimizing" a small, randomly selected part of that matrix.

4.2.2 Algorithm

Define the function $S(\_, i, j)$ for $i < j$ to input a matrix or vector and output a matrix or vector, respectively, with only the "i,j-elements." Specifically:

$$S(\mathbb{P}^{(I)}, i, j) = \begin{bmatrix} p_{i,i} & p_{i,j} \\ p_{j,i} & p_{j,j} \end{bmatrix}$$

$$S(\pi, i, j) = \begin{bmatrix} \pi_i \\ \pi_j \end{bmatrix}$$

We also borrow the `normalize()` function from GRS and take `sum()` to be the column sum operation. For instance:

$$\texttt{sum}\left(\begin{bmatrix}1\\2\end{bmatrix}\right) = 3, \quad \texttt{sum}\left(\begin{bmatrix}1 & 3\\2 & 4\end{bmatrix}\right) = [3, 7]$$

Finally, note that on Line 5, rS and bS can be used in place of brS.

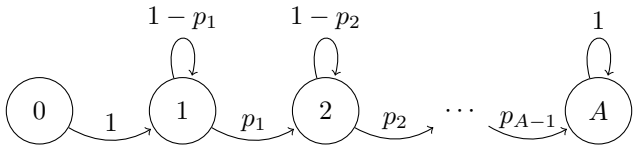
1. **Initialize** $\forall$ columns $\vec{p} \in \mathbb{P}^{(I)}, \vec{p} \sim U[0,1]^n$,
   $\mathbb{P}^{(I)} \coloneqq \texttt{normalize}(\mathbb{P}^{(I)})$
   $threshold \coloneqq 0.001$
   $iterations \coloneqq n * 100000$
2. **While** $\texttt{sum}(|\mathbb{P}^{(I)}\pi - \pi|) > threshold$ AND $iterations > 0$:
3. $i, j \sim U\{1, \ldots, n\} \ni i < j$
4. $\pi^* = \texttt{normalize}(S(\pi, i, j))$
5. $b = \texttt{sum}(S(\mathbb{P}^{(I)}, i, j))$
6. $P^* = \texttt{brs}(\pi^*)$
7. $\mathbb{P}^{(I)}[i, \_] = b_1 * P^*[i, \_]$
8. $\mathbb{P}^{(I)}[j, \_] = b_2 * P^*[j, \_]$
9. $iterations \mathrel{-}= 1$

4.2.3 Performance

In practice, GI performs extremely well for small n. Relative to the other methods, GI is still fast for larger n and also exhibits a relatively good accuracy.

The Gibbs Sampling algorithm is considered to sample from the intended distribution when all parameters have been updated at least once (or a constant factor number of times). We will make the same assumption with GI. Given d-digit precision and when $n = 2$, bS will, at worst, cost some amount in $O(F(d)log_2(d))$. For $n > 2$, all possible pairs of row indices from $\mathbb{P}^{(I)}$ will eventually be subjected to bS. The expected time at which this occurs will be the time we calculate.

Define a Markov Chain $X = (X_1, X_2, ..., X_T)$ with $\binom{n}{2} + 1$ nodes with associated values $S = \{0, 1, ..., \binom{n}{2}\}$. These values represent the number of unique pairs of the indices $1, 2, ..., n$ parsed by GI. After setting $A \coloneqq \binom{n}{2}$ and realizing that we uniformly choose among all possible pairs of indices, we can represent the Markov Chain graphically as follows:



where $\forall j \in \{0 \cdots A\}, p_j = \frac{A-j}{A}$. What we seek is $\mathbb{E}[T]$, where T is the total number of steps until GI completes. We embark on a *First-Step Analysis* to find $\mathbb{E}[T]$. If we set $v_j = \mathbb{E}[T|X_0 = j]$, and assert that we'll always be starting at $X_0 = 0$, then by the Law of Total Probability:

$$\mathbb{E}[T] = \sum_{j=0}^{A} v_j \mathbb{P}(X_0 = j) = v_0$$

because $\mathbb{P}(X_0 = 0) = 1$ and $\forall j = 1 : A, \mathbb{P}(X_0 = j) = 0$. Furthermore,

$$v_0 = 1 + v_1$$
$$\forall j \in \{1 \cdots (A-1)\}, v_j = 1 + (1 - p_j)v_j + p_j v_{j+1}$$
$$v_A = 0$$

All that is unknown are the conditional expectations v_j. Note that to solve for them is equivalent to solving the system of equations $\mathbb{M}\vec{v} = \vec{v}$, where:

$$\mathbb{M} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & \dots & \dots & 0 \\ 1 & 0 & 1-p_1 & p_1 & 0 & \dots & \dots & 0 \\ 1 & 0 & 0 & 1-p_2 & p_2 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & & 1-p_{A-1} & p_{A-1} \\ 0 & 0 & 0 & 0 & 0 & \dots & \dots & 0 \end{bmatrix}, \vec{v} = \begin{bmatrix} 1 \\ v_0 \\ v_1 \\ \vdots \\ v_A \end{bmatrix}$$

Note that the transition matrix for the Markov Chain X is embedded within $\mathbb{M}$ - if we remove the first column and last row of $\mathbb{M}$, we arrive at the transition matrix. In practice, it may be more computationally efficient to calculate the eigenvector of $\mathbb{M}$ corresponding to an eigenvalue of 1. Otherwise, we may continue to directly and analytically solve for v_0. We know from above that:

$$v_{A-1} = 1 + (1 - p_{A-1})v_{A-1} + p_{A-1}v_A = 1 + (1 - p_{A-1})v_{A-1}$$
$$\Rightarrow (1 + p_{A-1} - 1)v_{A-1} = 1$$
$$\Rightarrow v_{A-1} = \frac{1}{p_{A-1}}$$

Similarly,

$$v_{A-2} = 1 + (1 - p_{A-2})v_{A-2} + p_{A-2}v_{A-1} = 1 + (1 - p_{A-1})v_{A-1} + \frac{p_{A-2}}{p_{A-1}}$$
$$\Rightarrow (1 + p_{A-2} - 1)v_{A-2} = 1 + \frac{p_{A-2}}{p_{A-1}}$$
$$\Rightarrow v_{A-2} = \frac{1}{p_{A-1}} + \frac{1}{p_{A-2}}$$
$$\vdots$$
$$\Rightarrow v_1 = \sum_{j=1}^{A-1} \frac{1}{p_j}$$
$$\Rightarrow v_0 = 1 + v_1 = 1 + \sum_{j=1}^{A-1} \frac{1}{p_j} = \sum_{j=0}^{A-1} \frac{A}{A-j}$$
$$\therefore \mathbb{E}[T] = A(\psi^{(0)}(A+1) + \gamma)$$

[Wol]

where $\psi^{(0)}$ is the 0^{th} derivative of the digamma function (the digamma function itself) and γ is the Euler-Mascheroni Constant. Again, note that, just as in the Gibbs Sampling algorithm, this expectation holds up until some constant factor.

Note that this expected runtime can easily be divided by s, where $s \in \{1 \cdots \frac{n}{2}\}$ is the number of computational nodes (e.g. servers, processors) available for parallelization of GI. Pairs of indices from $1, \ldots, n$ would be uniformly sampled without replacement. Each sampled pair would be assigned to a node, and bS (or rS or brS) would be computed on each node. The results from all nodes would be aggregated back into a central node, which would repeat the process of assigning random pairs of indices to nodes. One can also borrow from the field of Deep Learning and implement *dropout*, whereby each sampled pair of indices undergoes bS on a node only if $x = 1$, where $x \sim \text{Bernoulli}(p)$ and p is usually between 0.5 and 0.8. Droupout is conventionally used to prevent neural networks from overfitting their supplied training data. In this case, the author has no reason to suspect that performance enhancements, whether in accuracy or efficiency, may not be realized by implementing dropout.

4.3 Direct Methods with Constrained Parameters

All methods allow us to find optimal $\mathbb{P}^{(I)}$ with constrained parameters, where some of the elements of $\mathbb{P}^{(I)}$ hold already-specified values and cannot be altered. We assume that the addition of constraints is intuitive for the "n=2"-case methods. GRS and GI may lack this intuitiveness in the face of constrained parameters.

4.3.1 Constrained GRS Example

Given a dictionary that lets us check constraints on elements indexed by column, the following procedure would be inserted between Line 5 and Line 6 of the GRS algorithm.

