# Mcmc- Based Optimization And Application

Xiaobin Wu
*University of Mississippi*

# MCMC-BASED OPTIMIZATION AND APPLICATIONS

A Thesis
presented in partial fulfillment of requirements
for the degree of Master of Science
in the Department of Engineering Science
The University of Mississippi

by

XIAOBIN WU

November 2015

## ABSTRACT

In the thesis, we study the theory of Markov Chain Monte Carlo (MCMC) and its application in statistical optimization.

The MCMC method is a class of evolutionary algorithms for generating samples from given probability distributions. In the thesis, we first focus on the methods of slice sampling and simulated annealing. While slice sampling has a merit to generate samples based on the underlying distribution with adjustable step size, simulated annealing can facilitate samples to jump out of local optima and converge quickly to the global optimum.

With this MCMC method, we then solve two practical optimization problems. The first problem is image transmission over varying channels. Existing work in media transmission generally assumes that channel condition is stationary. However, communication channels are often varying with time in practice. Adaptive design needs frequent feedback for channel updates, which is often impractical due to the complexity and delay. In this application, we design an unequal error protection scheme for image transmission over noisy varying channels based on MCMC. First, the problem cost function is mapped into a multi-variable probability distribution. Then, with the detailed balance", MCMC is used to generate samples from the mapped stationary distribution so that the optimal solution is the one that gives the lowest data distortion. We also show that the final rate allocation designed with this method works better than a conventional design that considers the mean value of the channel.

In the second application, we consider a terminal-location-planning problem for intermodal transportation systems. With a given number of potential locations, it needs to find the most appropriate number of terminals and their locations to provide the economically most efficient operation when multiple service pairs exist simultaneously. The problem also has an inherent issue that for a particular planning, the optimal route paths must be determined for the co-existing service pairs. To solve this NP-hard problem, we design a MCMC-based two-layer method. The lower-layer is an optimal routing design for all service pairs given a particular planning that considers both efficiency and fairness. The upper-layer is finding the optimal planning based on MCMC with the stationary distribution that is mapped from the cost function. The effectiveness of this method is demonstrated through computer simulations and comparison with one state-of-the-art method.

The work of this thesis has shown that a MCMC-method, consisting of both slice sampling and simulated annealing, can be successfully applied to solving practical optimization problems. Particularly, the method has advantages in dealing with high-dimensional problems with large searching spaces.

## DEDICATION

This work is dedicated to my family and parents for all their support and encouragement.

ACKNOWLEDGEMENTS

My best fortune is always that having the advice and guidance of many great people who are so knowledgeable and skillful to enhance my thesis in all ways. First and foremost, I want to offer my gratitude to my advisor, Dr. Lei Cao, for his intelligent support and skillful instruction to my study and thesis. He gave me the inspiration to do my research and master study.

Also, I would like to thank Dr. Paul M. Goggans, for his help, feedbacks and suggestions in my study and thesis. In addition, I want to express my appreciation to my committee member, Dr. Ramanarayanan Viswanathan, for his patience, and time. Finally, I would like to thank department staff, peer students and my family for their support.

TABLE OF CONTENTS

ABSTRACT . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ii

DEDICATION . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . iv

ACKNOWLEDGEMENTS . . . . . . . . . . . . . . . . . . . . . . . . . . . . v

LIST OF FIGURES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . vii

LIST OF TABLES . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ix

MARKOV CHAIN MONTE CARLO METHOD . . . . . . . . . . . . . . . . . 1

APPLICATION OF IMAGE TRANSMISSION . . . . . . . . . . . . . . . . . 19

APPLICATION OF LOCATION PLANNING IN INTERMODAL TRANSPORTATION 39

CONCLUSIONS AND FUTURE RESEARCH . . . . . . . . . . . . . . . . . . 65

BIBLIOGRAPHY . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67

VITA . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72

LIST OF FIGURES

1.1 Uniform random variable $u$ is generated here. It constructs the slice region, which is the grey area. In the grey area, condition $u < P[x(t) = i]$ is followed. 7

1.2 $(I'_{left}, I_{right})$ is the interval enclosing current state $i$ in the beginning. Left end of the interval $P(I'_{left})$ is evaluated and is known to be greater than $u$ (because it is in the grey area, which is slice region). After two times stepping out to left of size $w$ is made, which make left end to $I_{left}$. Check $P(I_{left})$, found $P(I_{left})$ is smaller than $u$, so stop step out. In addition, the right end of the interval $P(I_{right})$ is evaluated and is smaller than $u$, so no stepping out should be done. When process above are done, both ends should be smaller than $u$. Then the interval is hold as $(I_{left}, I_{right})$. . . . . . . . . . . . . . . . . . . . . . . . . . 8

1.3 An interval $I_{left}$ and $I_{right}$ is found for current state $i$. A new state $j$ is drawn from interval, and is rejected because it is outside of slice region. Then the program will shrink the left end to new position $I'_{left} = j$. As we can see, shrinking interval increase acceptation probability of new state. . . . . . . . 9

1.4 $x, y$ are both in the slice. No intermediate point drawn in process. . . . . . . 12

1.5 $x, y$ are both in the slice. With intermediate state $i$ drawn in process. . . . . 13

2.1 A message via channel contains N packets. Each packet could contain diferent length of parity part and data part, but each packet has the same length of bits. 21

2.2 Turbo code performance for fixed coded packet length. Package error rate in different AWGN channel SNR (Cao, 2007a). . . . . . . . . . . . . . . . . . . 23

2.3 MCMC vs Enumeration: Uniform / 10 packets / $SNR = [0, 4]dB$ and fixed $\tau = 1000$. The histograms of all possible distortion based on both MCMC and enumeration are collected. . . . . . . . . . . . . . . . . . . . . . . . . . . 33

2.4 Simulation result with reducing temperature, Uniform distribution, 10 packets. The probability is the histogram of different distortion value divided by number of samples. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34

2.5 PSNR of optimal $\mathcal{R}$: 32 packets / $SNR = [0, 4]dB$ . . . . . . . . . . . . . . . 38

3.1 Intermodal transportation network with consolidation terminals. . . . . . . . 43

3.2 Slice sampling process. $[L, R]$ is the neighbor space enclosing current state $i$. Left end of the interval $P(L)$ is evaluated and is known to be greater than $u$, so a step to the left of size $w$ is made, which make left end to $L'$. Check $P(L')$, found $P(L')$ is smaller than $u$, so stop and step out. Also at the right end of the interval $P(R)$ is evaluated and is smaller than $u$, so no stepping out should be done. When the above process it done, $P$ of both ends should be smaller than $u$. Since the neighbor space is decided as $[L', R]$, then the next state candidate will be picked randomly in the interval $[L', R]$. . . . . . . . . 53

3.3 With the predication factors, the routing paths of service pairs $(1 \to 2)$, $(2 \to 3)$and $(3 \to 1)$. The three services have the same distance and have the same chance to use the potential terminals. Their routing paths are shown in Table 3.3. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

3.4 Without the predication factor. Terminal 6 has not been used. In this case, $(1 \to 2)$ uses all the capacity of terminals $4, 5$. Clearly this is not an efficient way to route the shipping paths. . . . . . . . . . . . . . . . . . . . . . . . . 57

3.5 Map contains service nodes and potential terminals. 1-8 are service nodes and 9-16 are potential terminals . . . . . . . . . . . . . . . . . . . . . . . . . 58

3.6 Paths and terminals after planning. Terminal $12, 13, 14, 15$ have been fully used. $F_k$ depends on terminal capacity. Terminals with large capacity associate with large operational cost. . . . . . . . . . . . . . . . . . . . . . . . . . 60

3.7 Increased 80% of each termial cost $F_k$. Since the high cost $F_k$, the terminal $9, 10, 16$ are closed in order to save operation cost. The terminals $11, 12, 13, 14, 15$ are open and shared by multiple services. . . . . . . . . . . 60

## LIST OF TABLES

2.1 Rates corresponding to numbers. Number will be used in later chapter to represent the rate of each input. . . . . . . . . . . . . . . . . . . . . . . . . . 31

2.2 Table of Minimum: Uniform / 10 packets / $SNR = [0, 4]dB$ . . . . . . . . . . 32

2.3 Frequency from brute force: Uniform / 10 packets . . . . . . . . . . . . . . . . 32

2.4 Table of Minimum: Uniform / 32 packets / $SNR = [0, 4]dB$ . . . . . . . . . . 35

2.5 Performance based on different *bpp* on range of $SNR = [0, 4]dB \sim N(2, 0.5)$ . 36

2.6 Performance based on channel of $SNR = [0, 4]dB \sim N(2, 0.5)$ and channel of Mean $SNR = 2dB$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

2.7 Performance based on channel of $SNR = [0, 4]dB \sim N(2, 1)$ and channel of Mean $SNR = 2dB$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

2.8 Performance based on channel of $SNR = [0, 4]dB \sim Uniform$ and channel of Mean $SNR = 2dB$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

2.9 Cost based on different bpp with $N(2, 1)$ and $SNR = [0, 4]dB$ . . . . . . . . . 38

3.1 Data structure of Table $M_{ij}$ for service pair $\{i, j\}$. . . . . . . . . . . . . . . . 47

3.2 Service pairs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 55

3.3 With the predication factor. Routing path of Fig 3.3. Each service use 25 units on potential terminal. Potential terminals $4, 5, 6$ are fully used. . . . . . 55

3.4 Without predication factor. The routing paths of Fig. 3.4. The first service pair $(1 \rightarrow 2)$ used all capacity of potential terminals 4 and 5. This results in terminal 6 not being used and large costs for other service pairs. . . . . . . . 56

