Degree Paper for the Master's Program in Computational and Applied Mathematics

Haochun Wang

Advisor: Dr. John Reinitz

Approved 

Date 5/9/22

THE UNIVERSITY OF CHICAGO

THE EFFECTS OF MOVE GENERATION DISTRIBUTION ON LAM SIMULATED ANNEALING

A THESIS SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
FOR COMPLETION OF THE DEGREE OF
MASTER OF SCIENCE

COMMITTEE ON COMPUTATIONAL AND APPLIED MATHEMATICS

BY
HAOCHUN WANG

CHICAGO, ILLINOIS
May 5th, 2022

Abstract

The Simulated Annealing (SA) is a nonlinear global optimization algorithm for complex cost functions where other methods do not work. For applications in natural sciences and engineering, the most effective implementation of SA is that of Lam. The Lam algorithm has two key features. One, adaptive cooling, is shared by other SA algorithms. The second and unique feature of the Lam algorithm is move control. The average size of moves is adjusted to keep the acceptance rate close to 0.44, thus maximizing the variance of energy increment. The effects of other features of the distribution used to generate moves have been largely unexamined, but a previous study of Lam SA performance on the Traveling Salesman Problem (TSP) indicated that the choice of move generation distribution could have major effects on computational efficiency. In this work, I extended these investigations to an ODE model used to solve a certain problem in developmental biology. I used a one parameter family of distributions which contain the Normal and Cauchy-Lorentz distributions at special cases. The effect of this parameter is to vary the amount of probability mass in the tails, with the Normal distribution having the least mass in the tails. My major result is that the Normal distribution produces the highest computational efficiency for this test problem. This result is in sharp contrast to a previous study of the TSP.

1 Introduction

The Simulated Annealing (SA) algorithm [8, 12] is a global nonlinear optimization algorithm for complex cost functions where common optimization methods do not work. Under a set of certain constraints, the SA algorithm will yield the global extremum of the cost function [1, 2, 5]. However, even the algorithm is good for its theoretical capability of approximating the global extremum, it has a relatively high computational requirements in practice. To deal with this issue, multiple previous studies have proposed various serial and parallel algorithms that apply different annealing setups [3, 4, 6, 7, 9, 10, 11, 13, 17]. Among them, the most effective implementation of SA is that of Lam [9, 10, 11]. The Lam algorithm has two key features, which are adaptive cooling and move control. However, the effects of other features of the distribution used to generate moves (move generation) have been largely unexamined by Lam. A subsequent study [6] of Lam SA performance on the Traveling Salesman Problem (TSP) indicated that the choice of move generation distribution could have major effects on computational efficiency. In this thesis, I extended these investigations to the gene circuit problem to further discuss the effects of move generation distribution on Lam Simulated Annealing.

In Chapter 2, we are going to introduce more details of the original SA algorithm and the SA algorithm with Lam adaptive schedule [9, 11]. In Chapter 3, we are going to discuss the details of our test problem for SA algorithm – the gene circuit problem [3, 4, 15, 16]. In Chapter 4, we will discuss the experiments and the computational efficiency of different move generation distributions on the gene circuit problem.

2 Simulated Annealing Algorithm

2.1 Original Version of Simulated Annealing Algorithm

The Simulated Annealing (SA) algorithm, proposed by Kirkpatrick [8] and Cerny [13] independently, solves large combinatorial problems with the generalization of Metropolis algorithm [12] by changing the temperature during the algorithm. Specifically, an arbitrary state of a system, denote it s_i, is weighted by its Boltzmann factor, $e^{-E(s_i)/k_BT}$, where $E(s_i)$ is the energy of the state, k_B is the Boltzmann constant and T is the temperature of the system. Then, for a system under thermal equilibrium at temperature T, the probability of the system being in state s_i can be expressed as

$$Pr(E(s_i)) = \frac{1}{Z(T)} e^{-\frac{E(s_i)}{k_B T}} \ ,$$

which is called a Boltzmann distribution. Note here $Z(T)$ is the summation of the Boltzmann factors that serves as the normalizing constant, which is:

$$Z(T) = \sum_i e^{\frac{-E(s_i)}{k_B T}}$$

To simulate the evolution within the system, Metropolis [12] proposed the idea below: given the current state s_i, a subsequent proposal state s_j is generated by applying a small random move from s_i. Now, if $E(s_j)$ is lower than $E(s_i)$, the system "accept" the proposal state and use it as the current state; if the $E(s_j)$ is higher than or equal to $E(s_i)$, the probability of accepting the proposal is $e^{-\frac{\Delta E}{k_B T}}$, where ΔE equals to $E(s_j) - E(s_i)$.

Generalized by Kirkpatrick [8] and Cerny [13], this idea is used to solve complex optimization problems. Specifically, different from the original Metropolis algorithm, Kirkpatrick [8] and Cerny [13] proposed the idea of letting temperature T drop as well. At the beginning of the run, high temperature enables a relatively thorough sampling of the entire search space. As the run reaches to the end, low temperature lets the run converge to the global minimum. The rough structure of the algorithm is below.

Algorithm 2.1 Original Simulated Annealing Algorithm from Kirkpatrick [8] and Cerny [13]

Initialization: initial state s_0, energy $E(s_0)$ and temperature T_0

While Temperature does not decrease to zero:
Step1: generate new state s_{prop} according to the current state s_{curr}
Step2: if $E(s_{prop}) < E(s_{curr})$, update $s_{curr} := s_{prop}$; otherwise update $s_{curr} := s_{prop}$ with probability $e^{-\frac{\Delta E}{T}}$
Step3: lower Temperature T according to a cooling schedule

Return: final energy and state

2.2 Cooling Schedule

2.2.1 Typical Cooling Schedules

In the SA algorithm, cooling schedule for temperature T is important since it directly affects the performance of the algorithm [11, 17]. More specifically, a poor cooling schedule might lead to bad approximation of the global extremum or unacceptable computational cost. There are three most typical cooling schedules for the SA algorithm, which are logarithmic schedule, Cauchy schedule and exponential schedule. The logarithmic schedule is given by $T_k = T_0/$ ln k, where k refers to the iteration step. The Cauchy schedule is given by $T_k = T_0/k$ [17]. For the logarithmic schedule, it has been proved that the SA algorithm will converge to the global extremum if we generate the states according to a Gaussian distribution [5]; for Cauchy schedule, it has also been proved that convergence to the global extremum will be attained if we generate the state from a Cauchy distribution [17].

Even though the asymptotic convergence theories are well constructed for the two schedules mentioned above, it is not practical to apply them in numerical computations due to the high computational cost. In real-life problems, exponential schedule, given by the form of $T_k = T_0 e^{-ck}$, is the baseline of a cooling schedule [3].

Those three typical cooling schedules share two common disadvantages: lack of flexibility and ignorance of the information of cost function during the run. For all three of them, the evolution of temperature only depends on the initial temperature T_0 and do not gather any information about the cost function during the run. To resolve this issue, Lam [9, 11] proposed the adaptive schedule which takes in the information during the run and adjusts the temperature according to the gathered information.

2.2.2 Lam Schedule

In this section, we talk about the outline of the derivation of the Lam schedule [9, 11] along with some necessary details. According to Lam [11], since the SA algorithm is a probabilistic algorithm, the final average energy, rather than a single observation of a final energy for a specific run, is a more important quality for us to analyze and optimize. For the SA algorithm, the evolution of average energy depends on two things: the annealing schedule and the move generation strategy. Considering this, Lam [11] proposed an adaptive schedule following a two-step pipeline:

I. Fix the move generation, minimize the final average energy with respect to the annealing schedule.

II. Fix the annealing schedule brought out in the first step, minimize the final average energy with respect to the move generation strategy.

Now we start to discuss the outline of the derivation. First we introduce the new notation and the key assumptions for the further derivation in this section. In this section, x_i means the energy at state i, s_n denotes the inverse temperature at the n-th step, which is $1/T_n$ and $g_{i,j}$ denotes the probability of proposing a move from state i to state j. Using the new notation here, the probability of a move from state i to state j is accepted is $\min(1, e^{-s(x_j - x_i)})$. At the n-th step during the run, the one-step probability from state i to state j is $P_{i,j}(n) = g_{i,j} \cdot \min(1, e^{-s(x_j - x_i)})$. Now, define $\mathbb{P}(n)$

to be the square matrix that contains elements $P_{i,j}(n)$. Then, we assume the two properties below to facilitate the further derivations:

1. Irreducible: $\forall i, j$, $\exists$ finite k such that $P_{i,j}^{[k]}(n) > 0$, where $P_{i,j}^{[k]}(n)$ denotes the (i,j) entry in the k-step transition probability matrix. From the property of transition matrix, $\mathbb{P}^{[k]}(n) = \mathbb{P}(n+k-1)\mathbb{P}(n+k-2)...\mathbb{P}(n)$.