For each element with a constraint i in the selected column k, take $c_i \sim U[a_i, b_i] \ni 0 \leq a_i \leq b_i \leq 1$. Normalize the column by multiplying all non-constrained values in column k by $1 - \sum_i c_i$. Proceed as usual.

Note that when $a_i = b_i$, we have an equality constraint. Otherwise, we have an inequality constraint.

4.3.2 Constrained GI Example

Within GI, we persistently form various $n = 2$ subproblems. We check various permissible values for p_{22}, solve for p_{11} using p_{22}, return the result to the original problem and repeat. The space of possible values for p_{22} is already constrained to ensure that $p_{11} + (1 - p_{11}) = p_{22} + (1 - p_{22}) = 1$ and $p_{11}, p_{22} > 0$. We can simply further constrain this space to incorporate our additional constraints.

In the case of equality constraints, Equation (2) subject to Constraint (3) can be used to immediately recover p_{11} whenever p_{22} is given and vice versa. In the case of inequality constraints, we can still use Equation (2) but subject it to Constraint (3) in addition to whatever other constraints we must implement, and take pause when all constraints cannot be satisfied at once. Here is an example when all constraints can be simultaneously satisfied:

Suppose we must enforce that $p_{13} < \frac{1}{2}$, then whenever p_{13} is selected to be p_{11} in one of many $n = 2$ subproblems of a GI instance, we randomly select a value p_{22} such that:

$$p_{22} \geq 1 - \frac{\pi_1}{\pi_2}$$

and

$$p_{11} = 1 - (1 - p_{22})\frac{\pi_2}{\pi_1} < \frac{1}{2}$$

We can manipulate that second, additional constraint in the following manner:

$$\frac{1}{2} < (1 - p_{22})\frac{\pi_2}{\pi_1}$$
$$\frac{\pi_1}{2\pi_2} < (1 - p_{22})$$
$$p_{22} < \frac{\pi_1}{2\pi_2}$$

Therefore, we randomly choose a p_{22} such that:

$$0 \leq 1 - \frac{\pi_1}{\pi_2} \leq p_{22} < \frac{\pi_1}{2\pi_2} \leq 1$$

Here and in all cases, the values π_1 and π_2 determine whether or not such a p_{22} value even exists i.e. whether or not the constraints admit a *feasible* solution.

4.4 Real-Time Runtimes of "$n \geq 2$"-case Direct Methods

In this section, we list the runtime complexities (the worst-case runtime for deterministic methods and expected runtime for probabilistic methods) for all $n \geq 2$ direct methods – GRS and GI. We then show their real-time performances for various inputs. Remember that for all methods, iteration limits can be imposed to sacrifice accuracy for a quicker runtime.

4.4.1 Real-Time Runtimes

Here, we provides graphs showing the real-time runtimes for all "$n \geq 2$"-case direct methods for various n values. For each $n \in \{2, 3, 4, 5\}$, we plot the average runtime over 10 trials vs. the n-value.

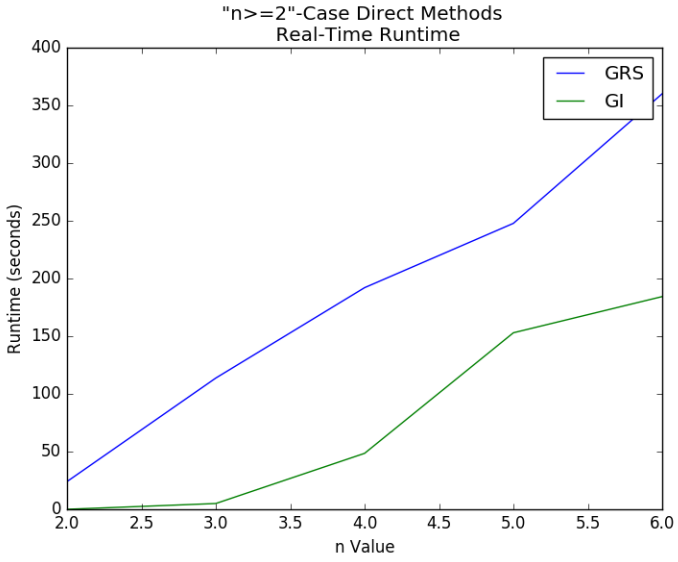


Figure 3: Real-time Runtimes of GRS and GI

4.4.2 GI vs. Expected-GI

The expected runtime of GI appeared to be predictive of the actual runtime of GI. Figures (4) and (5) show GI vs. the Expected Runtime of GI (labelled as "Expected GI"), and (6) notes the ratio between the actual runtime GI over the Expected GI. The author suspects that this ratio is a rugged, generally increasing function because of the presence of "outlier trials" where GI would seem to inexplicably "get stuck."

Sometimes, for the same n, GI would converge in less than a second for multiple trials. Occasionally, and especially as n increased, GI would appear to get "stuck" and either take a very long time to converge or not ever converge. The variance in seconds of real-time runtimes for repeated GI trials where $n = 5$ was an astounding 13151.8180598. Perhaps parallelizing GI, implementing dropout, or simply restarting GI if it exceeds a threshold time limit may help to mitigate this variance.

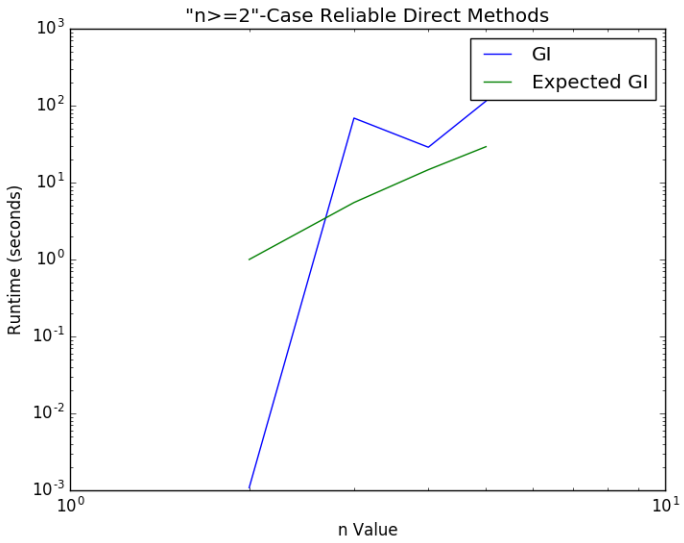


Figure 4: GI and Expected GI real-time runtime (logy vs. logx)

4.4.3 Proportion of Completed Trials

In the absence of parallelization, GRS and GI both perform miserably as n grows. We now attempt to capture the extent of this misery. Here, we plot the proportion of times the algorithm finishes running in $60n$ seconds for various n.

4.4.4 Accuracy Under Proportional Iteration Limits

Iteration limits can help our algorithms finish quickly at the expense of computational accuracy. We place iteration limits of $10000n$ for various n on both GRS and GI to observe the effects of these limits on the accuracy of these methods. We do not claim that linearly increasing iteration limits in proportion with n is representative of an efficient trade-off between accuracy and runtime – we implement it here for simplicity.

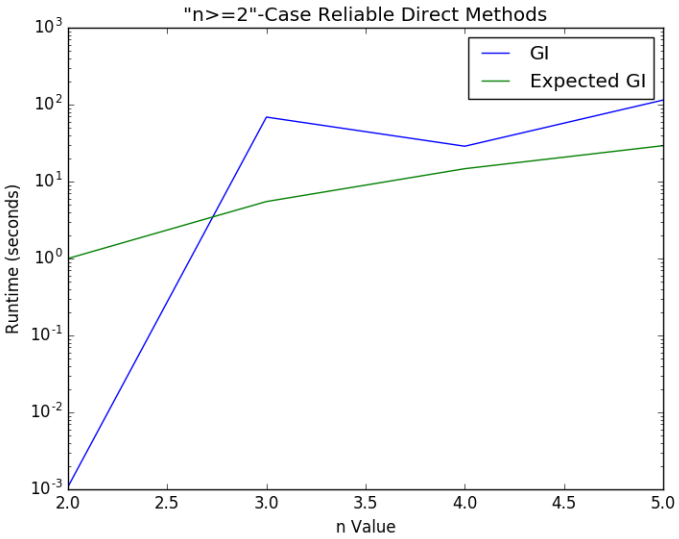


Figure 5: GI and Expected GI real-time runtime (logy vs. x)

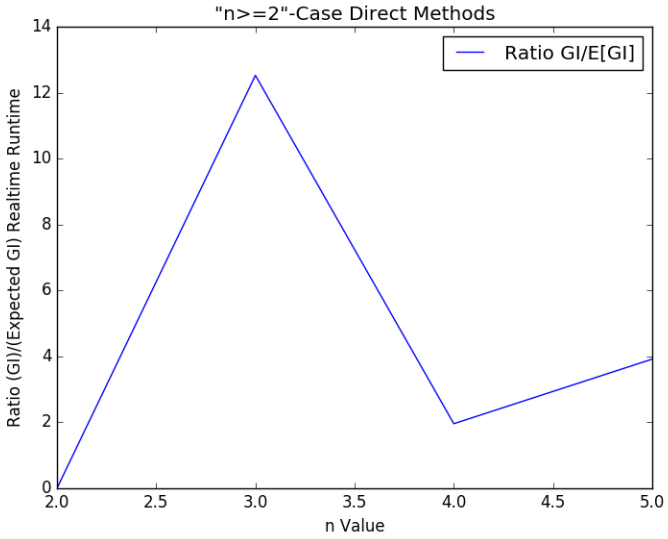


Figure 6: GI/Expected GI Ratio

We calculate accuracy using three methods: *exact distance*, *deviation distance*, and *cosine distance*. The former two are supplied by Sörensen, 2007, [Sör07].