3.5 Service pairs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58

3.6 In the final planning, all potential terminals are all selected except terminal 11. The total cost is 7098, which includes the terminal operational cost 2930 and shipping cost 4168. . . . . . . . . . . . . . . . . . . . . . . . . . . . 59

3.7 Final planning when $F_k$ are increased by 80%. 5 out of 8 terminals are open. The total cost is 9332.8, which includes the operation cost 4374 and transportation cost 4958.8. If we use the planning obtained in table 3.6, the total cost would be 9441, which includes the operation cost 5274 and transportation cost 4168. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

3.8 Terminal status comparison between table 3.6 and 3.7 based on different $F_k$. In the case of increased $F_k$, the number of closed terminals is also increased to avoid extra cost. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

3.9 Three methods are compared in order to verify if using the wheeling function (WF) could result better routing plan. In this case terminals would be share by multiple shipping flows based on predication factors. . . . . . . . . . . . . 61

3.10 Comparison for cases $I16K16$, $I20K20$, each with 10 independent network scenarios. GRASP and MCMC used the same under-layer method presented in the chapter, i.e., table-based heuristic method with prediction factor. The average improvement in percentage of cost is listed in for MCMC-based method. 62

3.11 Comparison for the cases of $I16K16$, $I20K12$, $I28K12$, $I30K20$. Both GRASP-HEP and heuristic MCMC are applied to each map to make comparison. The max, min, and average costs are listed. . . . . . . . . . . . . . . . . . . . . . 63

# CHAPTER 1

# MARKOV CHAIN MONTE CARLO METHOD

In statistics, Markov Chain Monte Carlo is a class of methods for generating samples from probability distributions. By constructing a Markov Chain with a desired distribution and after running the chain for large steps, the states of the Markov Chain can be used as the samples of the stationary probability distribution. Usually, a large initial "burn-in step" should be used to make sure that sample generated by MCMVC matches well with the desired probability distribution. One critical point in MCMC is developing methods to generate the useful samples quickly.

An example of using MCMC is to calculate the area of a specific county in the world map. The shape of a country on map is usually irregular and it is hard to use the coordinates to measure its area. With the Monte Carlo process, we can drop small dots on the map randomly. After a large number of dots painted on the map, we compare the dots inside and outside the country. So that we can find the ratio of the area of the country and that of the whole world. In this procedure, the system reaches its equilibrium distribution after a large number of dots painted. At that moment, the area of the country could be calculated accurately. Apparently, a small number of samples will generate inaccurate result.

There are various of Markov Chain Monte Carlo methods, such as, the Metropolis-Hastings algorithm, Gibbs Sampling algorithm, Slice Sampling and successive over-relaxation. They can all generate samples of the equilibrium distribution after large number of steps running. Some of the methods approach the equilibrium distribution with completely random walk that does not give a specific direction in the parameter space. These MCMC algorithms are easy to construct, such as the Metropolis-Hastings algorithm, but usually take excessively long running time to reach the underlying probability distribution.

To avoid unnecessary steps and slow moving, we use Slice Sampling as our MCMC method (Neal, 2003). The idea of Slice Sampling is that any distribution could be sampled uniformly from the area under the distribution function. The Slice Sampling also uses an area called slice to accelerate the sampling speed and avoid being trapped in local basin areas.

Compared with a popular MCMC method like the Metropolis-Hastings, Slice Sampling is very efficient in speed for two reasons. First, Slice Sampling has strategy of modify the neighbor space of current sample in the whole process; this makes better chance to find a useful sample than the Metropolis method. Second, normal Metropolis method takes a random walk to find next sample, for a new sample only $n$ steps away, it could take about $n$ order of 2 steps to reach, if these steps moved consistently in one dimension.

## 1.1 MCMC Design

In (Ross, 2010), let $x$ be a discrete variable or vector, which has possible value space. Say $\chi_j$ is a possible value of $x$. Then the probability of $x$ could be represent as $P\{x = \chi_j\}$ ( $\chi_j$ could be any value in the space). If we want to calculate the mean value of a function $g(x)$, it is:

$$E[g(x)] = \sum_{j}^{\infty} g(\chi_j) P\{x = \chi_j\} \tag{1.1}$$

Intuitively $g(x)$ could be any function. However if $g(x)$ is very complicated for calculation or input $x$ has extremely large number of possible values, for example, $x$ is a 128-bit-vector of binary numbers. Then the searching space of $x$ is $2^{128}$, which is too large for normal computers to handle. In this case, we should only find an approximate value of $E[g(x)]$. MCMC could be used to generate a large number of samples, such as $x_1, x_2, \ldots$. By the strong law of probability,

$$\lim_{n\to\infty} \sum_{i}^{n} \frac{g(x_i)}{n} = E[g(x)] \tag{1.2}$$

when $n$ is large enough, we could approximately consider equation (1.2) holds.

It can be observed that we can calculate the mean value of a function through samples of the MCMC, and do not need to consider whether the underlying probability distribution function is integrable or not. We avoid the hardest part which is the integration.

The way to construct such a MCMC procedure is not difficult. The following is a brief process of how the Metropolis-Hastings algorithm works. Assume that we have a function $h(\chi_j)$, and the sum is $H = \sum h(\chi_j)$, $j = 1, 2, \ldots$, and $H$is finite. Then for each state $x = \chi_j$ of Markov Chain has the stationary probability $\pi(\chi_j) = h(\chi_j)/H$, and obviously $\sum \pi(\chi_j) = 1$. When current state is $x = \chi_i$, we assume that $q(i, j)$ is the probability of moving $x = \chi_i$ to a new state $x = \chi_j$. With $q(i, j)$ found, a table of probability of transition from any state $\chi_i$ to $\chi_j$ could be accomplished. In addition, we need a probability of acceptance of $x = \chi_j$, namely $\alpha(i, j)$. Then Markov Chain state change probability represents as:

$$P\{i \to j\} = q(i, j)\alpha(i, j), i \neq j \tag{1.3}$$

Since we already have the stationary probability, and the Markov Chain has to be an equilibrium chain, then,

$$P\{i \to j\}\pi(\chi_i) = P\{j \to i\}\pi(\chi_j), i \neq j \tag{1.4}$$

Combined equation (1.3) and (1.4) and rearrange, we can get,

$$\frac{\alpha(i, j)}{\alpha(j, i)} = \frac{\pi(\chi_j)q(j, i)}{\pi(\chi_i)q(i, j)} \tag{1.5}$$

The equation (1.5) is still complicated, because $\alpha(i, j)$ and $\alpha(j, i)$ are depending on each other. Therefore in this situation, the Metropolis choice is given as:

$$\alpha(i,j) = min\left\{\frac{\pi(\chi_j)q(j,i)}{\pi(\chi_i)q(i,j)}, 1\right\} \tag{1.6}$$

Intuitively there are only two options. First, if $\alpha(i,j) = 1$, then:

$$\alpha(j,i) = \frac{\pi(\chi_i)q(i,j)}{\pi(\chi_j)q(j,i)}$$

Second, if we have:

$$\alpha(i,j) = \frac{\pi(\chi_j)q(j,i)}{\pi(\chi_i)q(i,j)}$$

then $\alpha(j,i) = 1$.

Finally, if put $\pi(\chi_j) = h(\chi_j)/H$ into equation (1.6), we can get:

$$\alpha(i,j) = min\left\{\frac{h(\chi_j)q(j,i)}{h(\chi_i)q(i,j)}, 1\right\} \tag{1.7}$$

Briefly the Metropolis-Hastings algorithm is like following:

1. set $\chi_i$ as initial/current state of Markov Chain
2. generate $\chi_j$ as new state by $q(i,j)$
3. use $\alpha(i,j)$ to decide whether $\chi_j$ is accepted or not.
4. if it is not accepted, Markov Chain has no state change. Then go to step 2.
5. if it is accepted, Markov Chan has a state change. The process restarts with $\chi_j$ as current state.

Note that this is the process of the Metropolis-Hastings method. Different MCMC algorithms may have different ways to generate the new samples. However they are all sharing the same idea, i.e., achieving the "detailed balance" represented in equation (1.4).

## 1.2 Slice Sampling

Slice Sampling could be applied to any case that Metropolis method could be used( the Metropolis-Hastings algorithm is considered as the most basic method of MCMC). However Slice Sampling has better performance than the Metropolis methods with more flexible choice in searching parameters.

### 1.2.1 Single variable Slice Sampling

Depending on configuration of parameters, MCMC may have very different performance even for the same problem. The Metropolis-Hastings method is sensitive to the parameter setting, for example, if the step size is too small, it may take very long running time to find an acceptable new sample; otherwise, if step size is too big, program may miss some useful points. Slice Sampling changes the step size automatically to fit into the local shape of the probability distribution so that it can generate samples more quickly. The only requirement of Slice Sampling is that the target density function $P(x(t))$ has to be evaluated at any point $x(t)$ (MacKay, 2003).

Because of the automatically change of step size, Slice Sampling could locate a neighboring space near the current state, and draw samples from it. The idea of neighboring apace is to guarantee that the acceptable state/sample is contained inside. This improvement makes Slice Sampling to have high chance find a new acceptable state.

Compared with the Metropolis-Hastings method, the better performance of slicing sampling (Gilks et al., 2003) is achieved through two more steps: "create interval" and "modify interval", in the searching space. In the process below, they are *step* 3 and *step* 8. Based on these two steps, slice sampling could change region of neighbor space to allocate the next possible state, and accelerate the process of finding a new sample.