2. Reversible: $\forall i, j$, $g_{i,j} = g_{j,i}$.

With the these two assumptions, supposing the Markov chain is time-homogeneous, at a fixed temperature T along with $s = \frac{1}{T}$ fixed as well, the equilibrium distribution (alternatively called stationary distribution) is given by $\pi_i(s) = \frac{e^{-sx_i}}{z(s)}$ where $z(s) = \sum_{i \in I} e^{-sx_i}$. Then the average energy of the system is $\mu(s) = \sum_{i \in I} x_i \frac{e^{-sx_i}}{z(s)}$. Note that since $z(s) = \sum_{i \in I} e^{-sx_i}$, we have $\frac{\partial z(s)}{\partial s} = -\sum_{i \in I} x_i e^{-sx_i}$, thus $\mu(s) = -\frac{1}{z(s)} \frac{\partial z(s)}{\partial s}$.

Another thing we care about is the variance, which can be written as $\sigma^2(s) = M_2(s) - \mu^2(s)$ where $M_2(s)$ denotes the second moment. By the notations above, $\sigma^2(s) = \sum_{i \in I} \frac{x_i^2 e^{-sx_i}}{z(s)} - \mu^2(s)$. Here, we obtain the relationship between $\mu(s)$ and $\sigma^2(s)$:

$$\begin{aligned}
-\frac{\partial \mu(s)}{\partial s} &= [\frac{1}{z(s)} \frac{\partial z(s)}{\partial s}]' \\
&= -(\frac{\partial z(s)}{\partial s} \frac{1}{z(s)^2}) \frac{\partial z(s)}{\partial s} + \frac{1}{z(s)} \frac{\partial^2 z(s)}{\partial s^2} \\
&= -\mu^2(s) + \frac{1}{z(s)} \frac{\partial^2 z(s)}{\partial s^2} \\
&= -\mu^2(s) + \sum_{i \in I} \frac{x_i^2 e^{-sx_i}}{z(s)} \\
&= \sigma^2(s). \qquad (1)
\end{aligned}$$

This relationship will be used in the later derivation.

We introduce more notation based on the previously introduced notation. Here, consider the evolution of energy within the system, $X(s_n)$, and note that as n $\to \infty$, the system approaches equilibrium and $X(s_n)$ becomes a stationary process. To specify such stationarity, we denote the process $\underline{X}(s_n)$ along with its mean $\mu(s_n)$ and variance $\sigma^2(s_n)$. Ideally, if we wait for sufficiently long time, the equilibrium will be reached. However, in practice, we need an equilibrium criterion to decide when the system is approximately in equilibrium so that we can lower the temperature. Following this idea, Lam [9, 11] proposed the "quasi-stationary" criterion. The definition goes like this:

A process $X(s_{n-1})$ is called **quasi-stationary** at inverse temperature s_n if it satisfies $|\bar{X}(s_{n-1} - \mu(s_n)| \leq \lambda\sigma(s_n)$, where $\bar{X}(s_{n-1}$ denote the energy at (n-1)-th step and λ is s value specified by us. Generally, the smaller λ is, the better performance with respect to the final energy will be achieved.

Lam [11] further defined λ-schedule. A schedule is called λ-schedule if this schedule leads to a process that satisfies the quasi-stationary criterion at all inverse temperature s within the schedule. Among all the λ-schedules, the schedule that minimizes the final average energy at all steps is called efficient λ-schedule. This seems to be a quite strict condition, but Lam proved that the efficient λ-schedule exists and can be constructed. The construction will be introduced below.

Firstly, we discuss how the average energy of a system evolves. Considering a stationary process, denote it $\underline{X}(s)$. We model this process as a first order autoregressive process (AR(1)), which is:

$\underline{X}_n(s) = \xi + \phi_1\underline{X}_{n-1}(s) + \epsilon_n$, where ϵ_n is the white noise and $\xi = (1 - \phi_1)\mu(s)$.

We write ϕ_1 as r(s) here since it depends on inverse temperature s. Now the model can be rewritten as

$$\begin{aligned}\underline{X}_n(s) &= \xi + \phi_1 \underline{X}_{n-1}(s) + \epsilon_n \\ &= (1 - r(s))\mu(s) + r(s)\underline{X}_{n-1}(s) + \epsilon_n \\ &= r(s)(\underline{X}_{n-1}(s) - \mu(s)) + \mu(s) + \epsilon_n. \end{aligned} \tag{2}$$

This model is considered because of two major reasons. First, this model has the property that the next state only depends on the current state, the move generation and the inverse temperature; second, it assumes the idea that the next energy depends linearly on the current energy. Now, we may use this model to model the evolution of the quasi-stationary process $X(s_{n-1})$ given by

$$X(s_n) = r(s_n)(X(s_{n-1}) - \mu(s_n)) + \mu(s_n) + \epsilon_n.$$

Taking expectations on both sides, we get: $\bar{X}(s_n) = r(s_n)(\bar{X}(s_{n-1}) - \mu(s_n)) + \mu(s_n)$.

From this, we can write an expression of the relationship between $\bar{X}(s_{n-1})$ and $\bar{X}(s_n)$, i.e, the decrement in average energy, given by

$$\begin{aligned}\bar{X}(s_{n-1}) - \bar{X}(s_n) &= \bar{X}(s_{n-1}) - r(s_n)(\bar{X}(s_{n-1}) - \mu(s_n)) - \mu(s_n) \\ &= -(r(s_n) - 1)(\bar{X}(s_{n-1}) - \mu(s_n)) \\ &= (1 - r(s_n))(\bar{X}(s_{n-1}) - \mu(s_n)). \end{aligned} \tag{3}$$

Lam showed [9, 11] that under the two constraints below, we have $\frac{\partial}{\partial s_n}(\bar{X}(s_{n-1}) - \bar{X}(s_n)) \geq 0$.

$$\text{I. } \sigma(s_n) \geq \frac{\lambda}{1-r(s_n)} \left| \frac{\partial r(s_n)}{\partial s_n} \right|,$$

$$\text{II. } \sigma^2(s_n) >> \lambda \left| \frac{\partial \sigma(s_n)}{\partial s_n} \right|.$$

Also, Lam [11] showed that the maximum decrement happens as $\bar{X}(s_{n-1}) - \mu(s_n) = \lambda\sigma(s_n)$ under the quasi-stationary constraint. Under this condition, we can express

$\bar{X}(s_n)$ in terms of $\mu(s_n)$, $r(s_n)$, $\sigma(s_n)$ and λ by

$$\begin{aligned}\bar{X}(s_n) &= \bar{X}(s_{n-1}) - (1 - r(s_n))(\bar{X}(s_{n-1}) - \mu(s_n)) \\ &= \mu(s_n) + \lambda\sigma(s_n) - (1 - r(s_n))\lambda\sigma(s_n) \\ &= \mu(s_n) + \lambda r(s_n)\sigma(s_n). \end{aligned} \tag{4}$$

Using this expression, we get $\bar{X}(s_{n-1}) = \mu(s_{n-1}) + \lambda r(s_{n-1})\sigma(s_{n-1})$. Then we can express the relationship between $\mu(s_n)$ and $\mu(s_{n-1})$.

Because

$$\bar{X}(s_{n-1}) - \bar{X}(s_n) = [\mu(s_{n-1}) + \lambda r(s_{n-1})\sigma(s_{n-1})] - [\mu(s_n) + \lambda r(s_n)\sigma(s_n)] \tag{5}$$

and

$$\bar{X}(s_{n-1}) - \bar{X}(s_n) = \lambda(1 - r(s_n))\sigma(s_n), \tag{6}$$

by moving terms we have

$$\begin{aligned}\mu(s_{n-1}) - \mu(s_n) &= \lambda(1 - r(s_n))\sigma(s_n) - \lambda r(s_{n-1})\sigma(s_{n-1}) + \lambda r(s_n)\sigma(s_n) \\ &= \lambda[\frac{\sigma(s_n)}{\sigma(s_{n-1})} - r(s_{n-1})]\sigma(s_{n-1}). \end{aligned} \tag{7}$$

Based on the current information, we may start to derive the relationship between s_{n-1} and s_n. Now consider the Taylor expansion of $\mu(s_n)$ around s_{n-1}, we have:

$$\mu(s_n) = \mu(s_{n-1}) + (s_n - s_{n-1})\frac{\partial\mu(s_{n-1})}{\partial s_{n-1}} + \frac{(s_n - s_{n-1})^2}{2!}\frac{\partial^2\mu(\hat{s})}{\partial\hat{s}^2} \tag{8}$$

where $\hat{s} \in [s_{n-1}, s_n]$.