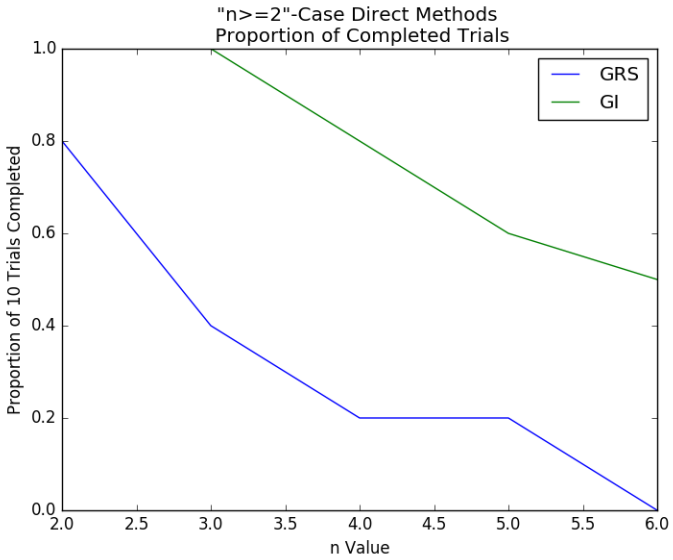


Figure 7: Average Proportion of Successful Trials of GRS and GI

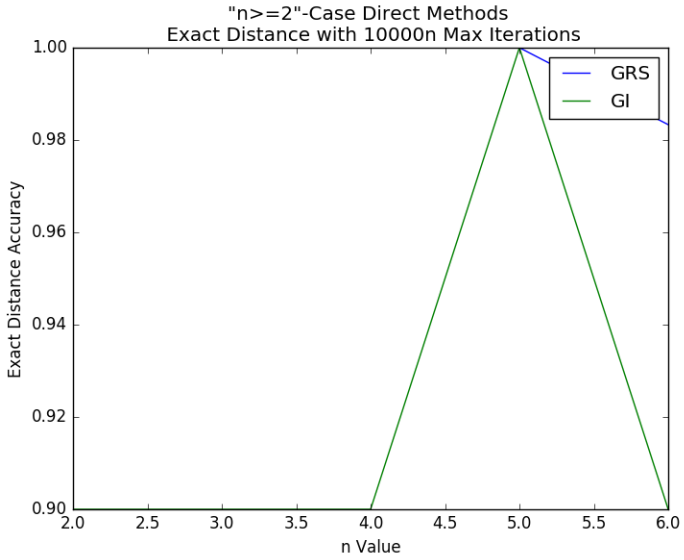


Figure 8: Iteration-Limited Accuracy of GRS and GI Measured by Exact Distance Metric

4.4.5 Insights

From the data, we learn that GRS under-performs GI in both runtime and efficacy. Under iteration limits, it is difficult to tell if GRS or GI leads to a better accuracy. We also note GI's relatively erratic behavior, even under iteration constraints, from its erratic change in average real-time runtime, average proportion of completions, and average accuracy with increasing n.

n	GRS	GI
2	1.0	0.9
3	1.0	0.9
4	1.0	0.9
5	1.0	1.0
6	0.98333333	0.9

Figure 9: Exact Distance Data for Figure 8

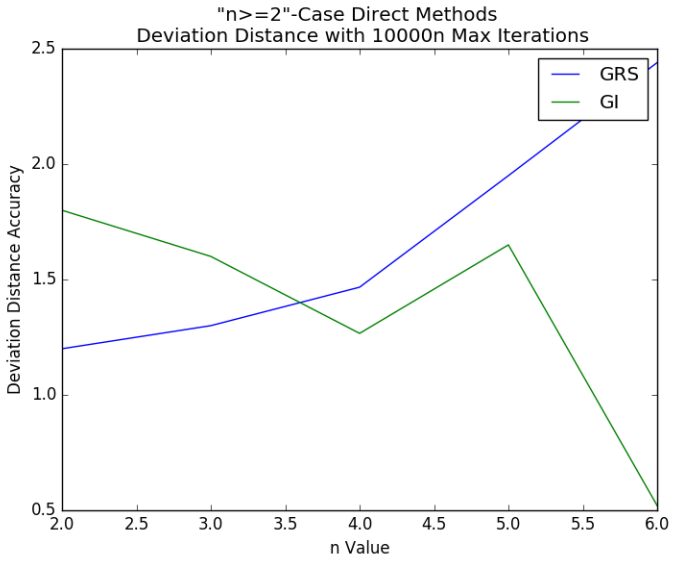


Figure 10: Iteration-Limited Accuracy of GRS and GI Measured by Deviation Distance Metric

5 Relaxing the Assumptions of Direct Methods

Here, we discuss the consequences of independently relaxing both core assumptions.

5.1 Relaxing the Identityless Assumption

By relaxing the Identityless Assumption, we allow for the inclusion of multiple unique "types" of individuals to be considered. If the number of types of individuals is C, then we will have C equations:

$$\forall i = 1, ..., C; \mathbb{P}_i^{(I)} \pi_i = \pi_i \tag{4}$$

where $\mathbb{P}_i^{(I)}, \pi_i$ refer to the transition matrix and stationary distribution for the ith "type" of unique individual, and are $n \times n$ and $n \times 1$, respectively. Each of these C equations can be solved independently of the others using the aforementioned methods.

5.2 Relaxing the Independence Assumption

With the Independence Assumption dropped, the importance of a single individual's movements is lost – $\mathbb{P}^{(I)}$ becomes useless. We now need information pertaining to the positions of all individuals, and how each

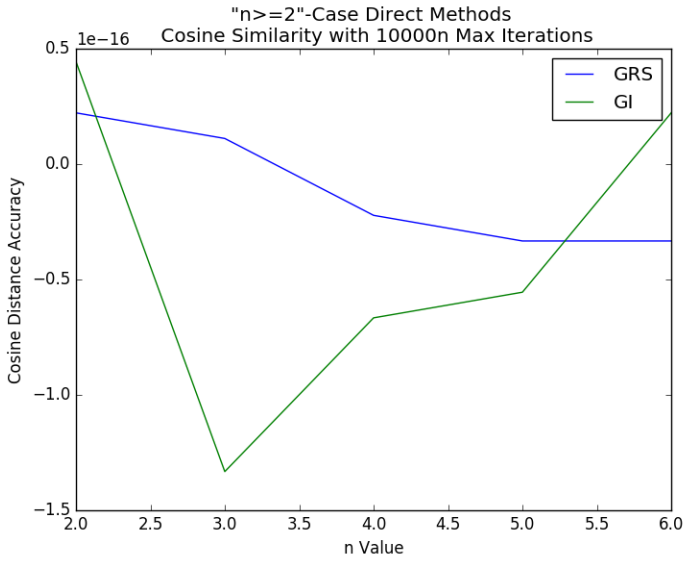


Figure 11: Iteration-Limited Accuracy of GRS and GI Measured by Cosine Distance Metric

of their positions affects the positions of other individuals. This motivates the development of the *Mass Matrix*, $\mathbb{P}^{(M)}$, which allows us to insert information regarding dependencies between individuals or types of individuals. In this paper, when we drop the Independence Assumption, we will primarily be concerned with dynamics that arise from a population identityless individuals; We will tend to not drop Independence Assumption unless we maintain the Identityless Assumption.