Assume at current time $t$, the system $x(t)$ is in state $i$. A brief process of searching for next state $x(t+1) = j$ by Slice Sampling in the one-dimension parameter space is like following:

```
1. evaluate $P[x(t) = i]$
2. draw a vertical coordinate,
       $u \sim Uniform(0, P[x(t) = i])$
3. create a interval enclosing current state $i$,
       $i \in (I_{left}, I_{right})$
4. start loop {
5.     draw $j$ from neighbor space, $j \in (I_{left}, I_{right})$
6.     evaluate $P[x(t+1) = j]$
7.     if $P[x(t+1) = j] \geq u$,
           the next state $x(t+1) = j$ is accepted
           break out loop
8.     else modify the neighbor space and repeat loop
9. }
```

Here $P[x(t) = i]$ is the probability function. $i, j$ represents possible states of system. The time is indexed as $t = \{1, 2, 3, \ldots, \}$. $(I_{left}, I_{right})$ is an interval.

In *step* 1 and 2, program generates a random value, $u$, which is uniformly within range $(0, P[x(t) = i])$. Then the "slice region" would be constructed as $\{x(t) : u < P[x(t) = i]\}$. All values in the "slice" will be the region where next state $x(t+1)$ could be possibly selected. Slice region indicates the vertical region of neighbor space. It should be noted that the region is not necessarily continuous. Figure 1.1 show how this process works:

*step* 3 creates an interval (for one dimension case). The interval, $(I_{left}, I_{right})$ indicates the neighboring space. Any new states should be within the range. Note that the union

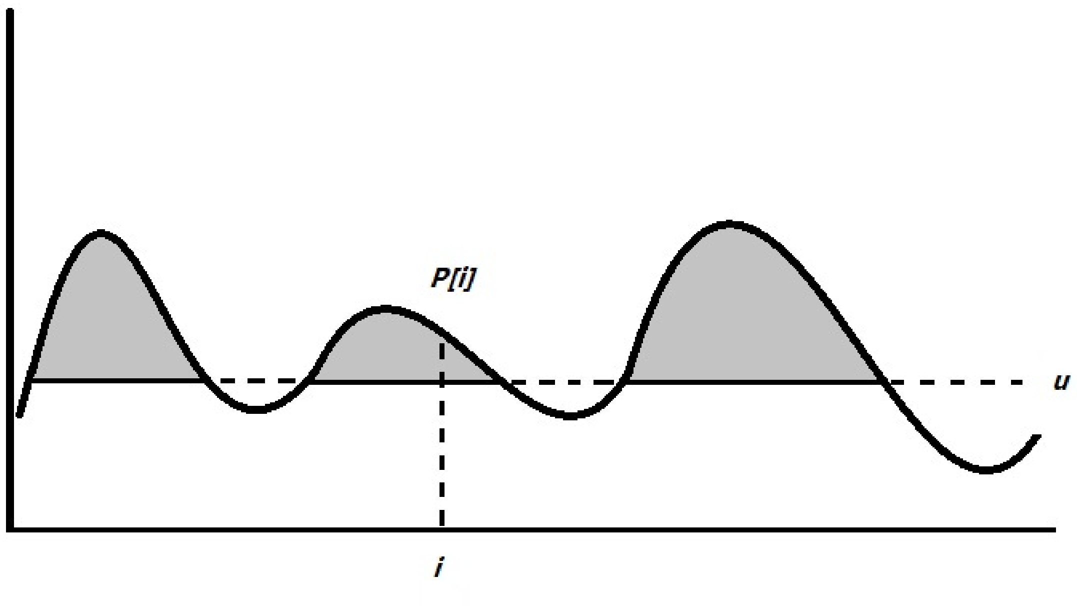

Figure 1.1. Uniform random variable $u$ is generated here. It constructs the slice region, which is the grey area. In the grey area, condition $u < P[x(t) = i]$ is followed.

of interval $(I_{left}, I_{right})$ and slice region is the neighbor space, where $x(t)$ could move to. Interval indicates the horizontal range of neighbor space. More details of *step* 3 is as follows:

1. set a value for the "width", termed as $w$
2. generate $rand \sim Uniform(0, 1)$
3. $I_{left} = i - rand * w$
4. $I_{right} = i + (1 - rand) * w$
5. while $P[I_{right}] > u$, then $I_{right} = I_{right} + w$
6. while $P[I_{left}] > u$, then $I_{left} = I_{left} - w$

Here $w$ is a constant set in the beginning by the user. It is used to represent the step size at each time. The $(I_{left}, I_{right})$ is the interval space of state $i$. In the "creating" process, the Slice Sampling use two *loops* to find the adaptive region of neighbor space. Figure 1.2 show how this process works:

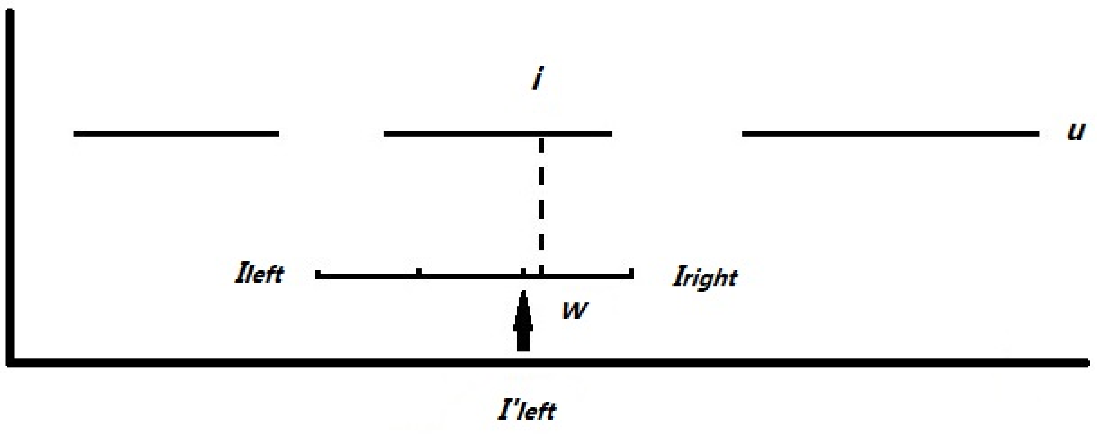

Figure 1.2. $(I'_{left}, I_{right})$ is the interval enclosing current state $i$ in the beginning. Left end of the interval $P(I'_{left})$ is evaluated and is known to be greater than $u$ (because it is in the grey area, which is slice region). After two times stepping out to left of size $w$ is made, which make left end to $I_{left}$. Check $P(I_{left})$, found $P(I_{left})$ is smaller than $u$, so stop step out. In addition, the right end of the interval $P(I_{right})$ is evaluated and is smaller than $u$, so no stepping out should be done. When process above are done, both ends should be smaller than $u$. Then the interval is hold as $(I_{left}, I_{right})$.

As we could see, *step* 3 is actually a process to extend the range of interval. It intends to create a large enough neighbor space for drawing next state. And the extension stops when it first meet condition $P[I_{left}] < u$ in order to avoid decreasing the percentage of acceptable states in neighbor space. Users should always try to use different $w$. An adaptive step size $w$ makes Slice Sampling work better.

It should be noted that *step* 3 above is just one way to setup the interval. There are also many other useful ways to construct the interval. Again, finding a proper way is always important to make Slice Sampling performing well. Some schemes are listed below:

1. We could set interval same as slice region. This also results neighbor space same as slice region. It is easy to understand, but most time it is hard to find infimum and supremum of all grey area (slice region). Apparently we do not need any further modification in *step* 3, since it is fixed range.

2. We first fix $w$. Then every time when extension is needed, we can double the size of

interval. For example, if the left end $I_{left}$ does not fit $P[I_{left}] < u$, then extension step of current interval size will be stepped out. Note that the stepping out direction here is random. As long as stepping out is needed, the interval extends randomly to left or right (Neal, 2003).

In any MCMC algorithm, we know there are always some chances to reject a new state. Slice Sampling method may "recycle" the rejected new state in order to improve the successful chance in next drawing. Strategy used here is to shrink the region again, which is implemented in *step* 8. Whenever a new state $x(t+1) = j$ is rejected, the method modifies the interval.

*step* 8: modifying the interval of one dimension problem:

1. if $j > i$, then $I_{right} = j$
2. else $I_{left} = j$

Here $j$ belongs to neighbor space of $i$, which is constructed in the previous process. Figure 1.3 shows how it works.

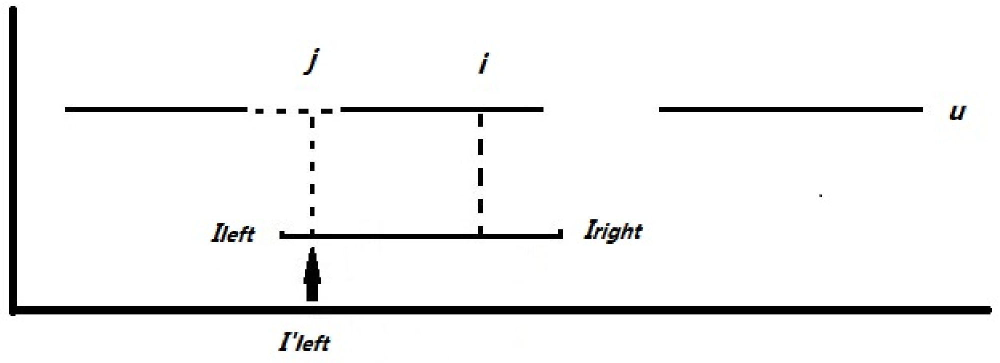

Figure 1.3. An interval $I_{left}$ and $I_{right}$ is found for current state $i$. A new state $j$ is drawn from interval, and is rejected because it is outside of slice region. Then the program will shrink the left end to new position $I'_{left} = j$. As we can see, shrinking interval increase acceptation probability of new state.