By equation (1), we have an alternative relationship between $\mu(s_n)$ and $\mu(s_{n-1})$,

which is given by

$$\mu(s_{n-1}) - \mu(s_n) = -(s_n - s_{n-1})\frac{\partial \mu(s_{n-1})}{\partial s_{n-1}} - \frac{(s_n - s_{n-1})^2}{2!}\frac{\partial^2 \mu(\hat{s})}{\partial \hat{s}^2}$$
$$= (s_n - s_{n-1})\sigma^2(s_{n-1}) + (s_n - s_{n-1})^2\frac{\partial \sigma(\hat{s})}{\partial \hat{s}}\sigma(\hat{s}). \quad (9)$$

We may use the equivalence between (7) and (9) to derive the relationship between s_{n-1} and s_n, which is the annealing schedule under fixed move generation. Now, by Taylor's Theorem again, we have

$$\sigma(s_n) = \sigma(s_{n-1}) + (s_n - s_{n-1})\frac{\partial \sigma(\tilde{s})}{\partial \tilde{s}} \quad (10)$$

where $\tilde{s} \in [s_{n-1}, s_n]$. Therefore,

$$\frac{\partial \sigma(s_n)}{\partial \sigma(s_{n-1})} = 1 + \frac{s_n - s_{n-1}}{\sigma(s_{n-1})}\frac{\partial \sigma(\tilde{s})}{\partial \tilde{s}} \quad (11)$$

where $\tilde{s} \in [c]$.

Now, we have:

$$\lambda[1 + \frac{s_n - s_{n-1}}{\sigma(s_{n-1})}\frac{\partial \sigma(\tilde{s})}{\partial \tilde{s}} - r(s_{n-1})]\sigma(s_{n-1}) = (s_n - s_{n-1})\sigma^2(s_{n-1}) + (s_n - s_{n-1})^2\sigma(\hat{s})\frac{\partial \sigma(\hat{s})}{\partial \hat{s}} \quad (12)$$

where $\tilde{s}$ and $\hat{s} \in [s_{n-1}, s_n]$. By moving terms around, we have:

$$(s_n - s_{n-1})\sigma^2(s_{n-1}) = \lambda[\sigma(s_{n-1}) + (s_n - s_{n-1})\frac{\partial \sigma(\tilde{s})}{\partial(\tilde{s})} - r(s_{n-1})\sigma(s_{n-1})] - (s_n - s_{n-1})^2\sigma(\hat{s})\frac{\partial \sigma(\hat{s})}{\partial \hat{s}} \quad (13)$$

Therefore, we have an relationship between s_n and s_{n-1}:

$$s_n = s_{n-1} + \lambda\frac{1 - r(s_{n-1})}{\sigma(s_{n-1})} + \frac{s_n - s_{n-1}}{\sigma^2(s_{n-1})}[\lambda\frac{\partial \sigma(\tilde{s})}{\partial(\tilde{s})} - (s_n - s_{n-1})\sigma(\hat{s})\frac{\partial \sigma(\hat{s})}{\partial \hat{s}}] \quad (14)$$

In practice, although s_n can be theoretically computed from the implicit expression above if we have explicit expressions for $\sigma(\tilde{s})$ and $\sigma(\hat{s})$, such a computation is hard to carry out. Therefore, we omit the last term to obtain an explicit expression for s_n. Now we have

$$s_n = s_{n-1} + \lambda \frac{1 - r(s_{n-1})}{\sigma(s_{n-1})}. \tag{15}$$

It could be alternatively written as: [11]

$$s_n = s_{n-1} + \lambda \frac{\rho_2(s_{n-1})}{2\sigma^3(s_{n-1})} \tag{16}$$

where

$$\rho_2(s_{n-1}) = \mathbb{E}[(\underline{X}(s_n) - \underline{X}(s_{n-1}))^2]. \tag{17}$$

Here we have finished the first step of the two-step pipeline. In the next section, we will briefly introduce the second step of the pipeline.

2.3 Move Generation

2.3.1 Separable Move Generation

The variance of energy change, ρ_2, is dependent of move generation the final average energy can be minimized by maximizing ρ_2 according to Lam [11]. Thus, the next task is to maximize ρ_2 with respect to the move generation strategy. For this problem, Lam [11] considers two ideas: first, for a specific energy, if there are more states are having this energy, then it is more likely that this energy is proposed; second, the "distance" between the current energy and the proposed energy is a factor that affects the frequency – that is, assuming two energies are at the same distance from the current energy, then they will be proposed with roughly the same frequency.

Lam [11] used a separable model for move generation

$$f_p(x_p|x) \propto G(|x_p - x|)Q(x_p) \tag{18}$$

where x_p is the proposed energy and x is the current energy. In this model, $f_p(x_p|x)$ is the conditional probability density function of the proposed energy, $G(|x_p - x|)$ models the effect of distance between x_p and x on $f_p(x_p|x)$ and $Q(x_p)$ models the effect of energy density function $P(x_p)$ on $f_p(x_p|x)$.

Then define the things below:

I. Let $K(x)$ be the normalizing term for the separable model. That is, $K(x) = \int_{-\infty}^{+\infty} G(|y - x|)Q(y)dy = G(|x|) * Q(x)$, where $*$ denotes convolution.

II. Define $P_s(x) = \frac{P(x)e^{-sx}}{Z_p(s)}$ with $Z_P(s) = \int_{-\infty}^{+\infty} P(x)e^{-sx}dx$. Define $Q(x) = k\frac{P(x)}{K(x)}$ where k makes $Q(x)$ satisfy $\int_{-\infty}^{+\infty} Q(x)dx = 1$. Similarly, let $Q_s(x) = \frac{Q(x)e^{-sx}}{Z_Q(s)}$ where $Z_Q(s)$ is the normalizing term.

III.

$$A_s(x_n - x_{n-1}) = \begin{cases} G(|x_n - x_{n-1}|)e^{-s(x_n - x_{n-1})} & \text{if } x_n \geq x_{n-1} \\ G(|x_n - x_{n-1}|) & \text{otherwise} \end{cases}$$

Using the terms defined above, the n-th moment of the energy increment can be expressed as: [9, 11]

$$\rho_n(s) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} z^n \frac{A_s(z)Q(y)}{K(y - z)} P_s(y - z)dydz \tag{19}$$

where $z = y - x$.

Note that ρ_2 is hard to measure directly and it is "advantageous" [11] to express it using some value that may be measured directly. Acceptance ratio ρ_0, defined as the

probability of accepting a move, can be measured directly. With exponential locality model on G and Gamma model on Q, ρ_2 can be approximated by ρ_0. That is, with

I. $G(|y-x|) = G(|z|) = e^{-\beta|z|}$ where $\beta > 0$.

II. $Q(\beta, x) = \frac{b^N}{\Gamma(N)}(x-x_0)^{N-1}e^{-b(x-x_0)}$, where $x > x_0$,

we can approximate ρ_2 by:

$$\rho_2(s_n) \approx \frac{8\rho_0(1-\rho_0)^2}{s_n^2(2-\rho_0)^2} \tag{20}$$

For G, the exponential locality model is used because it is easy to use and analyze [11]. For Q, the Gamma density model is considered since it has a bounded left tail, which is a good way to model the finite global minimum [4, 11].

Using (20), we have:

$$\begin{aligned} s_{n+1} &= s_n + \lambda\frac{\rho_2(s_n)}{2\sigma^3(s_n)} \\ &= s_n + \lambda\frac{8\rho_0(1-\rho_0)^2}{2s_n^2(2-\rho_0)^2\sigma^3(s_n)} \\ &= s_n + \lambda[\frac{1}{\sigma(s_n)}][\frac{1}{s_n^2\sigma^2(s_n)}][\frac{4\rho_0(1-\rho_0)^2}{(2-\rho_0)^2}]. \end{aligned} \tag{21}$$

Note that to maximize $\rho_2(s)$, $\rho_0 \approx 0.44$. [3, 4, 10, 11] Also, for the expression above, $\frac{1}{\sigma(s_n)}$ measures the distance of the system from quasi-equilibrium and $\frac{4\rho_0(1-\rho_0)^2}{(2-\rho_0)^2}$ measures how effectively the state space is sampled. Here, we have finished the introduction of Lam schedule.