6 Finding the Mass Matrix ($\mathbb{P}^{(M)}$)

In this section, we define the *Mass Matrix*, $\mathbb{P}^{(M)}$, motivate its existence, and provide an algorithm one can use to generate a $\mathbb{P}^{(M)}$, namely the Series Method. We provide a description of how the Series Method works along with a quick word on its performance. We also define the notion of *successively-bounded weak compositions* and ways in which we may use $\mathbb{P}^{(M)}$ and the Series Method to relax the Independence Assumption.

6.1 Defining $\mathbb{P}^{(M)}$

In this section, we formally define $\mathbb{P}^{(M)}$ and provide an intuitive meaning of its structure.

Under our usual assumptions, $\mathbb{P}^{(I)}$ is an $n \times n$ transition matrix that describes how each individual hops about bins. On the contrary, $\mathbb{P}^{(M)}$ describes how the entire population hops about different population distributions and is an $s \times s$ transition matrix where:

$$s := \binom{N+n-1}{n-1}$$

We can actually completely determine $\mathbb{P}^{(M)}$ from just two arguments: n and N. Consider a generalized $\mathbb{P}^{(M)}$:

$$\begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1s} \\ q_{21} & q_{22} & \cdots & q_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ q_{s1} & q_{s2} & \cdots & q_{ss} \end{bmatrix}$$

What do each of these q_{ij} values mean? Each is the probability of the entire population changing from distribution j to distribution i in a single time step.

The index corresponding to each distribution is arbitrary but must be consistent. For instance, if $n = 2$ and $N = 3$, index 1 may correspond to the population distribution where 3 people in bin 1 and no people are in bin 2 (let's denote this state as [3,0]). Index 3 may correspond to the population distribution [1,2]. Thus, the element in column 1, row 3 of $\mathbb{P}^{(M)}$ will be the probability that the 3-person population hops to state [1,2] given that the population is currently in state [3,0].

Note that, with $\mathbb{P}^{(M)}$ generated from a feasible $\mathbb{P}^{(I)}$, we can decompose the given π by the following identity:

$$\pi = \sum_{j=1}^{s} v_j \vec{d_j} \tag{5}$$

where v_j is the jth element of the first eigenvector (stationary distribution) of $\mathbb{P}^{(M)}$, v, and $\vec{d_j}$ is the population distribution associated with index j of $\mathbb{P}^{(M)}$. Note that this decomposition of π simply reframes π as the expected value of the long-run population distribution. This identity, in part motivates the generation of $\mathbb{P}^{(M)}$.

6.2 Why do we care about $\mathbb{P}^{(M)}$?

In this section, we discuss why we may seek $\mathbb{P}^{(M)}$.

First, we may use $\mathbb{P}^{(M)}$ to verify our results (the values of $\mathbb{P}^{(I)}$ we obtain from any aforementioned algorithm). If Equation (5) holds, then our proposed $\mathbb{P}^{(I)}$ value is verified.

Second, $\mathbb{P}^{(M)}$ gives us an opportunity to generate new, perhaps more powerful solutions because it allows us to invoke new regularizations. For example, we may choose our objective function in the CMA-ES algorithm[1] to weight the minimization of the second eigenvalue of $\mathbb{P}^{(M)}$ or, additionally, to weight the maximization of the element of the first eigenvector of $\mathbb{P}^{(M)}$ corresponding to π.[2]

Finally, the generation of $\mathbb{P}^{(M)}$ provides a stepping stone one may use toward relaxing the Independence Assumption. We will discuss this in Section 6.3.5.

6.3 Series Method

In this section, we define the Series Method – an algorithm used to generate $\mathbb{P}^{(M)}$ given $\mathbb{P}^{(I)}$. To understand this method, we must first define a weak composition.

[1] This article was born from an undergraduate thesis penned by the same author. That thesis includes a discussion on how another algorithm the CMA-ES algorithm may be used to generate $\mathbb{P}^{(I)}$. The CMA-ES is adept at incorporating regularizations into objective functions and is thus suitable for our purposes. See the link in Section (9) for access to this thesis.

[2] For the purposes of efficiency, it should be noted that symbolic math packages exist (such as SymPy `http://www.sympy.org/en/index.html`) that would greatly facilitate the extraction of information from $\mathbb{P}^{(M)}$. In particular, whenever $\mathbb{P}^{(M)}$ is involved in some algorithm and needs to be recalculated, we know that $\mathbb{P}^{(I)}$ must also be recalculated since $\mathbb{P}^{(M)}$ is a pure function of $\mathbb{P}^{(I)}$. The calculation of $\mathbb{P}^{(M)}$ is expensive relative to $\mathbb{P}^{(I)}$, but with symbolic math packages, the structure of $\mathbb{P}^{(M)}$ need only be calculated once, and values of $\mathbb{P}^{(M)}$ for any new $\mathbb{P}^{(I)}$ can be determined quickly thereafter.

Definition 1 (Weak Composition)**.** *Let* $\texttt{WC}(N, n)$ *denote the set of all possible weak combinations of* N *identical balls and* n *unique bins. A Weak Combination* $w \in \texttt{WC}(N, n)$ *is any* n*-length list of non-negative integers that sums to* N*.*

[Csi]

The rigor in the Series Method comes in determining all the possible ways each individual may be allotted in the next time step given the number of individuals in each previously parsed bin and an upper limit on how many individuals are allowed in each bin. This is similar to any "Bin-Packing Problem" except that, here, we weight every permissible bin packing equally and we seek to return all solutions. The main insight that allows for the Series Method to function is that the many ways individuals from bin i may be allotted to new bins is equivalent to some subset $W_i \subset \texttt{WC}(b[i], n)$ where $b[i]$ is the number of individuals currently in bin i. In particular, $\forall i \in \{1 \dots n\}; W_i = \{w \in \texttt{WC}(b[i], n) | w + \sum_{j=1}^{i-1} k_j \leq T_i, k_j \in W_j\}$ i.e. we only choose weak compositions from W_i that do not cause any bins to "overflow" with more individuals than the vector of upper limits, T_i. Note that w, k_j, and T_i are all vectors. Luckily, we have created an algorithm that can generate all possible permissible weak compositions W_i for all $i \in \{1, \dots, n\}$. The output of this algorithm is a set of *successively-bounded weak compositions.*

6.3.1 Successively-Bounded Weak Compositions

In this section we define the notion of successively-bounded weak compositions.

Definition 2 (Successively-Bounded Weak Composition)**.** *A successively-bounded weak composition,* $L \in \texttt{SBWC}(v, w)$*, with support vector* v *and resistance vector* w *satisfying* $\sum_{i=1}^{n} w_i \geq \sum_{i=1}^{n} v_i$*, is a list* L *of* n*, n-length vectors satisfying:*

$$\forall i = 1, ..., n; \sum_{j=1}^{n} \vec{L}_i[j] = v_i$$

and

$$\sum_{i=1}^{n} L_i \leq w$$

where n *is the length of both* v *and* w*.* $\vec{L}_i[j]$ *represents element* j *of vector* $\vec{L}_i$*. Equality in the second constraint occurs when* $\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} v_i$*.*

Note that the first summation denotes scalar addition whereas the second denotes vector addition.

6.3.2 Description

In this section, an explanation of how the Series Method operates is provided. The Series Method is an implementation of the following series:

$$\mathbb{P}^{(M)}_{s^{(2)}, s^{(1)}} = \sum_{i \in \texttt{SBWC}(s^{(1)}, s^{(2)})} \left[\prod_{k=1}^{n} \binom{s_k^{(1)}}{i} \right] \left[\prod_{t=1}^{n} p_{t,k}^{i_t} \right] \tag{6}$$

where SBWC and WC are defined above, $s^{(l)}$ are vectors with elements $s_k^{(l)}$, and $p_{t,k}$ are elements of $\mathbb{P}^{(I)}$. Furthermore, every i is a vector with elements i_t, and $\binom{s_k^{(1)}}{i}$ is a multinomoial choose operation. We will now explore the intuition empowering this series formula.

6.3.3 Intuition

In this section, an intuitive explanation of why the Series Method operates successfully is provided.

In transitioning from $s^{(1)}$ to $s^{(2)}$, every bin in $s^{(1)}$ may distribute its contents among any of the bins in any permissible way. The one constraint restricting the number of these possible distributions is the *resistance*, $s^{(2)}$ and the cumulative allotment of individuals donated by previously parsed bins. In other words, the total amount of "content" gifted to any particular bin from all bins in $s^{(1)}$, the *support*, must equal the amount of "content" that bin ought to contain in $s^{(2)}$.