The reason to compare $j > i$ is trying to find out their position on the x-axis, and then shrink the neighboring space. Based on the strategy of "creating and modifying interval", Slice Sampling could setup and adjust the neighboring space automatically. Also by defining value of $w$, we could control the speed of step out and shrinking of the neighbor space. In most cases, interval strategy and $w$ are important, particularly for distributions with multiple variables.

### 1.2.2 Multivariate Slice Sampling

Practical problems generally have multiple variables, which means multivariate MCMC method would be needed. In this subsection, we adjust the single variable slice sampling to handle multi-dimensional case. The same strategy could be used to the multivariate case with some proper changes. In the multi-dimension space, values along difference axis need to be handled separately. A common representation of multiple inputs at time $t$ could be written as:

$$x(t) = \{x_1(t), x_2(t), x_3(t), \ldots, x_N(t)\}$$

A summary of multivariate Slice Sampling is as follows:

1. draw random variable, $u$, uniformly on $(0, P[x(t) = \{\chi_1 \ldots \chi_N\}])$.
2. define interval for every $x_i(t) \in x(t)$, say $\{(I_1^{left}, I_1^{right}), (I_2^{left}, I_2^{right}), \ldots, (I_N^{left}, I_N^{right})\}$. These intervals contains each $x_i(t)$ individually. Note that the interval group should be generated and modified independently.
3. find next state $x(t+1)$, if rejection occurs, modify interval group.

Apparently, the most important change in the multivariate Slice Sampling is to deal with the parameter vector $x(t)$. To process, one rule is that only one variable will be selected randomly and updated each time, while all other variables will be fixed.

Let us assume $x_1(t)$ be selected, and we find out its interval $(I_1^{left}, I_1^{right})$. When we are doing step-out of interval of $x_1(t)$, there should be no change made to other variables. By repeating this process for each $x_i(t)$, the interval group $(I^{left}, I^{right})$ could be created.

As long as we found out the interval $(I_1^{left}, I_1^{right})$, we can draw new state $x(t+1)$ from it. Assume $x_i(t+1)$ is chosen, the process is like following:

1. draw each $x_i(t+1)$ based on interval $(I_i^{left}, I_i^{right})$, and put into $x(t+1)$.
2. if $u < P[x(t+1)]$ then the new state accepted.
3. otherwise modify interval $(I_i^{left}, I_i^{right})$ for each $x_i(t+1)$.

If the new state is rejected, then shrinking interval should be done accordingly and modified individually. Since $x(t)$ is a vector, an interesting idea is considering each variable as a direction element of the state $x(t)$. Each variable has its own interval. By drawing and combining all variable, we know which way the state should move.

### 1.2.3 Detailed Balance Slice Sampling

In this section, we discuss the detailed balance of Slice Sampling, which means the probability of choosing a new state $y$ given the current state $x$ s the same as probability of choosing a new state $x$ give current state $y$, and this is true for any states $x, y$. In order to prove detailed balance, $x, y$ must be both in the same slice area, because new sample is generated uniformly on neighbor space. If both $x, y$ have the same neighbor space, we can say they have the same probability to be chosen. In addition, we can separate detailed balance into two cases: first, case without intermediate state chosen; second, case with intermediate state chosen.

We consider the case without intermediate point chosen first. In such case, $y$ is drawn in the union of the slice region and the interval, which means it is guaranteed to be accepted. Therefore Slice Sampling does not use $y$ as an intermediate state to shrink the interval. Figure 1.4 is as below:

Let's first assume the current state is $x$. In the step-out process, a uniform variable $rand_x$ is generated to define the initial position to hold point $x$ within range size $w$. Then Slice Sampling extends initial interval until both $I_{left}$ and $I_{right}$ are outside of the slice region.

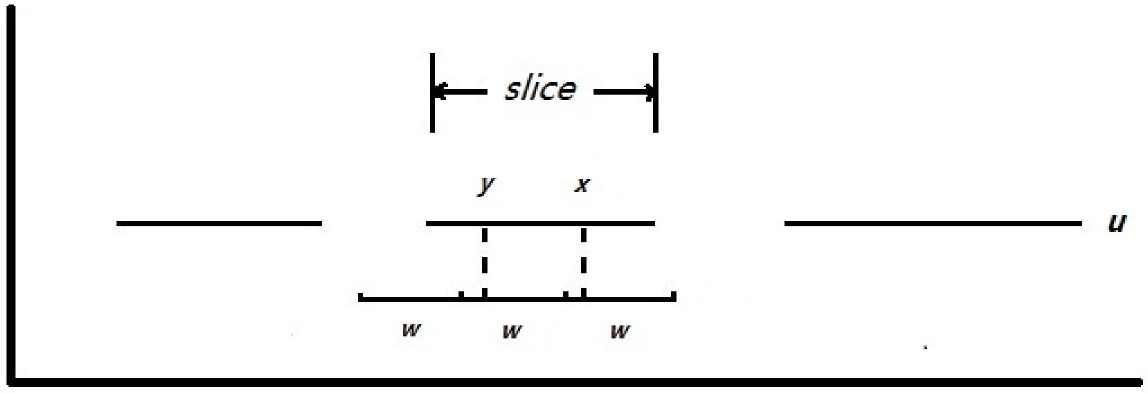

Figure 1.4. $x, y$ are both in the slice. No intermediate point drawn in process.

Then we can construct a variable $rand_y$ corresponding to $rand_x$, and define it as

$$rand_y = Frac(rand_x + (y - x)/w)$$

$Frac$ is the function that outputs the fraction part of its input. Intuitively, after two times extension of the left end $I^x_{left}$, the interval size is $3w$, which is shown in Figure 1.4 above.

Now we set the start/current state as $y$, and use $rand_y$ to indicate initial position of $y$ within $w$. We find both $I^y_{left}$ and $I^y_{right}$ need to step-out to meet the condition that both $I_{left}$ and $I_{right}$ must be outside of the slice region. After the construction of interval for $y$, we can find it has same range as interval of $x$. This means, by knowing $rand_x$ and $rand_y$, $x$ or $y$ have the same interval. Therefore, we proved Slice Sampling without intermediate state holds the detailed balance.

The second case is Slice Sampling with an intermediate state drawn. Figure 1.5 is as below:

In this case, state $i$ is outside of the slice region, and $x, y$ are in slice region. $i$ is the intermediate state between $x, y$.

Assume that $x$ is the start/current state. Then interval $\{I_{left}, I_{right}\}$ can be found with width $3w$ in Figure 1.5. Then $i$ is drawn with probability density $1/3w$. Since $i$ is

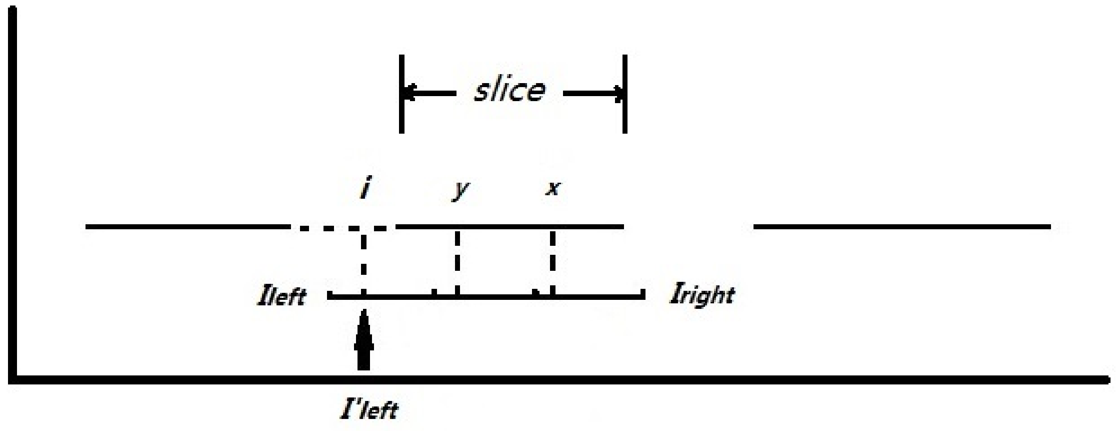

Figure 1.5. $x, y$ are both in the slice. With intermediate state $i$ drawn in process.

outside of the slice region, Slice Sampling shrinks the interval to $\{I_{left} = i, I_{right}\}$. Then $y$ is drawn in $\{I_{left} = i, I_{right}\}$.

Assume that $y$ is the start/current state. Then Slice Sampling extends both ends, and interval is $\{I_{left}, I_{right}\}$. The intermediate state $i$ is drawn with probability density $1/3w$. Then Slice Sampling shrinks $I_{left}$ to $i$. So $x$ can be drawn from $\{I_{left} = i, I_{right}\}$.

Therefore, with intermediate state $i$, $x, y$ hold the detailed balance. Because they have the same interval $\{i, I_{right}\}$, which means they have same neighbor space. Also we can prove from induction this is true for the case with $n + 1$ intermediate states by proving the case with $n$ intermediate states.

$$P(new = x; |current = y; intermediate = i) = P(new = y; |intermediate = i; current = x)$$

We can say Slice Sampling holds the detailed balance.