2.3.2 Move Size Control

Note that the schedule above does not depend on a particular problem. Note that the outcome is optimized as $\rho_0 \approx 0.44$, a move generation strategy that allows changes in average move size can be proposed. Note that as the size of move increase, the acceptance ratio will decrease and vice versa.

By Chu [3, 4], the move size can be controlled by $\bar{\theta}^i$, where $\bar{\theta}^i$ is changing periodically so that the acceptance ratio is close to 0.44. Specifically, $\bar{\theta}^i$ is changing following:

$$\ln\bar{\theta}^i_{new} = k(\rho_0 - 0.44) + \ln\bar{\theta}^i_{old} \tag{22}$$

By Chu [3, 4], k is chosen to be 3.0. The moves are drawn from an exponential distribution with mean of $\bar{\theta}^i$. In implementation, this step is done by inversion method. That is, for $X \sim \text{Exp}(\lambda)$ we have $F_X(x) = 1 - e^{-\lambda x}$ where $x \geq 0$. Then for $U \in [0, 1]$, we can find a unique x such that $U = 1 - e^{-\lambda x}$, thus $x = -\frac{1}{\lambda} ln(1 - U)$. That indicates $-\frac{1}{\lambda} ln(1 - U) \sim \text{Exp}(\lambda)$, which is equivalent to $-\frac{1}{\lambda} ln(U) \sim \text{Exp}(\lambda)$ since $U \in [0, 1]$. Therefore, for our case, a move is generated by:

$$x_i^{new} = x_i \pm \bar{\theta}^i \ln(U), U \sim \text{Unif}(0, 1). \tag{23}$$

Note that here the sign is randomly chosen.

Besides exponential distribution, Normal and Cauchy (Lorentz) distribution under the move control are tried in actual computations. Moreover, by Greenwald, Tsallis-Stariolo general visiting distribution is also tried and shows adequate performance on the Travelling Salesman Problem (TSP). In the next section, we will briefly introduce Tsallis-Stariolo distribution and how to sample from this distribution [6].

2.3.3 Tsallis-Stariolo General Visiting Distribution

Under the algorithm setup that we have discussed in the previous sections, the Tsallis-Stariolo general visiting distribution is given by: [6]

$$g_q(x) = \left(\frac{q-1}{\pi}\right)^{1/2} \left(\frac{\Gamma(\frac{1}{q-1})}{\Gamma(\frac{1}{q-1}-\frac{1}{2})}\right) \left(\frac{\frac{1}{\bar{\theta}^{1/(3-q)}}}{(1+(q-1)\frac{x^2}{\bar{\theta}^{2/(3-q)}})^{1/(q-1)}}\right) \tag{24}$$

where $\bar{\theta}$ is the average move size with periodical change as we introduced in the last section. Note that for this distribution, as $q \to 1$, this distribution is Normal distribution.

Next, we discuss how to sample from this distribution. For this distribution, we could not use the inverse transformation to sample from it. This distribution can be sampled using rejection sampling method [6]. Note that the implementation from Greenwald [6] is slightly different from the original rejection sampling method. The difference is Greenwald takes in a function that has finite integral as the comparison function instead of taking in a density function. The algorithm is stated below:

Algorithm 2.2 Rejection Sampling, Greenwald's Implementation [6]

Input: Target distribution f, comparison function b with finite integral A, which is $\int_{-\infty}^{\infty} b(x)dx = A$. Also, $\forall x, b(x) \geq f(x)$ and $\forall y \in [0, A]$, $B^{-1}(y)$ exists.

Step1: Pick $y \in [0, A]$ randomly and get $x = B^{-1}(y)$
Step2: Calculate $b(x)$, pick $y' \in [0, b(x)]$ randomly
Step3: If $y' \leq f(x)$, accept x as a sample; else, reject and return to step1

To guarantee reasonable running time, the comparison function needs to be fitted sufficiently close to the target distribution. Here, consider two separate cases: I. $q \in (1, 2)$ and II. $q \in (2, 3)$.

For case I, $q = 2$ is picked to be used for the comparison function. Note that

$g_q(x)$ in (23) can be written as

$$g_q(x) = \alpha \frac{(\frac{1}{d})}{(1+(q-1)\frac{x^2}{d^2})^{1/(q-1)}} \tag{25}$$

where $d = \bar{\theta}^{(1/(3-q))}$ and $\alpha = (\frac{q-1}{\pi})^{1/2} \frac{\Gamma(\frac{1}{q-1})}{\Gamma(\frac{1}{q-1}-\frac{1}{2})}$. As $q = 2$, we have

$$\begin{aligned} b_1(x) &= \alpha(\frac{\frac{1}{d}}{1+\frac{x^2}{d^2}}) \\ &= \alpha \frac{d}{x^2+d^2} \ . \end{aligned} \tag{26}$$

Here,

$$\alpha = (\frac{1}{\pi})^{1/2}(\frac{1}{\Gamma(\frac{1}{2})}) = \frac{1}{\pi} \ . \tag{27}$$

Therefore, it yields a Cauchy family of function with parameter d.

For case II, $q = 3$ is picked to be used for the comparison function. As $q = 3$, we have

$$\begin{aligned} b_2(x) &= \alpha \frac{\frac{1}{d}}{(1+2\frac{x^2}{d^2})^{1/2}} \\ &= \frac{\alpha}{\sqrt{2}} \frac{1}{(\frac{d^2}{2}+x^2)^{1/2}} \ . \end{aligned} \tag{28}$$

Then we integrate $b_2(x)$, we have:

$$\begin{aligned} \int_{-\infty}^{+\infty} b_2(x)dx &= \frac{\alpha}{\sqrt{2}}[\ln|x + (\frac{d^2}{2}+x^2)^{1/2}|]_{-\infty}^{+\infty} \\ &= \frac{\alpha}{\sqrt{2}} \lim_{t\to\infty}[\ln(t + (\frac{d^2}{2}+t^2)^{1/2}) - \ln(-t + (\frac{d^2}{2}+t^2)^{1/2})] \ . \end{aligned} \tag{29}$$

which is undefined. However, this can be approximated by truncating t at a suffi-

ciently large value in actual implementation [6]. Specifically, we may refer to the approximation that $(1+x)^m \approx 1+mx$ for $x \to 0$. Therefore, for our case, we have

$$(\frac{d^2}{2}+t^2)^{1/2} = t(1+\frac{d^2}{2t^2})^{1/2} \approx t+\frac{d^2}{4t} \quad . \tag{30}$$

Generally, the larger q is, the larger truncation value for t will be needed to guarantee stable results. For example, for $q = 2.5$, we need to have $t \geq 10^6$ and for $q = 2.7$, we need to have $t \geq 10^{12}$. [6]

3 Test Problem

Gene circuit is a set of techniques used for analyzing gene networks [14, 15, 16]. Our test problem is called the gene circuit problem. Specifically, it is an least square problem of numerical and experimental data for gene circuit. A set of ODEs can be utilized to model the concentration of the gene products in cell nuclei with respect to time [3, 4]. The problem here is to compute the values for the parameters in the set of ODEs, which is, minimize the cost function that measures the difference from the results produced by the model and the experimental data.

Before we introduce the details of the problem, we first introduce a set of notations that will be used.

I. $v_i^a(t)$ is the concentration of a-th gene product in nucleus i at time t.

II. T is a matrix of genetic regulatory coefficients with entries T^{ab}. $T^{ab} > 0$ means gene b activates gene a; $T^{ab} < 0$ means gene b represses gene a; $T^{ab} = 0$ means gene b does not have effect on gene a. N is the number of zygotic genes included in the gene circuit.

III. R_a is the maximum rate of synthesis from gene a. v_i^{bcd} is the concentration of Bicoid protein in nucleus i and m^a is the regulatory coefficient of Bicoid acting on zygotic gene a. The product of m^a and v_i^{bcd} is a bias term that arises from Bicoid protein.

IV. h^a is a term that describes the effect of general transcription factors on gene a. g_a is a regulation-expression function. A general form of g_a in terms of T^{ab}, v_i^{bcd}, m^a and h^a is given by Chu. [3, 4]

V. $D^a(n)$ is the diffusion parameter which depends on the number of cell divisions that have occured, given by n. λ_a is the decay rate of the gene product for gene a.