The total number of ways of using a particular set of individuals' movements to achieve any particular distribution of individuals $s^{(2)}$ from $s^{(1)}$ is given by $\prod_{k=1}^{n} \binom{s_k^{(1)}}{i}$ for a given successively-bounded weak composition i. The probability of any single one of these movements occurring is given by $\prod_{t=1}^{n} p_{t,k}^{i_t}$, where all $p_{t,k}$ are elements of $\mathbb{P}^{(I)}$.

6.3.4 Real-Time Runtimes

Here, we list the real-time runtimes for the Series Method for various (n, N)-tuple input values in lieu of providing a worst-case runtime complexity. As with all methods, the Series Method was tested on a computer with a single, standard laptop processor with values of $n \in \{2, 3, 4\}$, $N \in \{2, \ldots, 30\}$. The Series Method tends to always take matters of seconds, but it was not tested for very large values of n or N. For larger n or N, the Series Method tend to quickly explode in real-time runtime.

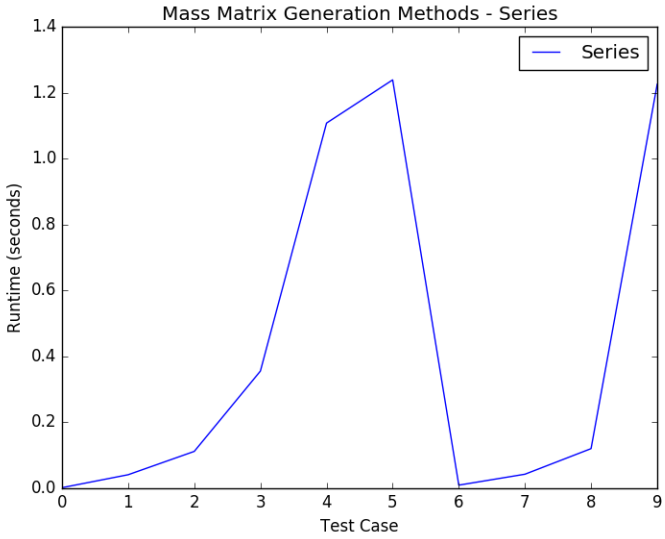


Figure 12: Series Method Real-Time Runtime

The test cases were as follows:

Test Case	n	N	Runtime (s)
0	2	2	0.01140809
1	10	2	0.03999686
2	14	2	0.09832096
3	21	2	0.31426907
4	30	2	0.93162298
5	32	2	1.14047718
6	2	3	0.0084362
7	2	4	0.03489685
8	2	5	0.11156487
9	4	4	1.10382104

6.3.5 Relation to Relaxation of the Independence Assumption

The 3 ways of dealing with the relaxation of the Independence Assumption are outlined below:

1. Mass-First Approach: This is the author's preferred method. One can construct a $\mathbb{P}^{(M)}$ from scratch or generate one given one's *current* $\mathbb{P}^{(I)}$ i.e. from the $\mathbb{P}^{(I)}$ that one's society currently abides by (methods of calculating a current $\mathbb{P}^{(I)}$ are included in Section (7.2)). Dependencies can then be directly encoded in the $\mathbb{P}^{(M)}$. We may then find its stationary distribution (if it exists), sum all possible population distributions together, weighted by the values in $\mathbb{P}^{(M)}$'s stationary distribution. Call that result π, and then find a $\mathbb{P}^{(I)}$ with stationary distribution π using an aforementioned method.

2. Quick-and-Dirty Approach: We may choose to remove certain elements from SBWC, suggesting that only certain distributions of individuals are allowed and the calculated $\mathbb{P}^{(M)}$ can be normalized after the series calculation has touched every element in $\mathbb{P}^{(M)}$.

3. Formal Approach: This process allows for the relaxation of both core assumptions of the direct methods but is bound to live just as theory for the foreseeable future as it would require a massive amount of data to put into practice. Consider again Equations (4), only this time, relax the Independence Assumption, thus allowing $\mathbb{P}_i^{(I)}$ and π_i to be massively larger than before. They now become very-high-dimensional tensors. Let's consider a supposedly simple case, when $n = 2$. Each element of $\mathbb{P}_1^{(I)}$ would be the probability of individual of type 1 moves from one bin to another given the presence of a certain number of type 2 individuals in bin 1, a certain number of type 2 individuals in bin 2, certain number of type 3 individuals in bin 1, etc. Clearly, the relaxation of this assumption quickly becomes unbearable despite being theoretically possible.

7 Applications

Why may we need to obtain a $\mathbb{P}^{(I)}$ from a given π? This section answers that question by example. In general, applications require the following specifications: "Setting, Individuals, Bins, Welfare Function." The "setting" is the context within which individuals move about. The "individuals" are the actors who we categorize into some set of bins. The "bins" are the classes or categories intrinsic to the setting. Moreover, the distribution of individuals among bins is assumed to contribute, in some arbitrary way, to the "welfare function," which we seek to optimize. The determination of π, which is to occur before any algorithm developed in this paper is applied, is assumed to be the ideal distribution of individuals among all bins that optimizes the welfare function. This distribution may be found through *integer programming* or some other optimization procedure.

For all applications, we will continue to hold the Identityless and Independence Assumptions for simplicity's sake, but we are in no way suggesting that these assumptions hold realistic merit and encourage further research to explore their relaxations.

7.1 Company, Employees, Wage, Profit

Consider a company with individual employees distributed among wage brackets. The company seeks to maximize its profit and knows of such a distribution that would maximize its profit. The company seeks to learn how often employees should be promoted in order to converge to this optimal distribution. In other words, a company seeks to converge to a given π and wishes to know what $\mathbb{P}^{(I)}$ spurs that convergence. Any "$n \geq 2$"-case method can be used to find such a $\mathbb{P}^{(I)}$. An additional bin can be added to symbolize both the source of new employees and the sink for retired employees. Constraints can be added to accommodate for recruitment and unavoidable employee churn.

What is the significance of the Identityless and Independence Assumptions in this context? The former implies that all employees experience the same impact of the same company promotion/demotion/churn policies $\mathbb{P}^{(I)}$ whereas the latter implies that an employee's wage bracket is unaffected by the wage bracket of any other employee. This is not necessarily reflective of corporate reality.

7.2 National Economy, Citizens, Rich-Poor, Wealth

This subsection presents an application to sustainable inequality management. The normative models presented by Galor and Zeira (1992) and the empirical evidence collected and corroborated by Kravis (1960), Lydall (1968) and the World Bank (1988, 1989, 1990, 1991) suggest that equity of income is correlated with income per capita. Tabellini (1990) also corroborates this claim but goes further to say that equity is also correlated with the rate of economic growth. [GZ93]

Clearly, there exists an incentive for any nation to distribute income equally among its citizens i.e. for $\forall i \in \{1 \dots n\}; \pi_i = \frac{1}{n}$, and we know that we can apply aforementioned methods to find the optimal set of policies $\mathbb{P}^{(I)}$ that converge to such a π. Furthermore, we may borrow, from Galor and Zeira, the 2-bin classification of individuals: the skilled workers who invest their wealth in human capital and the unskilled workers who largely borrow wealth from the former class. Therefore, $n = 2$. Our problem now is much clearer: Find a $\mathbb{P}^{(I)}$ given $n = 2$ and $\pi^T = [\frac{1}{2}, \frac{1}{2}]$. rS, bS and brS can each be applied to this end.

Of course many solutions for $\mathbb{P}^{(I)}$ exist. To demonstrate these algorithms in action, rS was selected to run six times, and for the input `rS(2, np.array([0.5,0.5]))` the following output was returned:

$$\begin{bmatrix} 0.9724358 & 0.0275642 \\ 0.0275642 & 0.9724358 \end{bmatrix}, \begin{bmatrix} 0.36960558 & 0.63039442 \\ 0.63039442 & 0.36960558 \end{bmatrix}, \begin{bmatrix} 0.4378469 & 0.5621531 \\ 0.5621531 & 0.4378469 \end{bmatrix}$$
$$\begin{bmatrix} 0.65705714 & 0.34294286 \\ 0.34294286 & 0.65705714 \end{bmatrix}, \begin{bmatrix} 0.75512384 & 0.24487616 \\ 0.24487616 & 0.75512384 \end{bmatrix}, \begin{bmatrix} 0.32285924 & 0.67714076 \\ 0.67714076 & 0.32285924 \end{bmatrix}$$

Call this output set A. Clearly A holds many different values for $\mathbb{P}^{(I)}$, and we may prefer some values over another. Perhaps one solutions converges quicker than others (the second eigenvalue, in fact, is closely tied to the speed of convergence [Kal]). but we should choose the value that is most similar to our current $\mathbb{P}^{(I)}$. This will minimize the amount of policy changes that need to be enacted and enforced to prompt society's convergence toward optimal wealth. Three methods are provided below to aid in finding a society's current $\mathbb{P}^{(I)}$ given their current assortment of policies.