### 1.2.4 Convergence of Slice Sampling

In this section, we discuss the complexity of Slice Sampling. Many MCMC algorithm use the "rejection-acceptation method to generate samples. In this case, these algorithms need a compatible probability density function. For an example of the Metropolis-Hastings

Algorithm, assume $x$ is from density function $f(x)$. Then transmission distribution function is defined by user, assume $q(x, y)$. For the Metropolis-Hastings algorithm, a new value $y$ is drawn from distribution function $q(x, y)$. Then the acceptance variable $\alpha(x, y)$ is calculated. Based on $\alpha$, we will decide whether $y$ is accepted as new sample.

In this process, the most important thing is the transmission distribution, i.e., the proposal distribution. A high compatible distribution may greatly decrease the chance of rejection. Apparently looking for a compatible proposal distribution is always not easy. Sometimes any minor change could make the target density function difficult to find, such as, drawing samples from any distribution with lots of interval region of $x$. Such a distribution only has values, when $x \in (2n-1, 2n)$, $n = 1, 2, 3, 4$. It is hard to find an efficient transmission distribution for such a distribution.

However for Slice Sampling, the above case becomes extremely easy, since proposal distribution is not important in Slice Sampling. All Slice Sampling does is to compare variables drawn from uniform distribution. Slice Sampling still has to deal with the region problem above. However it is just a uniform distribution, which makes the process much easier. This is why Slice Sampling is easy to use than the Metropolis-Hastings method.

Also Slice Sampling has been proved to have better convergence performance than Independence the Metropolis Hastings Algorithm (IMHA) in many cases (Mira and Tierney, 2002). In order to explain that, we assume a mass density $f(x)$, which is our target density function.

$$f(x) \propto p(x)Q(x)$$

The $Q(x)$ is always not less than 0, which is also considered as likelihood probability, and $p(x)$ is prior probability. In order to construct a joint distribution of auxiliary variable $u$ and $x$, we could set up the region of $0 < u < Q(x)$. Then the equation will be:

$$f(x, u) \propto p(x)I_{0<u<Q(x)}(x) \tag{1.8}$$

$I_{0<u<Q(x)}(x)$ indicates the region of $u$, when current state $x$ is given. General case of Markov chain (discrete case) constructs transition kernel as following:

$$K(\chi, Y) = Prob(x_{i+1} = y \in Y | x_i = \chi) = Prob(Y|x_i) \tag{1.9}$$

The $K(\chi, Y)$ is the transition kernel of slice sampling. Given the current state $x_i$, transition kernel gives the probability of new state $Y$. Apparently the new state $y$ is always dependent on the current state, but not relate to the sequence of former states $\{x_1, x_2, x_3, \}$. For notation, we write kernel as $K(x, Y)$ to indicate the transition probability from $x$ to $Y$. If an invariant probability distribution $f(x)$ and $K$ exist, then they must follow $f(Y) = \int K(x, Y)f(x)dx$. Since we know the relationship among $x, u, y$ in Slice Sampling, then we write the transition kernel [1.9] as following:

$$\begin{aligned} K(x, Y) &= \int_Y \int Prob(y|u)Prob(u|x)dudy \\ &= \int_{u=0}^{Q(x)} Prob(u|x) \int_Y Prob(y|u)dydu \\ &= \int_{u=0}^{Q(x)} \frac{1}{Q(x)} \int_Y p(y|A_u)dudy \end{aligned}$$

Since $u$ is uniformly on range $(0, Q(x))$, probability of u given x is $1/Q(x)$. $A_u$ here indicates $\{x : 0 < u < Q(x)\}$. Given current state $x$, transition kernel $K(x, Y)$ generates possible set $Y$. Let $p(y|A_u) = p(y)I_{A_u}(y)/p(A_u)$ by Bayes' theorem. Then it could be written as

$$K(x, Y) = \frac{1}{Q(x)} \int_Y p(y) \int_{u=0}^{Q(x)} \frac{I_{A_u}(y)}{p(A_u)} dudy \tag{1.10}$$

Similarly the transition kernel of IMHA could be written as following (Mira and Tierney, 2002):

$$K_{IMHA}(x,Y) = \frac{1}{Q(x)} \int\limits_{Y} p(y) \int\limits_{u=0}^{min\{Q(x),Q(y)\}} dudy \tag{1.11}$$

In the process of Slice Sampling, if any $y$ is accepted, $q(y)$ must be greater than auxiliary variable $u$, otherwise $y$ is rejected. Then we can modify equation (1.10):

$$K(x,Y) = \frac{1}{Q(x)} \int\limits_{Y} p(y) \int\limits_{u=0}^{Q(x)} \frac{I_{A_u}(y)}{p(A_u)} dudy = \frac{1}{Q(x)} \int\limits_{Y} p(y) \int\limits_{u=0}^{min\{Q(x),Q(y)\}} \frac{1}{p(A_u)} dudy$$

Comparing kernel of Slice Sampling and IMHA, we get

$$\int\limits_{u=0}^{min\{Q(x),Q(y)\}} \frac{1}{p(A_u)} du \geq \int\limits_{u=0}^{min\{Q(x),Q(y)\}} du \tag{1.12}$$

In (Mira and Tierney, 2002) (Tierney, 1998), based on equation (1.12), it can be proved that Slice Sampling converge faster than IMHA, because Slice Sampling has less total variation distance to converge to any distribution. Hence Slice Sampling method can be applied to problems computable by IMHA.

## 1.3 Simulated Annealing

Another important algorithm used in our thesis is Simulated Annealing. Markov Chain Monte Carlo is good at sampling from desired distributions. However, our applications are more likely to search optimal solution for a particular question. Therefore Simulated Annealing has also been employed in this thesis.

Simulated Annealing is a general strategy used for locating the global best solution by given a desired distribution. It is usually used in the situation of a large searching space. Instead of searching for the best solutions, Simulated Annealing generates approximate solution which is close to the best solutions. In a large searching space, which could not be searched by brute force, Simulated Annealing could give solution depending on the running time. Usually the longer running time given, the better solution will be generated.

An important idea in Simulated Annealing is the feature of "temperature", $\tau$. In the beginning, temperature is usually set very high, while with time goes by, it slowly decreases. This process comes from the idea of annealing in metallurgy. The atoms are moving strenuously when temperature is high. And they keep steady, when temperature gets low. We reduce temperature gradually in order to make it possible for the search process to reach all areas of the search space and jump out of local optimal points. When temperature is low, search process is not as active as in the beginning, which makes search process to reach the peak of optima.

Simulated Annealing is easy to apply to any evaluable function. Say, we want to find out the global minimal value of function $h(x)$, and assume $h(x) > 0$. Intuitively, $h(x)$ could be any function. It is hard to search all possible $x \in (-\infty, \infty)$. In Simulated Annealing, we change $h(x)$ into a new form:

$$P(x) = \frac{1}{Z} * \exp^{-h(x)} \tag{1.13}$$

$Z$ is a fixed value of function, and $h(x)$ could be found out at any point $x$. For equation (1.13), we know that $P(x)$ is always between range $(0, 1)$. To introduce the feature of Simulated Annealing, a temperature variable can be used.We could rewrite equation (1.13) into new form as below:

$$P_\tau(x) = \frac{1}{Z_\tau} * \exp^{-h(x)/\tau} \tag{1.14}$$

Since equation (1.14) is in the form of distribution, MCMC methods is capable of finding global optimal solution of target function $h(x)$.

Simulated Annealing was first designed based on the Metropolis-Hastings method. Then its idea has been applied to a lot of algorithms of MCMC as an advanced usage. Simulated Annealing works well with most algorithms of MCMC, for example, the Metropolis-

Hastings, the Gibbs Sampling, and the Slice Sampling. There are two important ideas of Simulated Annealing. First is its mapping strategy, which converts any normal function into a distribution function by using the exponential function. Second is its feature of temperature. When the temperature is low, Simulated Annealing increase the difficulty of accepting new states. So if we want to find minimal value of $h(x)$, Simulated Annealing generates 1000 samples. Then the average value of the first hundred samples would be higher than average value of the last hundred samples. The detailed process of Simulated Annealing will be presented in later chapters for applications.

# CHAPTER 2

# APPLICATION OF IMAGE TRANSMISSION

Progressive image compression, such as SPIHT (Said and Pearlman, 1996), is an approach that exploits the inherent similarities across the sub-bands in a wavelet decomposition of an image, and the algorithm codes the most important wavelet coefficients first, and transmits the bits so that an increasingly refined copy of the original image can be obtained progressively. The progressive compression is widely used in many applications, because the media can be restored with the best quality by receiving a sequence of continuous error-free data. However, in the coded data stream, any error bit due to channel noise would cause the loss of synchronization between the sender and receiver, which means that all the data after that bit error has to be completely discarded. Therefore, an important issue in image transmission is to design a protection strategy for the source data, i.e., allocating channel code rates to different data packets, based on the channel condition and the rate-distortion feature of the source, in order to optimize the overall recovery quality of image in the noise channel.

In (Sherwood and Zeger, 1997), the cyclic redundancy check codes and rate compatible punctured codes (CRC/RCPC) were employed to protect SPHIT coded data and obtained performance better than previous results in binary symmetric channels (BSCs). This work used equal error protection and was then extended to the product code protection (Sherwood and Zeger, 1998) when the Gilbert-Elliot channel (GEC) model was considered. Since then, many error-control solutions for progressive image transmission (Sherwood and Zeger, 1998; Mohr et al., 2000; Chande and Farvardin, 2000; B.A., 2002; Stankovic et al., 2003; Nosratinia et al., 2003; Stankovic et al., 2004, 2005; Thomos et al., 2005) have been proposed.