The cost function, E, is given by:

$$E = \sum_{a \in A, i \in I, t, g \in G} (v_i^a(t)_{model} - v_i^a(t)_{data})^2 + penalty \tag{31}$$

where G refers to the set of genotype and penalty applies on limiting the search space. Specifically, if the parameters are going out of the search space, a very large value will be added to the energy. Here, the details about the penalty term is omitted. Those details could be found in Chu. [3, 4]

By optimizing this cost function, we can get the values of parameters for the model given below. Note that this model below is written out with the analogy of chemical kinetic equations.

$$\frac{\partial v_i^a}{\partial t} = R_a g_a (\sum_{b=1}^{N} T^{ab} v_i^b + m^a v_i^{bcd} + h^a) + D^a(n)[(v_{i-1}^a - v_i^a) + (v_{i+1}^a - v_i^a)] - \lambda_a v_i^a \tag{32}$$

For the experiments, we consider two types of problems: two-gene problem and four-gene problem. The two-gene problem is artificially generated by using the data that directly comes from the solutions of a given set of equations. That means, if we get all the parameters recovered, then the energy (for cost function) will be 0. This problem has eight nuclei and 27 discrete time steps. In Chu's experiments [3, 4], there are seven parameters that are annealed on while in our experiments, 11 parameters are annealed on if there is no external gene. Those parameters are T (four entries in T), D (same for both of the genes, so only one parameter is annealed on), m, h and λ^p for both of the genes (λ^p is written to avoid confusion with the λ in the next Chapter). If there is an external gene, seven parameters will be annealed on.

The four-gene problem does not have a ground truth value for the parameters. For this problem, 40 parameters are annealed on.

4 Experiments and Performance Analysis on Gene Circuit Problem

4.1 Detail of Experiments

As we mentioned in the previous section, the experiments are performed on the gene circuit problem. Our two major goals of the experiments are: (I) observing how the two annealing parameters λ, the overall cooling rate and κ, the stopping criterion affect the final energy and the number of iterations for an annealing run to terminate; (II) observe how different move generations affect the final energy and number of iterations. Specifically, the experiments are tried on three problems, which are "hkgn", "g17" and "g9" problem. Among them, the "hkgn" is a four-gene problem and "g17" and "g9" are two-gene problems. As we have discussed in the previous section, the two-gene problem is artificially generated by using the ground truth solution of a particular set of equations as data and the answer is known with global minimum equals to zero. However, the four gene problem has no known global minimum.

During the experiment, we first run the algorithm on the four-gene "hkgn" problem. With the serial setup, the computation cost is relatively high, which takes approximately 160 hours for a single core on Exxact series machine to complete under benign stopping criterion. That means it is not possible to produce batches of results for further analysis in a reasonable period of time.

Then we turn to the "g17" problem. The "g17" problem is a two-gene problem with an external gene. According to implementation by Chu [3, 4], an annealing run terminates when the measured average energy fails to change more than the threshold of the stopping criterion κ for three consecutive times. We first run an initial test with a reasonably small (λ, κ) pair with $\lambda = 0.001$ and $\kappa = 0.001$. The computation finishes in two minutes, which takes much shorter time than the four-gene problem. Because of this, we start to bring out more computations on this problem. However, for smaller (λ, κ) pairs, the computations still take long period of time to complete.

Finally we try the "g9" problem, which is also a two-gene problem but without the external gene. Using the (λ, κ) pair with $\lambda = 0.001$ and $\kappa = 0.001$, the initial test annealing run completes in several seconds. Making both λ and κ smaller, the annealing also completes within an acceptable amount of time. Based on this, we perform further computations and experiments on this problem. The further experiment is carried out in this way: we consider the parameter range for the (λ, κ) pair with $\lambda = 10^{-m}$ where $m \in \{2.0, 2.5, ..., 4.5, 5.0\}$ and $\kappa = 10^{-n}$ where $n \in \{1, 2, ..., 8\}$. Now we have 56 (λ, κ) pairs. For each pair, we generate 100 problems that are with different random initial values for the parameters (not including λ and κ) within a given range. Note that the number 100 is chosen empirically to make sure we have enough samples for further analysis while at the same time the computation cost is not unaffordable. For this step of experiment, we carry out the computations serially with one problem running on one core of the Exxact series machine (with 64 cores) at a single time. After we get the results, which are 100 annealing results for each of the 56 (λ, κ) pairs, we filter out the outliers with significantly high final energies. We will introduce more details about outlier filtering in the next parts.

Below is the plot for the results of experiment using the SA algorithm with Lam Schedule and Exponential move generation, after filtering out the outliers. Note that for other move generations (Normal, Lorentz and Tsallis-Stariolo), the step of experiment mentioned above will produce plots following the similar pattern for the final energy and number of iterations with respect to the λ and κ values. The plot is in log-log scale with x-axis referring to the final energy and y-axis referring to the iterations. The dashed lines refer to the iteration-energy results for different λ values and the solid lines refer to the results for different κ values. The intersection of a solid line and a dashed line represents the results for a (λ, κ) pair. Specifically, the results are the average final energy and the average iterations of the annealing runs for this (λ, κ) pair after taking out the outliers. Note that lower final energy and smaller number of iterations indicate better performance.

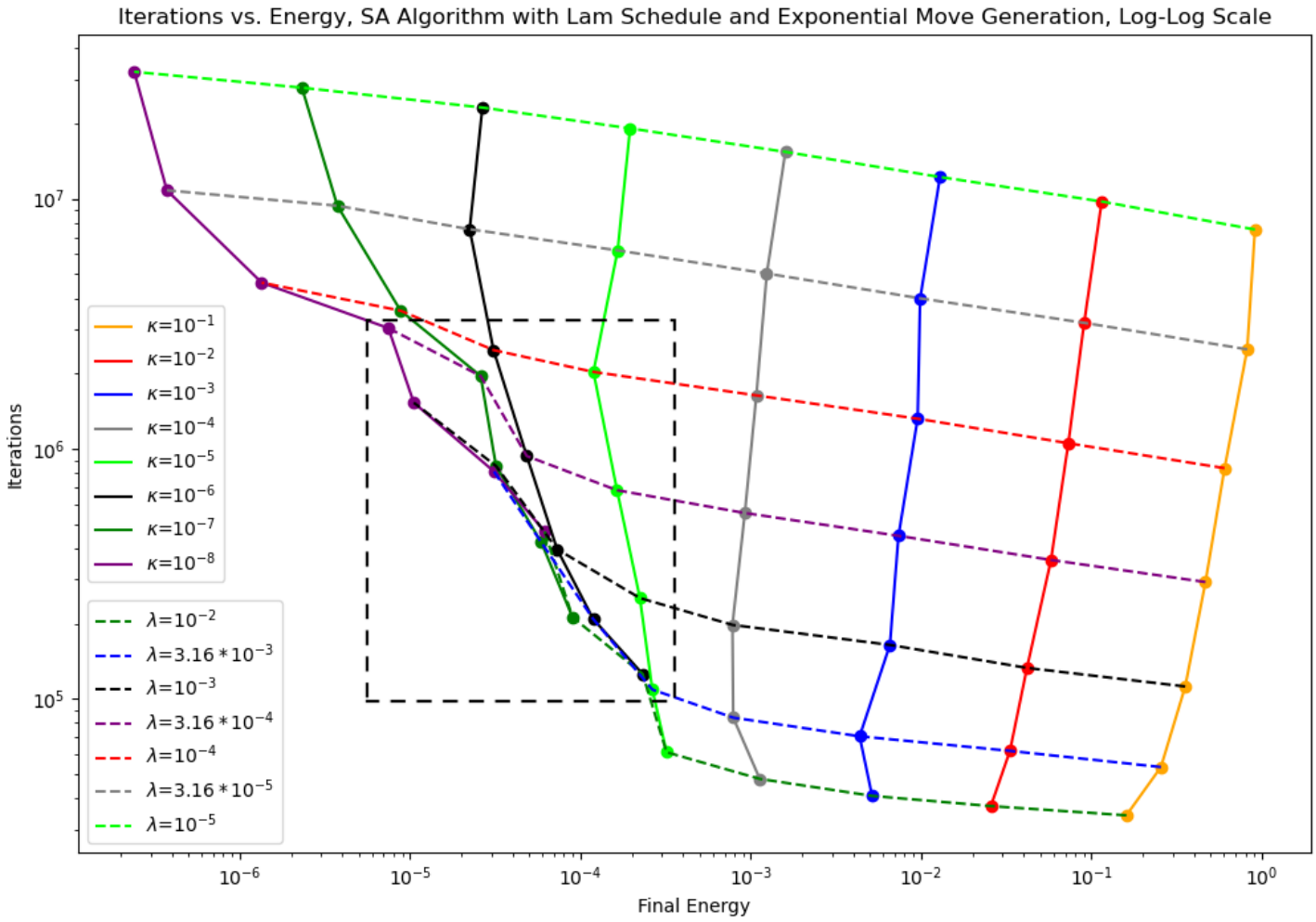


Figure 1: SA algorithm with Lam Schedule and Exponential Move Generation, Log-Log Scale on Iterations and Final Energy, Envelope Specified