7.2.1 Appeal to Regularization

First, one may appeal to any single regularization or combination thereof. For instance, one may prefer a proposed $\mathbb{P}^{(I)}$ with the largest eigengap or trace to maximize the stability of a convergence [HK03]. For situations involving larger $n > 2$, this approach may be impractical, because generating multiple $\mathbb{P}^{(I)}$ may be costly (especially if some property of $\mathbb{P}^{(M)}$ is exploited) and values for $\mathbb{P}^{(I)}$ that maximize some regularization are found through an exhaustive search.

7.2.2 Long-Run Assertion

Alternatively, one may decide to assume their nation already exists in the "long-run," in which case their current distribution of "skilled" and "unskilled" people, as defined earlier, is taken to be π_{old}. Direct methods can then be applied thereafter to generate any number of $\mathbb{P}^{(I)}_{current}$. Given a new, ideal π, new $\mathbb{P}^{(I)}$ can be chosen that most closely resemble perhaps the most different $\mathbb{P}^{(I)}_{current}$ values. However, the society may not actually be in the "long-run" – recently enacted public policies may be forcing the society to tend toward a stationary distribution that is not π_{old}. Thus, it may be erroneous of us to assume that we are currently afflicted by $\mathbb{P}^{(I)}_{current}$.

7.2.3 Estimate Current $\mathbb{P}^{(I)}$

Finally, one may choose to estimate their current $\mathbb{P}^{(I)}$ directly from available, updated census data. For example, we can find an *maximum likelihood estimator (MLE)* for each element in the current $\mathbb{P}^{(I)}$. Fix one time step to be amount of time between any 2 census surveys. Organize the census data such that $X = (x_i)_{i=1}^{D}$ for a population of D people where

$$\forall i \in \{1 \dots D\}; x_i = \left\{ \begin{array}{cc} [1,0,0,0] & \text{if person } i \text{ remained "skilled"} \\ [0,1,0,0] & \text{if person } i \text{ moved from "unskilled" to "skilled"} \\ [0,0,1,0] & \text{if person } i \text{ moved from "skilled" to "unskilled"} \\ [0,0,0,1] & \text{if person } i \text{ remained "unskilled"} \end{array} \right\}$$

In the absence of the Long-Run Assertion, we would have the following optimization problem to solve:

$$\begin{aligned} \theta^* = [p_{11}^*, p_{22}^*] &= \text{argmax}_{\theta \in [0,1]^2} L(\theta|X) = \text{argmax}_{\theta \in [0,1]^2} P(X|\theta) \\ &= \text{argmax}_{\theta \in [0,1]^2} \prod_{i=1}^{D} P(x_i|\theta) \\ &= \text{argmax}_{\theta \in [0,1]^2} \sum_{i=1}^{D} \log(P(x_i|\theta)) \\ &= \text{argmax}_{\theta \in [0,1]^2} \left[\sum_{i=1}^{D_1} \log(p_{11}) + \sum_{i=1}^{D_2} \log(1-p_{22}) + \sum_{i=1}^{D_3} \log(1-p_{11}) + \sum_{i=1}^{D_4} \log(p_{22}) \right] \\ &= \text{argmax}_{\theta \in [0,1]^2} \left[D_1 \log(p_{11}) + D_2 \log(1-p_{22}) + D_3 \log(1-p_{11}) + D_4 \log(p_{22}) \right] \end{aligned}$$

where $D_1 + D_2 + D_3 + D_4 = D$. Taking the gradient of both sides with respect to the elements of $\mathbb{P}^{(I)}$ and setting the entire gradient to $\vec{0}$, the $1 \times 2n$ zero vector:

$$\begin{aligned} \vec{0} = \nabla \log(P(X|\theta)) &= \left[\frac{D_1}{p_{11}} - \frac{D_3}{1-p_{11}}, \frac{D_4}{p_{22}} - \frac{D_2}{1-p_{22}} \right] \\ \Rightarrow D_3 p_{11} &= D_1(1-p_{11}) \text{ and } D_2 p_{22} = D_4(1-p_{22}) \\ \Rightarrow D_1 &= p_{11}(D_1 + D_3) \text{ and } D_4 = p_{22}(D_4 + D_2) \\ \therefore p_{11}^* &= \frac{D_1}{D_1 + D_3} \text{ and } p_{22}^* = \frac{D_4}{D_2 + D_4} \end{aligned}$$

where $L(\theta|X)$ is the likelihood function for parameters θ and data X, D_1 is the number of data samples (people) that remain skilled across any time step, D_2 is the number of people that move from unskilled to

skilled across any time step, D_3 is the number of people that move from skilled to unskilled across any time step, and D_4 is the number of people that remain unskilled across any time step.

If the Long-Run Assertion is used in conjunction with this estimation method, then we would begin just as we did before, by taking the logarithm of the MLE:

$$[p_{11}^*, p_{22}^*] = \text{argmax}_{\theta \in [0,1]^2} L(\theta|X) = \text{argmax}_{\theta \in [0,1]^2} \sum_{i=1}^{D} \log(P(x_i|\theta))$$

However, we need to impose a new constraint in light of the Long-Run Assertion; We need to enforce that we exist in the "long-run" of our current $\mathbb{P}^{(I)}$, which suggests that:

$$\mathtt{M}(\theta)\pi = \pi$$

which is equivalent to:

$$\|\mathtt{M}(\theta)\pi - \pi\|^2 = 0$$

where

$$\mathtt{M} : [a, b] \mapsto \begin{bmatrix} a & (1-b) \\ (1-a) & b \end{bmatrix}$$

is our "current $\mathbb{P}^{(I)}$." So we modify our objective function to now be:

$$f(\theta, \lambda) = \sum_{i=1}^{D} \log(P(x_i|\theta)) - \lambda \|\mathtt{M}(\theta)\pi - \pi\|^2$$

But we also know that both Equation (2) and Constraint (3) must hold whenever a 2×2 transition matrix has stationary distribution π. In other words, we can write our entire $\mathbb{P}^{(I)}$ in terms of a single parameter, $p \coloneqq p_{22}$. Therefore, our optimization problem simplifies to the following form:

$$p^* = \text{argmax}_{p \in [1 - \frac{\pi_1}{\pi_2}, 1]} \left[\log(L(p|X)) - \|\mathtt{M}_0(p, \pi)\pi - \pi\|^2 \right]$$

where

$$\mathtt{M}_0 : [a, \pi] \mapsto \begin{bmatrix} 1 - (1-a)\frac{\pi_2}{\pi_1} & (1-a) \\ (1-a)\frac{\pi_2}{\pi_1} & a \end{bmatrix}$$

Let us discuss why the second term, $-\lambda\|\mathtt{M}_0(p, \pi)\pi - \pi\|^2$, exists as a consequence of the Long-Run Assertion. The matrix $\mathtt{M}_0(p, \pi)$ is the estimate of the current $\mathbb{P}^{(I)}$, and the Long-Run Assertion tells us that our economy of interest already exists in the long-run. Therefore, $\mathtt{M}_0(p, \pi)\pi = \pi$. We want to minimize the extent to which this equality is not true, hence why we penalize the entire objective function whenever the difference between $\mathtt{M}_0(p, \pi)\pi$ and π is nonzero. The difference between vectors is classically given by the L_2-norm. We multiply this difference by $-\lambda$ for some $\lambda > 0$ because, again, we seek to penalize the objective function, which we intend maximize, whenever this difference exists. The λ is the extent to which we care about minimizing this difference.

Notice how the Long-Run Assertion acts simply as a regularization term within our MLE calculation, weighted by λ, penalizing any deviation between $\mathtt{M}_0(p, \pi)\pi$ and π for some p. This type of regularization, utilizing the L_2-norm, is called L_2 *Regularization.*

We may elect to treat this as an application of the method of *Lagrange Multipliers*, with multiplier λ. A necessary condition for this Lagrange Multiplier problem is that the function $f(p, \lambda)$ has a critical point and this implies that $\frac{\partial}{\partial p}\log(L(p|X)) - \lambda\frac{\partial}{\partial p}\|\mathbb{M}_0(p,\pi)\pi - \pi\|^2 = 0$ and $\|\mathbb{M}_0(p,\pi)\pi - \pi\|^2 = 0$. For a crucial reason, we will analyze the latter condition and *then* solve for the former condition.