In these methods, different codes are used, including CRC/RCPC (Chande and Farvardin, 2000; Stankovic et al., 2003; Nosratinia et al., 2003), cyclic redundancy check codes and rate compatible punctured turbo (CRC/RCPT) codes (B.A., 2002; Stankovic et al., 2004, 2005; Thomos et al., 2005), Reed-Solomon (RS) codes (Mohr et al., 2000; Thomos et al., 2005) and their product codes. Different channel conditions are also considered, including BSC, GEC, and packet loss channels. In multimedia network, JPEG2000 lea12:fast can be used to scale the performance, based on the data rate. Therefore, a research focus is to provide optimal unequal error protection (UEP) to the coded data packets where bits coded early represents greater significance in image reconstruction than bits coded later.

All existing optimal methods consider only the fixed channel condition. In practice, however, the channel condition always change with the time, mainly due to the mobility and multi-path factors in the communication. As a result, the designed UEP system has to be updated with frequent feedback of the channel condition. This is impractical as the multi-dimensional optimization process is often very complicated and the channel variation can be fast. Therefore, a more realistic solution is to design the protection method considering all the channel conditions. The channel variation, which is captured by the probability density distribution (PDF) of SNR, can often been estimated from a long period of operation of the communication network. Considering the channel PDF, the optimization problem is even more complicated than the previous design because channel effects must also be marginalized.

In this paper, the simulation is calculating the image distortion of different allocation of UEP system, since any error would cause distortion in image transmission in SPIHT. So the target is to find out the optimal allocation to make system has lowest distortion in the varying SNR channel. The MCMC technique is applied to optimal UEP system design. We use the exponential function of simulated annealing method to map the cost function into a probability distribution, and use the slice sampling of MCMC. Since the samples drawn from MCMC approach the mapped stationary probability distribution, the MCMC method provides more probabilistic information than other heuristic optimization method. We show that the MCMC method provides a low complexity design with a solution approaching the optimal one. Finally, we show that the system design based on channel distribution indeed outperforms the design based on a specific channel statistic such as the mean value.

## 2.1 Problem Description

With time goes by, the channel condition is always varying. Optimization of allocation of message packets based on a specific $SNR$ is not realistic, especially the channel is changing dramatically. So it is advantageous to optimize the rate allocation based on the statistics of channel on time period instead of a particular channel $SNR$. We consider a joint source-channel coding system with $N$ coded packets with fixed packet length as shown below,

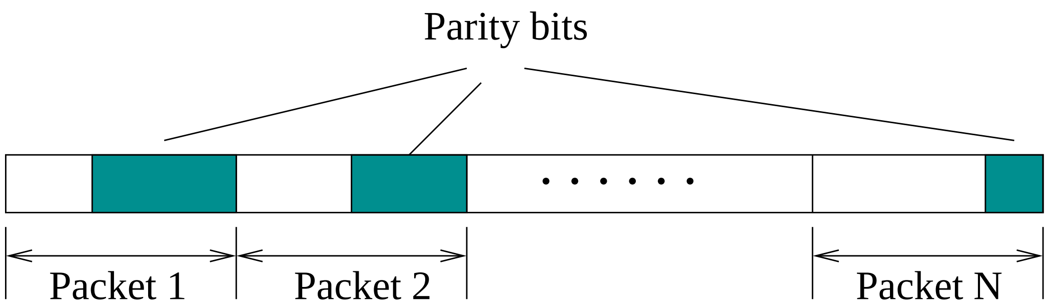

Figure 2.1. A message via channel contains N packets. Each packet could contain diferent length of parity part and data part, but each packet has the same length of bits.

This data structure was provided by B. Banister in (B.A., 2002) and has been adjusted for many applications. Rate compatible punctured turbo (RCPT) codes are used to provide unequal error protection. Assuming the space of different channel code rates is in order, and in form of,

$$\mathcal{C} = \{c_1, c_2, c_3 \ldots, c_M\}, c_1 > c_2 > \cdots > c_M$$

which represents the rate of the source data over the packet length. Since each packet has the fixed length $L$ ( In our work, we set coded packet size $L = 4096$ ), the corresponding number of source bits for each code rate is $s_i = c_i \cdot L$. Therefore the number of the overall source bits is

$$\sum_{i=1}^{N} c_i \cdot L$$

and the number of overall channel parity bits is

$$(1 - \sum_{i=1}^{N} c_i) \cdot L$$

After being transmitted over a specific channel with $SNR = x$ dB, the packet error probabilities after decoding are denoted as $\mathcal{E} = \{e_1(x), e_2(x), \ldots, e_M(x)\}$, where $e_1(x) > e_2(x) > \cdots > e_M(x)$ (Cao, 2007b).

In this paper, the Turbo code $(15, 17)_{oct}$ with mother code rate of 1/3 is considered. A set of code rates of $\{4/4, 4/5, 4/6, 4/7, 4/8, 4/9, 4/10, 4/11, 4/12\}$ are obtained through puncturing (Rowitch and Milstein, 2000). The first bit of number represent the data (the white part of each packet in Figure 2.1. The second bit of number represents the total length of a packet. For example, 4/8 means half of the packet is data, and the other half is protection code.

Figure 2.2 shows the residual packet error rate (PER) in different additive white Gaussian noise (AWGN) channel conditions. It needs to be noted that the PER also depends on the packet size and can be different if a different packet length is used. A larger data length increases the coding gain in turbo decoding in general.

Suppose a message contains $N$ coded packets being transmitted, each packet being protected with channel code rate of $r_i \in \mathcal{C}, i = 1, ..., N$. $N$ is determined by the total transmission rate and the packet length. For channel $x$, the corresponding packet error rate is $p_i(x) \in \mathcal{E}$, which is probability of error occur on $i$th packet. Denote $P_i(x)$ as the probability that the first $i$ th packets are decoded without errors, howerver the $(i + 1)$ th packet is not correctly decoded. Then we have

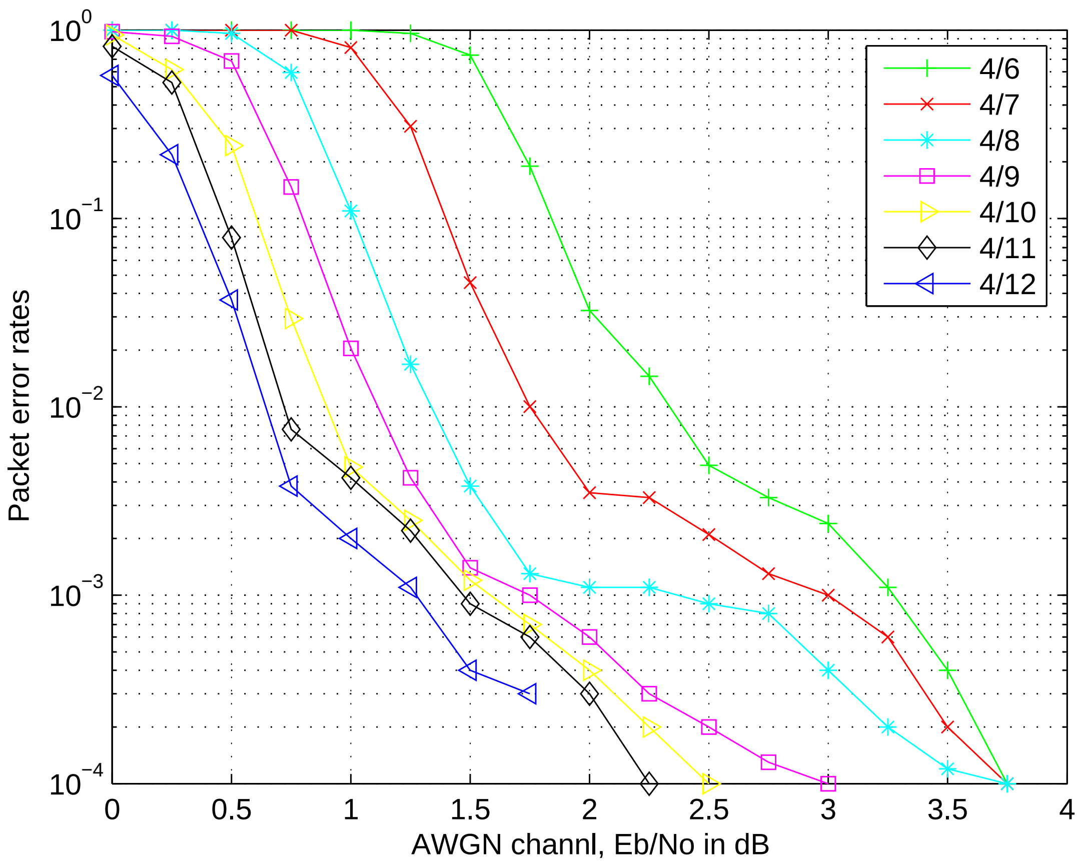

Figure 2.2. Turbo code performance for fixed coded packet length. Package error rate in different AWGN channel SNR (Cao, 2007a).

$$P_i(x) = \begin{cases} p_1(x) & \text{if} \quad i = 0 \\ \Pi_{j=1}^{i}(1 - p_j(x))p_{i+1}(x) & \text{if} \quad i = 1, \dots, N-1 \\ \Pi_{j=1}^{N}(1 - p_j(x)) & \text{if} \quad i = N \end{cases} \tag{2.1}$$

Intuitively, any packets after error occurs is no longer considered, because SPIHT discard all packets after any error.