From the plot we have some first-step observations:

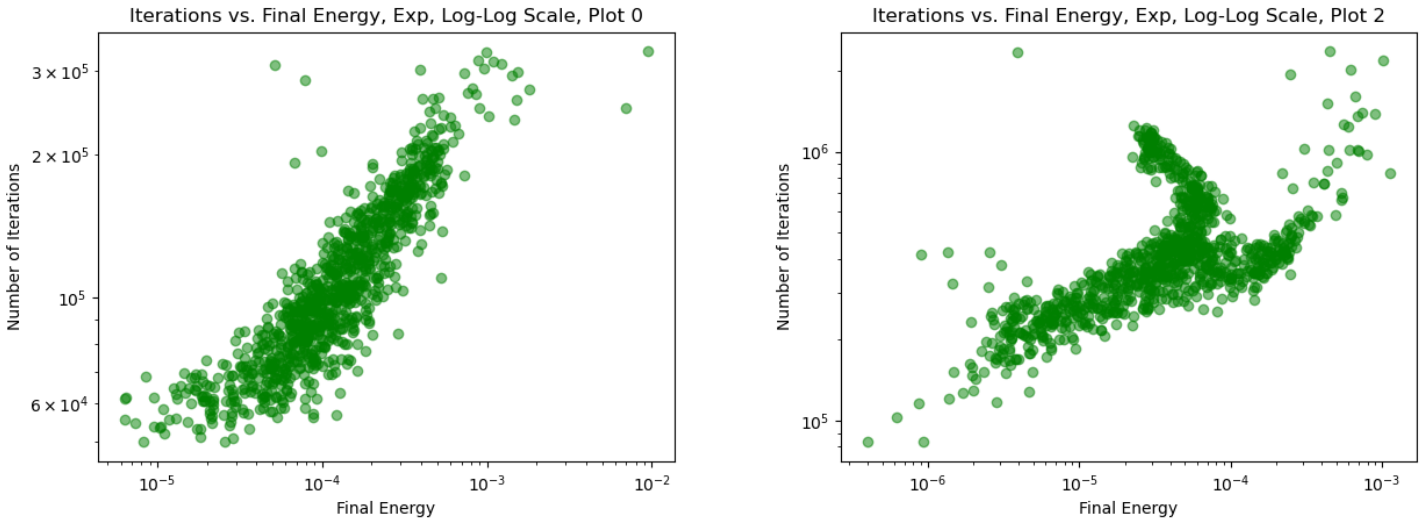
(I): For a fixed λ, as κ is relatively large ($10^{-5} \sim 10^{-1}$), as kappa value decreases, the final energy decreases in a much more significant level compared to the increase of number of iterations. That is, on figure1, the dashed lines on the middle-right part are relatively flat.

(II). For relatively large λ values ($10^{-4} \sim 10^{-2}$), when κ becomes sufficiently small ($\leq 10^{-5}$), we see that the final energy decreases in a less significant level compared to the fact we mentioned in (I) above. On the figure, this phenomenon refers to the middle-left part of the plot.

We should be aware that even though the dashed line is still flat on the left-top corner of the plot, that does not mean we should consider picking small λ and κ at the same time for the annealing run. The reason is that the number of iterations, for small (λ, κ) pair, is not satisfactory. In practice, we care more about the rectangular area (alternatively called "envelope") in figure 1 below. The reason is that it is the area of max efficiency, which means we cannot get lower final average energy with smaller average number of iterations.

Then we do the next step of experiment to further analyze the behaviors for the (λ, κ) pairs within the envelope for all our move generations. Note that within the envelope of figure 1, we do not consider the three light green points on the right and the black point on the middle-top, since for those (λ, κ) pairs, if we further decrease the value for κ (times by 0.1), the decrease of final energy will be much more significant than the increase of number of iterations. For each of the remaining 13 (λ, κ) pairs, we generate 1000 test problems as before for them and perform computations in the same way as the previous step of experiment.

Below are two sample scatter plots of (λ, κ) pairs for Exponential move generation.



(a) (b)

Figure 2: Scatter Plots For Two (λ, κ) Pairs

Now we filter out the outliers. Here, we filter out the points with significantly high final energy. In the experiment, the final energy threshold is set to 0.01 or 0.001. In general, most runs with the annealing runs with significantly high final energy are also taking large number of iterations to complete. However, we do not filter out the runs with final energy within the normal range but with large number of iterations.

After filtering out the outliers, we take the average of the results from the annealing runs for each (λ, κ) pair. Then, for each move generation, we have 13 data points with each point reflects the results of average final energy and average number of iterations after filtering out the outliers according to the final energy threshold. Using the results, we may analyze the serial performance [3, 4] of the (λ, κ) Pairs within the envelope.

4.2 Serial Performance

From the experiment in the previous section, we have 13 data points for each of the move generations, which are Exponential, Normal, Lorentz, Tsallis-Stariolo (with q=2.3, q=2.5 and q=2.7) distribution. The results and plots are below.

For Exponential move generation:

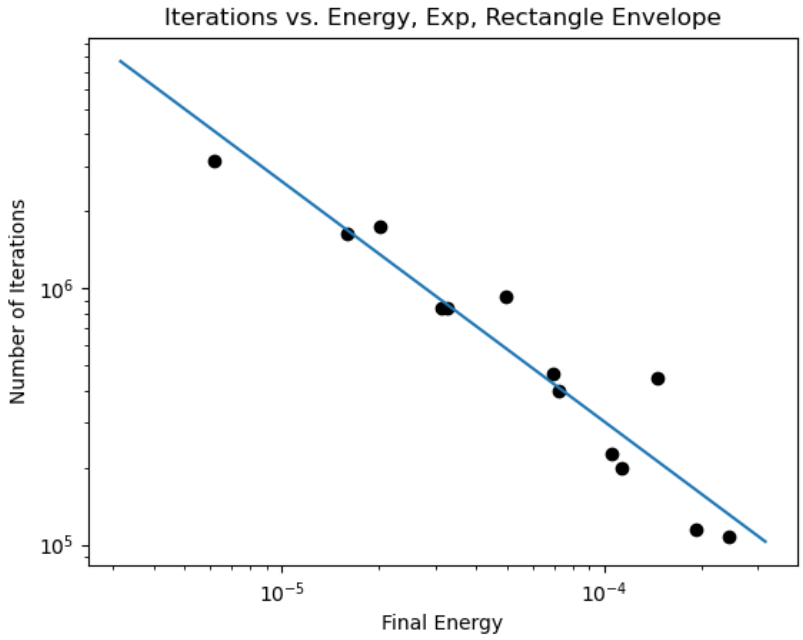


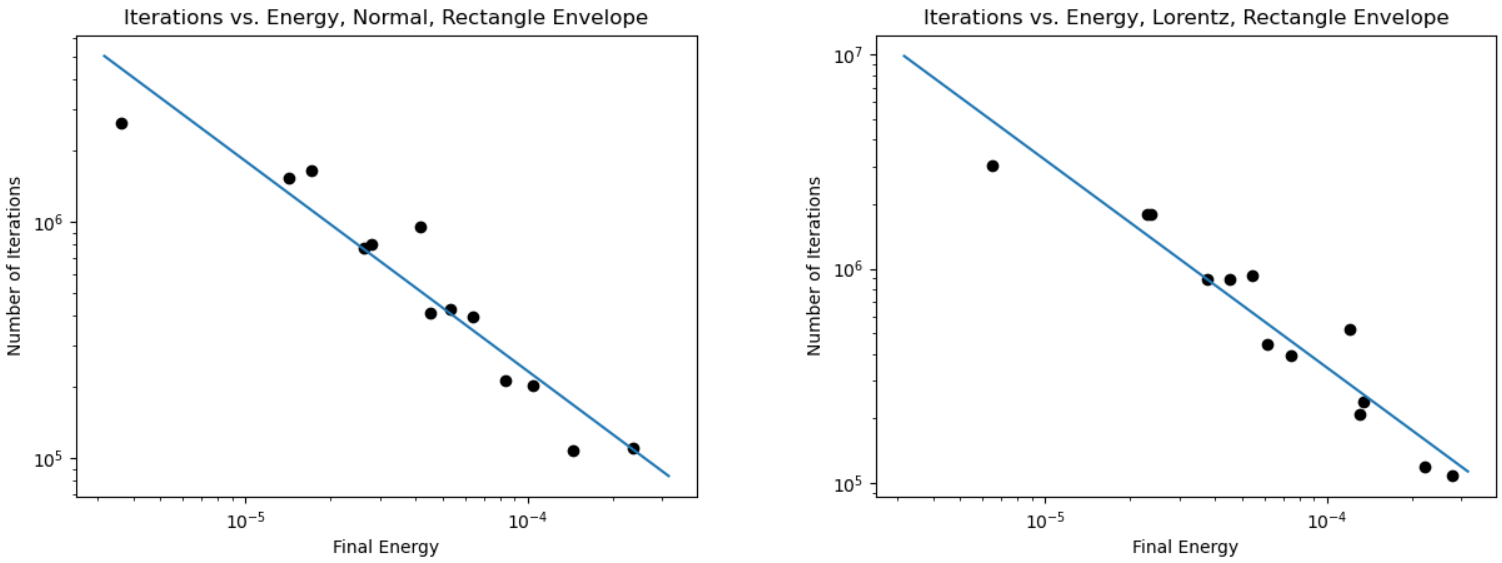
Figure 3: SA algorithm with Lam Schedule and Exponential Move Generation, Log-Log Scale on Iterations and Final Energy, (λ, κ) Pairs within Envelope

We notice that there is a potential linear relationship between log-iteration and log-energy (base 10). Fitting a simple linear regression on log-iteration with log-energy, we have

$$\log_{10} N = 1.73340 - 0.93697 \cdot \log_{10} E \tag{33}$$

where N is the number of iterations and E is the final energy. The p-values for the intercept (β_0) and the coefficient (β_1) are correspondingly 0.00098 and 5.481*10^{-7}, which are both smaller than the 0.05 cutoff so that both of them are significant. The R^2 and adjusted R^2 are correspondingly 0.90611 and 0.89757. From the information above, we may conclude there is linear relationship between log-iteration and log-energy for the results of Exponential Move Generation. On the plot, the blue straight line is the regression line.

Then we look at Normal and Lorentz move generation. The results of Normal and Lorentz move generation are quite similar to the results of Exponential move generation. We also notice the potential linear relationship. Below are the plots for the results of Normal and Lorentz move generation.



(a) (b)

Figure 4: Left: Normal Move Generation, Right: Lorentz Move Generation

Fitting simple linear regression on log-iteration with log-energy for results from Normal move generation, we have

$$\log_{10} N = 1.81004 - 0.88954 \cdot \log_{10} E \tag{34}$$

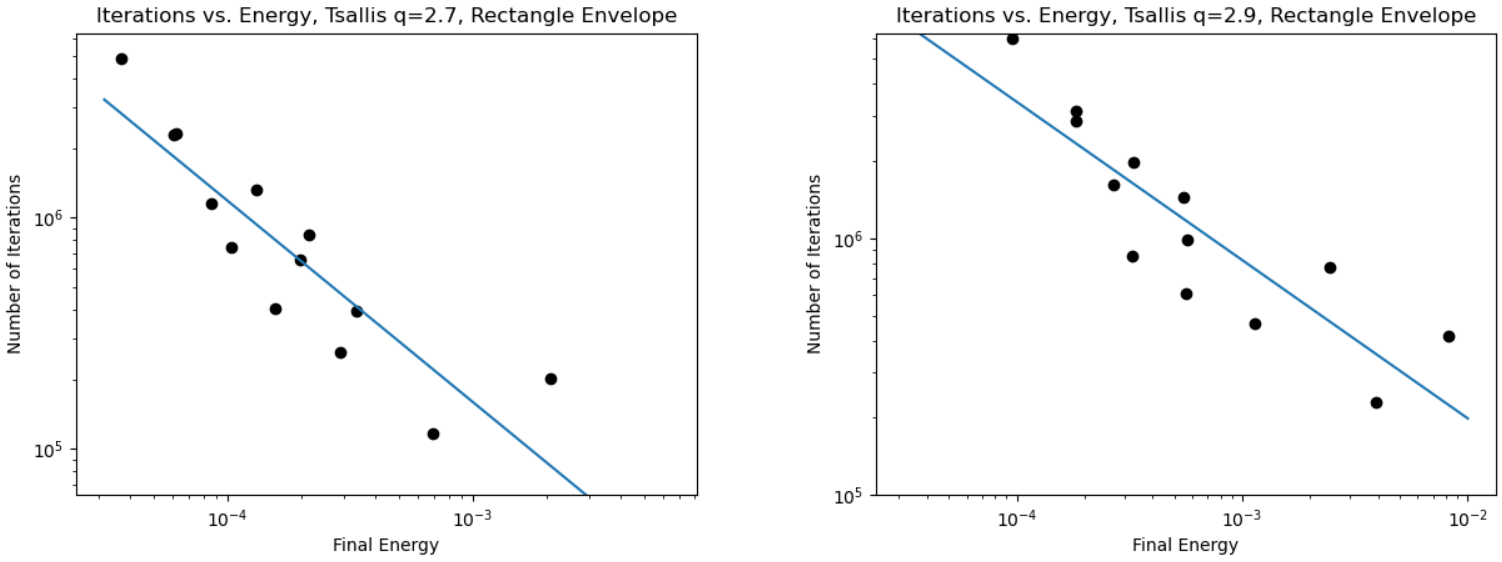
The p-values for the intercept (β_0) and the coefficient (β_1) are correspondingly 0.00057 and 5.3023*10^{-7}, which are both significantly smaller than the 0.05 cutoff. Therefore, both of them are significant. The R^2 and adjusted R^2 are correspondingly 0.90667 and 0.89818. We may conclude the linear relationship exists between log-iteration and log-energy for Normal move generation.

For Lorentz move generation, we have:

$$\log_{10} N = 1.66283 - 0.96898 \cdot \log_{10} E \tag{35}$$

The linear relationship exists between log-iteration and log-energy for Lorentz move generation according to the summary statistics. From here, we omit the reporting for summary statistics considering the repetition.

Next we look at the results of Tsallis-Stariolo move generation with different q values. For q=2.5 and q=2.7, the slope signs are same as the slope signs for the three move generations above and the linear relationship also exist.



(a) (b)

Figure 5: Tsallis-Stariolo Move Generation, Left: q=2.7, Right: q=2.9

However, for q=2.3, the slope sign changes and the final energy becomes significantly larger than the results from other move generations.

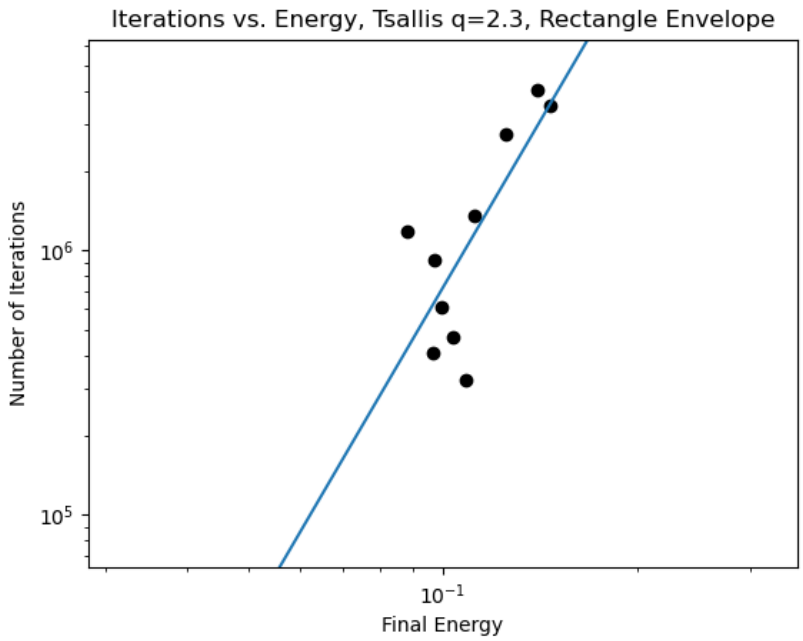


Figure 6: Tsallis-Stariolo Move Generation, q=2.3

Finally, we plot the results for different move generations (except for Tsallis-Stariolo with q=2.3) on the same plot below. From this plot, we see that setting Exponential Move Generation as baseline, only Normal move generation produces satisfactory result within the envelope. Even though Lorentz and Tsallis-Stariolo Move Generation (with q between $2.3 \sim 2.7$) works well for the TSP, they do not perform better than the baseline Exponential move generation for our problem.