The latter condition:

$$\|\mathbb{M}(\theta)\pi - \pi\|^2 = (p_{11}\pi_1 + (1-p)\pi_2 - \pi_1)^2 + ((1-p_{11})\pi_1 + p\pi_2 - \pi_2)^2$$
$$\Rightarrow 0 = \|\mathbb{M}(\theta)\pi - \pi\|^2 \Rightarrow (p_{11}\pi_1 + (1-p)\pi_2 - \pi_1)^2 = -((1-p_{11})\pi_1 + p\pi_2 - \pi_2)^2$$

which leaves us with only 1 situation for which there exists a real-valued solution p, namely when:

$$p_{11}\pi_1 + (1-p)\pi_2 - \pi_1 = (1-p_{11})\pi_1 + p\pi_2 - \pi_2$$

we must enforce this, and the importance of this enforcement will soon become crucial.

The former condition and remembering that $p_{11} = 1 - (1-p)\frac{\pi_2}{\pi_1} \Rightarrow \frac{\partial}{\partial p}p_{11} = -\frac{\pi_2}{\pi_1}$, we have:

$$\frac{\partial}{\partial p}f(p,\lambda) = \frac{\partial}{\partial p}\Big[\sum_{i=1}^{D_1}\log(p_{11}) + \sum_{i=1}^{D_2}\log(1-p) + \sum_{i=1}^{D_3}\log(1-p_{11}) + \sum_{i=1}^{D_4}\log(p) - \lambda\|\mathbb{M}(p,\pi)\pi - \pi\|^2\Big]$$
$$= \frac{\partial}{\partial p}\big[D_1\log(p_{11}) + D_2\log(1-p_{22}) + D_3\log(1-p_{11}) + D_4\log(p_{22})$$
$$-\lambda(p_{11}\pi_1 + (1-p)\pi_2 - \pi_1)^2 - \lambda((1-p_{11})\pi_1 + p\pi_2 - \pi_2)^2\big]$$
$$= \frac{-D_1}{p_{11}}\frac{\pi_2}{\pi_1} + \frac{-D_2}{1-p} + \frac{D_3}{1-p_{11}}\frac{\pi_2}{\pi_1} + \frac{D_4}{p} - 2\lambda(p_{11}\pi_1 + (1-p)\pi_2 - \pi_1)(-2\pi_2) - 2\lambda((1-p_{11})\pi_1 + p\pi_2 - \pi_2)(2\pi_2)$$
$$= \frac{-D_1}{p_{11}}\frac{\pi_2}{\pi_1} + \frac{-D_2}{1-p} + \frac{D_3}{1-p_{11}}\frac{\pi_2}{\pi_1} + \frac{D_4}{p} - 4\pi_2\lambda\big(-p_{11}\pi_1 - (1-p)\pi_2 + \pi_1 + (1-p_{11})\pi_1 + p\pi_2 - \pi_2\big)$$

From "The latter condition," we know that the last term is simply 0. We are ready to now set $\frac{\partial}{\partial p}f = 0$ to solve for p^*:

$$\frac{\partial}{\partial p}f(p,\lambda) = 0 = \frac{-D_1}{p_{11}}\frac{\pi_2}{\pi_1} + \frac{-D_2}{1-p} + \frac{D_3}{1-p_{11}}\frac{\pi_2}{\pi_1} + \frac{D_4}{p} = \frac{D_1}{1-p-\frac{\pi_1}{\pi_2}} + \frac{D_3 - D_2}{1-p} + \frac{D_4}{p}$$

We expect there to exist a value for p that solves the above equation for various values of constants D_1, D_2, D_3, D_4, and π. We also expect there to be permutations of constant values for which a solution for p does not exist. Regardless, it may be impossible to write an explicit formula for p^* given the overconstrained nature of the problem – we not only subject p to Constraint (3), but to the Long-Run Assumption as well. The *concavity/convexity* of f is also difficult to determine, so algorithms such as *gradient descent* cannot immediately be used to find p^* (unless a specific domain of p for which f is concave/convex is found). Considering how small the search space $[1 - \frac{\pi_1}{\pi_2}, 1]$ is, an exhaustive search of all possible values of p to some digit precision d may be practical. Such a search would incur a worst-case complexity of $O(\frac{\pi_1}{\pi_2}F(d)10^d)$, where $F(d)$ is the cost of d-digit precision arithmetic.

7.3 Nation, Genomes, Income, Income-per-Capita

This section presents an application to optimal genetic diversity policy as prescribed by Oded Galor. Galor notes that management of the genetic diversity of a nation aims to balance genetic diversity's "negative effect on the cohesiveness of society" with its "positive effect on innovations." Genetic diversity tends to increase mistrust among a citizenry and the amount of civil conflicts, which both tend to lead to inefficiencies in a macroeconomy relative to its *production possibility frontier* (everything that a macroeconomy can produce while operating at its peak efficiency). However, genetic diversity also "fosters innovations expands the production possibilities." Clearly, striking an optimal balance is of concern to those that value their society and can enact relevant policy change. Methods of managing national genetic diversity include the management of immigration and emigration policies and social norms that determine genetic mixing rates. [Gal]

7.3.1 Calculating and Managing Diversity

To calculate genetic diversity, we use the Index of Genetic Diversity from the online appendix to the work of Quamrul Ashraf and Oded Galor [AG], which takes the following form:

$$\mathbb{E}[H^{i,j}] = \frac{\theta_i \mathbb{E}[H^i] + \theta_j \mathbb{E}[H^j]}{1 - F^{i,j}}$$

where N is the total population size, θ_i is the share of people of ethnicity i in a group of people, H^i is the *heterozygosity* (genetic diversity) of ethnicity of i, $H^{i,j}$ is the heterozygosity of a population consisting of $\theta_i N$ member of ethnicity i and $\theta_j N$ members of ethnicity j, $F^{i,j}$ is the *genetic distance* between ethnicities i, j and $i, j \in \{1, \ldots, k\}$ when we consider k ethnicities. Oftentimes, regions and countries consist of more than 2 ethnicities, in which case, the procedure of calculating $\mathbb{E}[H^{i,j}]$ is applied recursively i.e. $\mathbb{E}[H^i] \coloneqq \mathbb{E}[H^j], \theta_i \coloneqq \theta_j, j \coloneqq j + 1$.

Unfortunately, $F^{i,j}$ values only exist for 53 ethnic groups (or pairs thereof), so instead of the above index, parametric estimates of $F^{i,j}$ and $\mathbb{E}[H^i]$ have been formulated. It has been empirically shown that $\mathbb{E}[H^i]$ is strongly, negatively correlated with d_i, the distance between that ethnicity's geographic location in 1 C.E. and East Africa (the alleged birthplace of humanity). Additionally, $F^{i,j}$ has been empirically shown to be strongly, positively correlated with $d_i j$, the *pairwise migratory distance* between ethnicities i and j (the geographic distance between these ethnicities' regions of origin). With this in mind, the following parametric index of genetic diversity is often employed:

$$\mathbb{E}[\hat{H}^{i,j}] = \frac{\theta_i \mathbb{E}[\hat{H}^i(d_i)|d_i] + \theta_j \mathbb{E}[\hat{H}^j(d_j)|d_j]}{1 - \hat{F}^{i,j}(d_j)}$$

If we were to use "country of origin" as a proxy for ethnicity, then i, j would parse over all indices possible countries of origin instead of indices of all ethnicities, and $\hat{H}^i(d_i)$ would be the heterozygosity of country i that is a distance d_i from East Africa.