Distortion between the recovered image and the original image is represented by the mean square error (MSE). Apparently it is based on the correctly decoded source bit. Let $D_i$ be the distortion of the image restored by the first $i$ packets that have been correctly decoded, and $(i+1)$ th packet is uncorrect. Then, the message has an allocation $\mathcal{R}_N =$

$\{r_1, r_2, \ldots, r_N\}, r_i \in \mathcal{C}$, of channel code rates, the expected distortion is

$$D_N(x, \mathcal{R}_N) = \sum_{i=0}^{N} P_i(x) D_i \tag{2.2}$$

This distortion in equation (2.2) has a recursive representation (Chande and Farvardin, 2000) as:

$$\begin{aligned} D_N(x, \mathcal{R}_N) &= D_0 - E_N(x, \mathcal{R}_N) \\ E_N(x, \mathcal{R}_N) &= (1 - p_1)[\Delta D_1 + E_{N-1}(x, \mathcal{R}_{N-1})] \end{aligned} \tag{2.3}$$

where $D_0$ is the distortion of the case that no correct packet is received. $\Delta D_i$ is the reduced distortion between $D_{i-1}$ and $D_i$. Also $\mathcal{R}_{N-1} = \{r_2, \ldots, r_N\}$ is the rate allocation for the last $N - 1$ packets.

When the channel SNR is given, a dynamic programming method was suggested in (Chande and Farvardin, 2000) for this distortion-based optimization problem. This method is a backwards updating process. That is, the reduced distortion of a latter packet should be used first to determine the optimal allocation. However, the reduced distortion in a latter packet actually depends on the channel code rates in former packets, which is evident in equation (2.3). Therefore, the forward-updating process along trellis is in fact more practically used (B.A., 2002; Thomos et al., 2005) to reduce the complexity. However, the solution becomes sub-optimal as indicated in (B.A., 2002).

For a varying channel, if the channel SNR density function $f_X(x)$ is known, then the joint source-channel coding problem becomes to find a channel code rate set $\mathcal{R}_N$ which minimizes, subject to a given overall transmission rate,

$$\begin{aligned} D_N(\mathcal{R}_N) &= \int_{-\infty}^{\infty} f(x) D_N(x, \mathcal{R}_N) dx \\ &= \int_{-\infty}^{\infty} f(x) \sum_{i=0}^{N} P_i(x) D_i dx \end{aligned} \tag{2.4}$$

Apparently, this is different from finding the optimal rate-allocation for the mean channel condition, because mean SNR is a fixed value, so equation (2.2) should be used,

$$D_N(\bar{x}, \mathcal{R}) = \sum_{i=0}^{N} P_i(\bar{x}) D_i \tag{2.5}$$

where $\bar{x}$ is the mean channel SNR. The varying channel case is more complicated than considering only on the specific channel instance as the channel variable $x$ must be marginalized in the optimization process. In later section, we propose to solve this optimization problem with the MCMC method.

## 2.2 Mapping Cost Function to Probability

The cost function has the channel code rate of each packet as input, and the distortion value as output. if the image message is transported by $N$ packets, there are $N$ inputs $r_i$. Intuitively the allocation $\mathcal{R}_N = \{r_1, \ldots, r_N\}$, should be considered in order to generate output. The optimization task is to find allocations $\mathcal{R}_N$ that gives the minimum distortion value. The first issue in using MCMC is the need of mapping the cost function into a likelihood function that could be used as the stationary probability distribution in the Markov Chain. We use the function in the Simulated Annealing for this purpose, i.e,

$$\frac{1}{Z_\tau} \exp\left\{-\frac{J(\cdot)}{\tau(t)}\right\} \tag{2.6}$$

where $Z_\tau$ is a normalization constant so that any possible value is always between $(0, 1)$, and $J(\cdot)$ is the cost function. Note the sum of all probability is 1.

If the current state $x(t)$ is equal to $i$, suppose a neighbor $j$ of $i$ is visited with a probability $q_{ij} = q_{ji}$. Then whether $j$ will be selected as a new sample is given by the probability of

$$min\left\{1, \exp\left[-\frac{J(j) - J(i)}{\tau(t)}\right]\right\}. \tag{2.7}$$

$i$ is current state $x(t) = i$ and next state $x(t+1) = j$. Any value greater than one will be set as one, since probability is always between $(0, 1)$. This is termed as the Metropolis process and it can be proved that with this process, the stationary probability of the Markov Chain is (2.6). It can be noted that

1. If $J(j) \to \infty$, then $\exp\{-[J(j) - J(i)]/\tau(t)\} \to 0$. In this case the accepting probability for $j$ is tending to 0. That is, a state $j$ associated with a high cost has a low chance to be accepted ( the higher the cost function $J()$ is, the lower chance the system $x(t+1)$ accept new state $j$.

2. On the other hand that $J(j) \to 0$ means the new cost is negligible. Then $-[J(j) - J(i)]/\tau(t)$ could be positive, which implies the new state $j$ is always selected due to its lowest cost value

The parameter $\tau(t)$, termed as the temperature in the simulated annealing, also needs to be considered carefully. To make the algorithm approach the optimal solution, the temperature changes with the time. The temperature function is a non-increasing function, which represents the cooling down process. $\tau(0)$ is usually set as a high value in the beginning of simulation. This makes sure that with a high probability a new state will be accepted. In this stage, system may reach more diverse and different states based on the probability contribution. When $\tau(t)$ approaches 0, the probability becomes very small. The system is reluctant to accept a new state $j$, which is often described as a “Frozen” stage. The process of decreasing of temperature is usually slow to give system enough time to be ”trapped” in the low cost area. In our simulation, we reduce the temperature after each new sample is generated.

In an ideal case, we hope $\tau(t)$ would reach 0 in the end of simulation, which means totally frozen of the system. However in most case, it is almost impossible to achieve that, because with the time $t$ passing through, new acceptable state becomes harder and harder to be found. That is why equation (2.8), uses limit and infinity time to represent the condition that Simulated Annealing reach the optimal solution set.

$$\lim_{t\to\infty} P[x(t) \in S^*] = 1 \tag{2.8}$$

$S^*$ is the optimal solution set. In any simulation, an adaptive temperature function $\tau(t)$ is always a important factor of the time performance. In case of big initial $\tau(0)$, temperature would take unnecessary extra time to reach 0; Also if $\tau(0)$ is too small, the system may still stay in a local minimum,without a chance to jump out yet. In Hajek's paper (Hajek, 1988) in 1988, a popular temperature function is given like following:

$$\tau(t) = \frac{d}{log(t)}$$

The $d$ here is a constant number which is considered as the measure of how difficult for state $x(t)$ to jump out from the current local minimum and travel to optimal state space $S^*$. In the other words, simulation need a large enough $d$ to make sure, simulation has enough time to jump out local minimum and achieve the global optimal set.

## 2.3 MCMC Pseudocode for UEP protection in Image Transmission

The distortion function is the cost function in the simulation. The goal is to find the lowest distortion based on the packets rate. We assume the channel SNR is in the range of $[0, 4]$dB as shown in Figure 2.2. . The turbo code rate space is one of the 9 different numbers in the set of

$$\mathcal{C} = \{4/4, 4/5, 4/6, 4/7, 4/8, 4/9, 4/10, 4/11, 4/12\}$$

Each number in the space represents the rate of data code and total code. $\mathcal{R}_N$ is the input of the cost function. It is a vector with $N$ elements, each representing the code rate for each packet, i.e., $\mathcal{R}_N = \{r_1, r_2, \ldots, r_N\}$, and $r_i \in \mathcal{C}$. Apparently, our simulation Markov Chain has the invariant probability distribution like

$$J(\mathcal{R}_N) = D_N(\mathcal{R}_N),$$

and then,

$$P[x(t) = \mathcal{R}_N] = \frac{1}{Z_\tau} \exp[-\frac{D_N(\mathcal{R}_N)}{\tau(t)}]$$

where $Z_\tau$ is a normalizing constant. Because the probability values are only compared in the slice sampling, we can simply set $Z_t = 1$. The method used is actually a combination of simulated annealing and slice sampling. We use the probability representation of simulated annealing and reduce the temperature with time, while use the slice sampling to access new states and generate new samples. When $\tau \to 0$, the probability distribution $\pi_\tau$ would trapped in the optimal set. Note that in a multiple input problem, it is possible for different inputs share same value of output, and the optimal solution is not necessarily unique.

The MCMC process is modified like below,

1. evaluate $P[x(t) = \mathcal{R}_N]$
2. draw $u \sim Uniform(0, P[x(t) = \mathcal{R}])$
3. create neighbor space to enclosing every $r_i \in \mathcal{C}$,

   $(r_i^{left} \leq r_i \leq r_i^{right})$

```
4. loop {
5.     draw new $x(t+1) = \mathcal{R}'_N$
6.     evaluate $P[x(t+1) = \mathcal{R}'_N]$
7.     if $P[x(t+1) = \mathcal{R}'_N] \geq u$, break out loop
           the next state $x(t+1) = \mathcal{R}'_N$ is accepted
8.     else modify the neighbor space
9. }
```

Since it is multiple input, Note that

Step 1: The state $x(t) = i$ becomes $x(t) = \mathcal{R}_N$ in the MCMC process. $P[x(t) = \mathcal{R}_N]$ is a function for the initial input $\mathcal{R}_N$.

Step 3: for $\mathcal{R}_N = \{r_1, r_2, \ldots, r_N\}, r_i \in \mathcal{C}$, constructing a interval space for each variable $r_i$. That makes the neighbor space look like:

$$space = \{(r_1^{left}, r_1^{right}), \quad (r_2^{left}, r_2^{right}), \ldots, (r_N^{left}, r_N^{right})\}$$

In the process, Current state $x(t) = \mathcal{R}_N$, and $\mathcal{R}'_N$ is the next state $x(t+1)$. Also $\mathcal{R}'_N = \{r'_1, r'_2, \ldots, r'_N\}$, $r'_i \in \mathcal{C}$ is the next state corresponding to $r_i$. In order to find $x(t+2) = \mathcal{R}''_N$, which is the next state of $\mathcal{R}'_N$, $\mathcal{R}'_N$ should be put in the *step* 1 and repeat the process. Simulation keeps doing that to generate samples.