Also we can measure the efficiency [6] for the move generation distributions. To measure the efficiency, we set the efficiency of exponential move generation as 1 for a chosen final average energy. Then, the efficiency ratios of other move generation distributions are given by the ratios between the average number of iterations of the Exponential move generation and the average number of iterations of those move generations to achieve the given final energy. Here, with final energy setting to 10^{-5}, we have efficiency ratios of the Normal, Lorentz, Tsallis-Stariolo distribution with q=2.7 and q=2.9 given by, correspondingly, 1.447, 0.814, 0.296 and 0.186.

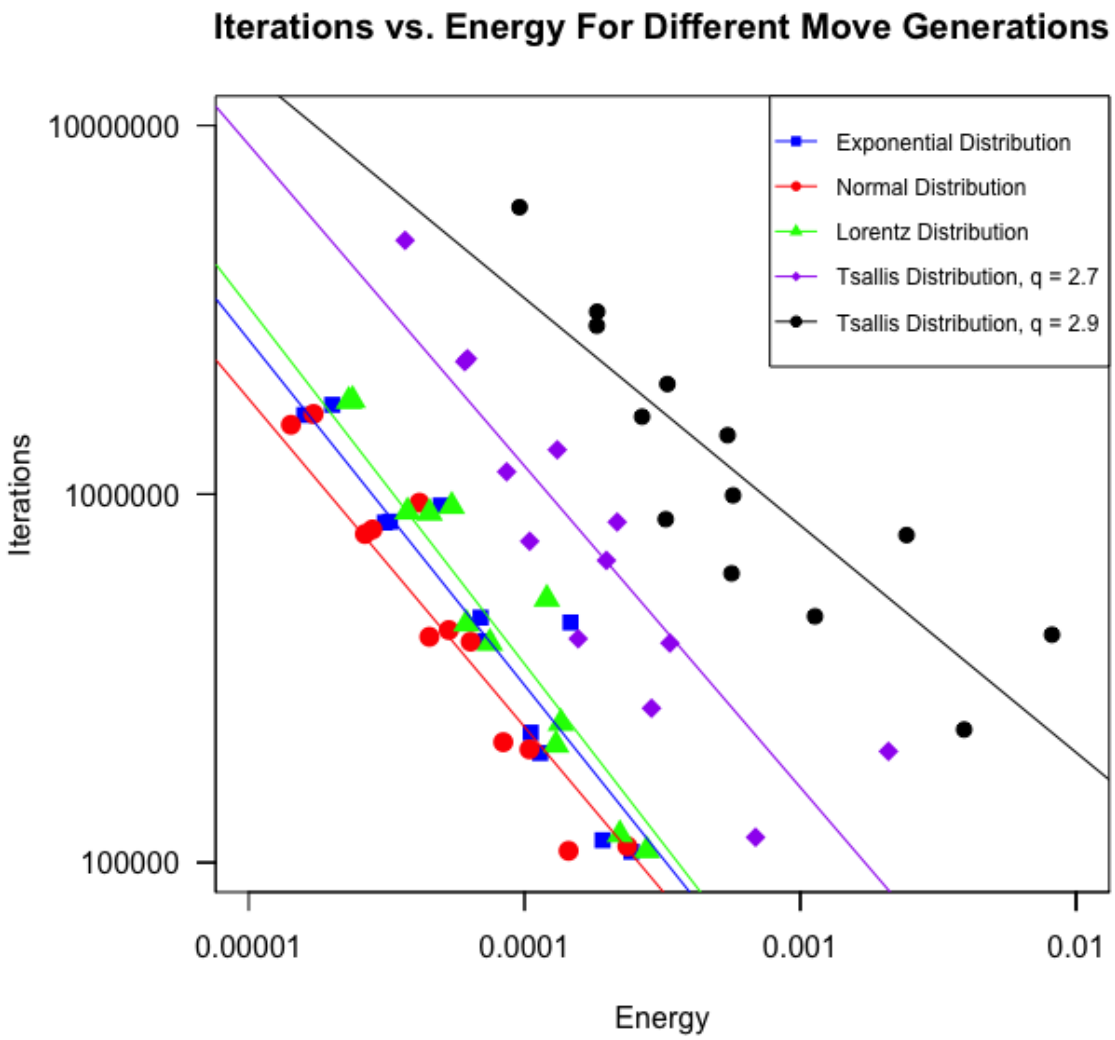


Figure 7: Serial Performance for Different Move Generation, Combined, Log-Log Scale

4.3 Conclusions and Summary

From the experiments in the previous sections, we observe that even though in general fatter tails for the move generation distributions imply higher computational efficiency in the TSP, it is no longer true for the "g9" problem. For the "g9" problem, the move generation distribution with least mass in the tails (among all distributions that we consider), Normal distribution yields the highest computational efficiency.

References

[1] E. H. L. Aarts and P. J. M. van Laarhoven. "Statistical cooling algorithm: A general approach to combinatorial optimization problems". In: *Philips Journal of Research* 40 (1985), pp. 193–226.

[2] R. Azencott. "Sequential simulated annealing: Speed of convergence and acceleration techniques". In: *Simulated Annealing: Parallelization Techniques*. New York: Wiley and Sons, 1992, p. 1.

[3] K. W. Chu, Y. Deng, and J. Reinitz. "Parallel simulated annealing by mixing of states". In: *The Journal of Computational Physics* 148 (1999), pp. 646–662.

[4] King-Wai Chu. "Optimal Parallelization of Simulated Annealing by State Mixing". PhD Thesis, Department of Applied Mathematics and Statistics. Stony Brook University, 2001.

[5] S. Geman and D. Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration Of Images". In: *IEEE Transactions On Pattern Analysis And Machine Intelligence* 6 (1984), pp. 721–741.

[6] L. Greenwald. "Performance of Adaptive Simulated Annealing with Generalized Move Generation Distributions". PhD thesis. Stony Brook, NY: Stony Brook University, 2003.

[7] M. Huang, F. Romeo, and A. Sangiovanni-Vincentelli. "An Efficient General Cooling Schedule for Simulated Annealing". In: *Proceedings of the IEEE International Conference on Computer Aided Design*. 1986, pp. 381–384.

[8] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *Science* 220 (1983), pp. 671–680.

[9] J. Lam and J.-M. Delosme. *An efficient simulated annealing schedule: Derivation*. Tech. rep. 8816. New Haven, CT: Yale Electrical Engineering Department, Sept. 1988.

[10] J. Lam and J.-M. Delosme. *An efficient simulated annealing schedule: Implementation and evaluation*. Tech. rep. 8817. New Haven, CT: Yale Electrical Engineering Department, Sept. 1988.

[11] J. K-C. Lam. "An Efficient Simulated Annealing Schedule". PhD thesis. New Haven, CT: Yale University, 1988.

[12] N. Metropolis et al. "Equation of state calculations by fast computing machines". In: *The Journal of Chemical Physics* 21 (1953), pp. 1087–1092.

[13] W. H. Press et al. *Numerical Recipes in C*. Second. Cambridge, U.K.: Cambridge University Press, 1992.

[14] J. Reinitz, E. Mjolsness, and D. H. Sharp. *Cooperative Control of Positional Information in* Drosophila *by* bicoid *and maternal* hunchback. Tech. rep. LAUR-92-2942. Los Alamos National Laboratory, 1992. URL: `http://sunsite.unc.edu/pub/academic/biology/ecology+evolution/papers/drosophila_theory/Positional_Info.ps`.

[15] J. Reinitz and D. H. Sharp. "Gene Circuits and Their Uses". In: *Integrative Approaches to Molecular Biology*. Ed. by J. Collado, B. Magasanik, and T. Smith. Cambridge, Massachusetts, USA: MIT Press, 1996. Chap. 13, pp. 253–272.

[16] D. H. Sharp and J. Reinitz. "Prediction of mutant expression patterns using gene circuits". In: *Biosystems* 47 (1998), pp. 79–90.

[17] H. Szu and R. Hartley. "Fast Simulated Annealing". In: *Physics Letters A* 122 (1987), pp. 157–162.