Oded Galor *et al.* [Gal] already empirically found the "ideal," income-per-capita maximizing value for $\mathbb{E}[H^i]$, which is $\hat{H}^* = 0.2792$. For context, the U.S. is $\hat{H}^{U.S.} = 0.2794$, a tad more diverse than the empirically optimum expected heterozygosity [Gal]. Given $\mathbb{E}[\hat{H}^i]$ and $F^{i,j}, \forall i, j \in \{1, \ldots, k\}$, we can solve for a vector $\vec{\theta}^* = (\theta_1, \ldots, \theta_k)$ that minimizes $|\hat{H}^* - \hat{H}^l|$ for a country l with with immigrants and citizens hailing from a total of k different countries, enumerated as $\{1, \ldots, l, \ldots, k\}$. We can solve for $\vec{\theta}^*$ by discretizing the domain of $\vec{\theta}$ and using *dynamic programming* (also called *memoization*), a method of optimizing recursively-defined functions that stores all potentially optimum values of subproblems in tables. In this case, there are k subproblems, which each solve for a $\theta_i, i \in \{1, \ldots, k\}$. To minimize the search space, we can consider some discrete, monotonic distribution, such as the Zipf Distribution, and vary its support by iterating across all permutations of orderings of the elements of $\vec{\theta}$. The distribution's range would define the values of $\vec{\theta}$ in each

iteration. Further search space reductions arise from not including populations with very small θ_i or with θ_i that cannot be changed without great effort. Alternatively, one may maintain a constant order for the elements of $\vec{\theta}$ and only search among small, discrete perturbations of a subset of θ_i values, holding the others to be constant. Again, finding an ideal $\vec{\theta}^*$, which is related to our ideal π, is beyond the scope of this paper. In practice, upon finding $\vec{\theta}^*$, we will learn to what extent we should adjust our immigration or emigration rates for a particular nationality to better optimize our income-per-capita.

7.3.2 Genomic Diversity Model

We will again use "country of origin" as proxy for genetic diversity, assume we have access to $\mathbb{E}[\hat{H}^i]$ and $F^{i,j}$ for each country i, and assert that there are k countries of origin with citizens found within and outside its borders. Thus, there are 2 bins per country of origin – domestic and abroad. Using $\mathbb{E}[\hat{H}^i]$ values for each nation i instead of each ethnicity allows our model to permit genetic mixing within nations. We end up with multiple $\mathbb{P}_i^{(I)}$ and π_i, in particular one per each nation i. Our model is:

$$\forall i \in \{1, \ldots, k\}$$
$$\mathbb{P}_i^{(I)} = \begin{bmatrix} p_{dd}^{(i)} & p_{ad}^{(i)} \\ p_{da}^{(i)} & p_{aa}^{(i)} \end{bmatrix}$$
$$\pi_i = \begin{bmatrix} \theta_i \\ 1 - \theta_i \end{bmatrix}$$

Note how this situation is identical to the $n = 2$ case of Equation (4), which represents what occurs upon relaxation of the Independence Assumption.

7.4 Generalization of Applications

In this section, we abstractly define the space of all possible applications of the methods discussed in this paper.

Define an *Application Space* to be any ordered pair (n, N, ρ) where $n, N \in \mathbb{N}, n \geq 2, N \geq 1$, and $\rho : U \subset \mathbb{R}^n \to \mathbb{R}_+$, where $\mathbb{R}_+$ are the nonnegative real numbers and U is the set of all elements of $\mathtt{WC}(N, n)$. Let ρ be a *preference metric* – our means of rating all π we could possibly seek given parameters n and N. The more we wish to converge to a particular π, the larger the value of $\rho(\pi)$. More concretely:

$$\forall \vec{x}, \vec{y} \in U; \vec{x} \prec \vec{y} \Rightarrow \rho(\vec{x}) < \rho(\vec{y})$$

where $\vec{x} \prec \vec{y}$ means "we prefer $\vec{y}$ more than $\vec{x}$." Examples of preference metrics may be GDP – a nation's GDP may vary as the distribution of its people among income classes varies, so we prefer population distributions associated with a greater GDP value. If we were to enumerate all of these possible π_i and sum them with weights $\rho(\pi_i)$, we would arrive at the π to which we seek to converge. In other words, if $s = \binom{N+n-1}{n-1}$,

$$\pi = \frac{\sum_{i=1}^{s} \rho(\pi_i)\pi_i}{\sum_{i=1}^{s} \rho(\pi_i)}$$

We can then find a $\mathbb{P}^{(I)}$ that converges to this π using the previously mentioned methods. Note how this summation over all possible π_i is the same as summing over all elements of the first eigenvector of $\mathbb{P}^{(M)}$, as described in Section (6.1).

8 Conclusion

8.1 Summary

In this paper, we worked backwards relative to what is taught in most undergraduate courses concerning Markov Chains; We learned the various ways of solving for a transition matrix given a stationary distribution. After motivating the problem, we described several methods for directly solving for what we called the individual matrix, $\mathbb{P}^{(I)}$, namely: bS, rS, brS, GRS, and GI. Worst-case, expected time, and real-time complexities were provided when available.

Our discussion then drifted toward dealing with constraints on $\mathbb{P}^{(I)}$ and learning what happens when our core assumptions – the Identityless and Independence Assumptions – were relaxed. This discussion transitioned into another discussion concerned with the generation of $\mathbb{P}^{(M)}$, the mass matrix. We learned about the Series Method and its performance. We then had the necessary knowledge to learn more about how we may relax the Independence Assumption, providing 3 approaches: the Mass-First Approach, the Quick-and-Dirty Approach, and the Formal Approach.

We then pivoted into learning about how these methods may be applied to real-world scenarios. Our applications were: "Company, Employees, Wage, Profit," "National Economy, Citizens, Rich-Poor, Wealth" and " Nation, Genomes, Income, Wealth." We provided a clear means of applying the mathematical machinery developed earlier to all application contexts. We concluded our discussion of applications with a generalization of all possible applications.

8.2 Future Research

Future research should concern itself with the optimization of each algorithm's complexity. Parallelization of GRS and the Series Method should be explored. The use of linear programs should also be explored.

With specific regards to the generation of $\mathbb{P}^{(M)}$, connections should be drawn to the field of analytic combinatorics. The literature currently contains plenty of information regarding n-Chamber Ehrenfest Models, for small n, usually $n \in \{2, 3\}$, but what is needed to potentially optimize the runtime complexity of generating $\mathbb{P}^{(M)}$ is research regarding n-Chamber Completely-Connected Ehrenfest Models, where $n > 3$, all n chambers are connected to one another, and no particle is forced to move between chambers between time steps with some probability. Perhaps the runtime of generating $\mathbb{P}^{(M)}$ can be lessened to an even greater extent by abstracting the space of population sizes (values for N) away from solely discrete spaces into the continuum.

9 Supplementary Resources

This articles builds off of an undergraduate thesis, completed by the author in April, 2018, toward an ScB. in Applied Mathematics with Honors degree at Brown University. The thesis was advised by Bjorn Sandstede with Anastasios Matzavinos as the second reader. The submitted thesis as well as all algorithms listed in this paper, which have been coded in Python 2.7.13 by the author, can be accessed on GitHub at:

`https://github.com/kpeluso/thesis/tree/master/Clean`

References

[GG84] S. Geman and D. Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (Nov. 1984), pp. 721–741. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1984.4767596.

[GZ93] Oded Galor and Joseph Zeira. "Income Distribution and Macroeconomics". In: *The Review of Economic Studies* 60.1 (1993), pp. 35–52.

[HK03] Taher H. Haveliwala and Sepandar D. Kamvar. "The Second Eigenvalue of the Google Matrix". In: *Stanford University Technical Report*. 2003.

[Sör07] Kenneth Sörensen. "Distance measures based on the edit distance for permutation-type representations". In: *Journal of Heuristics* 13.1 (2007), pp. 35–47.

[Pag12] Daniel R. Page. "Generalized Algorithm for Restricted Weak Composition Generation". In: *Journal of Mathematical Modelling and Algorithms* 12.4 (2012), pp. 345–372.

[AG] Quamrul Ashraf and Oded Galor. *The "Out of Africa" Hypothesis, Human Genetic Diversity, and Comparative Economic Development*. Available at `https://assets.aeaweb.org/assets/production/articles-attachments/aer/data/feb2013/20100971_app.pdf` (2018/04/10).

[Csi] Péter Csikvári. *composition$_p$artition.pdf*. Available at `http://math.mit.edu/~csikvari/composition_partition.pdf` (2018/04/10).

[Gal] Oded Galor. *Genetic Diversity and Comparative Development*. Available at `http://media.virbcdn.com/files/8a/78fedbd35de3e73a-Galor-Lecture4-2016-H.pdf` (2018/04/10).

[Kal] Sagar Kale. *Eigenvalues and Mixing Time*. Available at `https://math.dartmouth.edu/~pw/math100w13/kale.pdf` (2018/04/10).

[Wol] WolframAlpha. *Output*. Available at `http://www.wolframalpha.com/input/?i=sum+A%2F(A-j)+from+j%3D0+to+j%3DA-1` (2018/04/10).