Our simulation uses parallel process to find the neighbor space, i.e, neighbors of all $r_i$,$i = 1, \ldots, N$ be updated before next state is drawn and checked. A concrete process of *step* 3 is like this,

```
1. set a value for "wide", call $w_i$
2. loop ($r_1 \rightarrow r_N$){
 3.     generate $rand \sim Uniform(0,1)$
 4.     $r_i^{left} = r_i - rand * w_i$
 5.     $r_i^{right} = r_i + rand * w_i$
 6.     check $r_i^{left}, r_i^{right} \in \mathcal{C}$
 7.     while $P[\{r_1, \ldots, r_i^{left}, \ldots, r_N\}] > u$, then we have $r_i^{left} = r_i^{left} - w_i$
 8.     while $P[\{r_1, \ldots, r_i^{right}, \ldots, r_N\}] > u$, then we have $r_{right} = r_{right} + w_i$
 9.     check $r_i^{left}, r_i^{right} \in \mathcal{C}$
10. }
```

The process is same as the one-dimensional case, except that a *loop* is needed here. Because the neighbor space needs to be checked for each $r_i \in \mathcal{R}_N$. In addition, different $w_i$ may be used if necessary. For example, in some cases the input may have different range. A common practice is to set $w_i$ as a certain percentage of the whole range.

Step 8: Let $\mathcal{R}'_N = \{r'_1, \ldots, r'_N\}$ be the new state, then the neighbor space is modified as

```
1. loop $(r_1 \to r_N)\{$
2.     if $r'_i > r_i$, then $r_i^{right} = r'_i$
3.     else $r_i^{left} = r'_i$
4. }
```

Since the code rate space is $\{4/4, 4/5, 4/6, \ldots, 4/12\}$, which contains 9 possible values. Instead of multiplying $w$, a random integer, $rand \sim Randn(0, w)$, could be applied as $r_i^{right} = r_i + rand$ in simulation.

## 2.4 Simulation and Result

In this section, the MCMC method has been applied to design the UEP system for progressive image transmission. The Lena image with size of $512 \times 512$ and 8 bits per-pixel (bpp) was passed through the SPHIT algorithm to obtain the source data. RCPT codes are then used for channel protection, where a set of numbers is used to represent the set of channel coding rates, as shown in Table (2.1). The distortion value is the MSE between the reconstructed image and the original image.

| rate | 4/4 | 4/5 | 4/6 | 4/7 | 4/8 | 4/9 | 4/10 | 4/11 | 4/12 |
|---|---|---|---|---|---|---|---|---|---|
| represent | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Table 2.1. Rates corresponding to numbers. Number will be used in later chapter to represent the rate of each input.

We first consider a case with a small number of packets, with the purpose to check whether the MCMC method can perform as expected. Then we consider cases with more number of packets and different channel probability distributions to study the performance based on the MCMC method. We also compare the design results based on the channel distribution and the mean value of the channel. Also note that each simulation trial generates 10000 samples in this paper.

### 2.4.1 Case of 10 packets

The channel distribution is uniformly distributed, i.e., $f_X(x) \sim Uniform[0, 4]\ dB$. MCMC is used to generate the samples of $\mathcal{R}_N$ based on the probability mapped from the cost function. Different trials are conducted, i.e., the MCMC process are repeated to find independent UEP designs. It was found that the MCMC almost always gives the optimal $\mathcal{R}_N$. It closely approaches the optimal design even if the optimal result was not reached occasionally.

With the simulation runs by, new minimum distortion value and related rate allocation $\mathcal{R}_N$ would be found, the program is set to put these date into a table, called the minimum table, such as Table (2.2). It shows the last eight minimal results for the case of 10 packets.

| Input rate | Distortion |
|---|---|
| 9 8 6 6 6 4 3 3 2 1 | 243.77079 |
| 9 8 7 6 6 5 4 3 3 1 | 236.55292 |
| 9 9 9 7 7 5 5 3 2 1 | 234.74915 |
| 9 8 6 6 5 5 5 5 4 4 | 230.74674 |
| 9 9 9 6 5 5 5 4 3 3 | 225.47482 |
| 9 9 9 9 8 7 6 5 4 3 | 221.22013 |
| 9 9 9 8 8 7 6 5 5 3 | 221.20497 |
| 9 9 9 8 8 7 6 5 5 4 | 220.76209 |

Table 2.2. Table of Minimum: Uniform / 10 packets / $SNR = [0, 4]dB$

In this trial, the best solution given is $\mathcal{R}_N = \{9, 9, 9, 8, 8, 7, 6, 5, 5, 4\}$ with the lowest distortion value 220.76209 in the table.

To validate the results of MCMC, we exhaustively enumerate all possible $\mathcal{R}_N$ for this 10 packets case. Brute force method has been used to compute all the $9^{10}$ different rate allocations and took significant amount of time with a PC(in this case, it is two weeks). By using several computers, Table (2.4) may be generated in shorter time, but in most case it could take weeks or months to calculate the table. A few least distortion values and the number of corresponding allocations $\mathcal{R}_N$ to achieve each of these values are listed in Table (2.4).

Here is the partial table of brute force result:

| Distortion | 220.7 | 220.8 | 220.9 | 221.0 | 221.1 |
|---|---|---|---|---|---|
| Frequency | 1 | 2 | 5 | 9 | 13 |
| Distortion | 221.2 | 221.3 | 221.4 | 221.5 | 221.6 |
| Frequency | 27 | 34 | 40 | 68 | 103 |

Table 2.3. Frequency from brute force: Uniform / 10 packets

The frequency of distortion occurs by different input, $\mathcal{R}_N$ is shown. For example, distortion 221.5 has a frequency value of 68, in other words there are 68 different input, $\mathcal{R}_N$, result in distortion value of 221.5. Apparently, the global minimum is 220.7 here. Compared with the Table (2.2), it is clear $\mathcal{R}_N = \{9, 9, 9, 8, 8, 7, 6, 5, 5, 4\}$ actually achieved the global minimum of all possible rate allocation, which means it is the optimal solution.

Also by further testing, in the case of 10 packets, 8 out of 10 times, the MCMC simulation could achieve the best solution, with no matter what distribution $f_X(x)$ of the channel SNR is. And in the rest cases, MCMC gives results very close to the optimal solution.

An important issue here is if the MCMC simulation really draws samples from the target distribution plot. In order to prove this, we sample 10,000 samples by Slice Sampling, and plot the histogram together with the results obtained from brute force search. The results are shown in Figure 2.3.

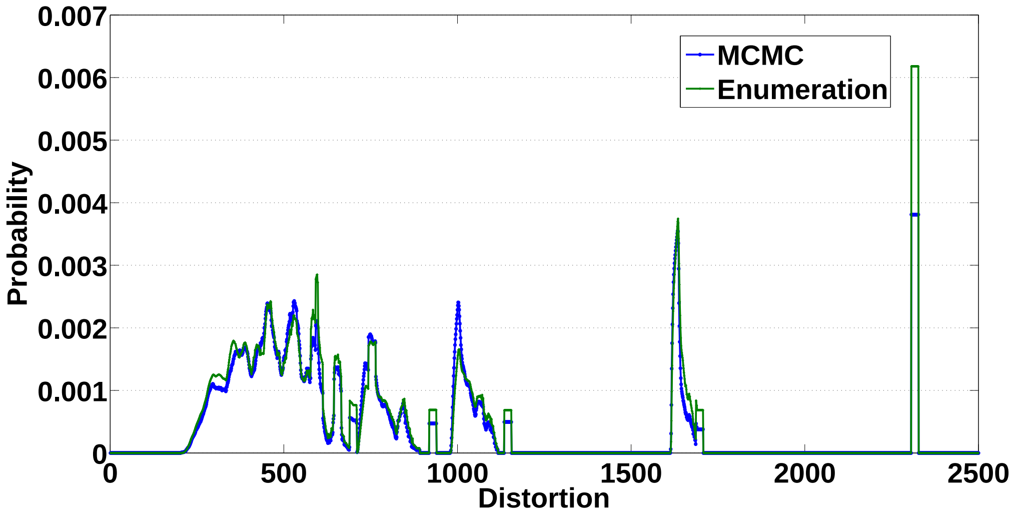

Figure 2.3. MCMC vs Enumeration: Uniform / 10 packets / $SNR = [0, 4]dB$ and fixed $\tau = 1000$. The histograms of all possible distortion based on both MCMC and enumeration are collected.

In this simulation, the temperature $\tau(t)$ was set as a large constant value (i.e., 1000 in simulation) and was not decreased after each sample generated, which means that the simulation become a normal slice sampling method. To make the shape clear in a large x-axis scale, an average function was also used. That is, every $x$ axis and $y$ axis value on this figure are the average of the a few neighbors and itself, i.e, $x_i^{new} = (x_{i-4} + x_{i-3} + \cdots + x_{i+4})/9$

(In our case, it is 8 neighbors. To make shape more smooth, more neightbors should be chosen). However result of bruce force has much more samplers than our simulation, which means brute force has much higher $y$ axis value than our simulation. So we have to normalize the $y$ axis to make they could be shown on same figure by dividing each distortion point by the number of samplers for both methods. According to Figure 2.3, the results of MCMC and brute force method have very similar shape, which verifies that MCMC indeed generates samples according to the mapped distribution. This property illustrates that the MCMC have visited all regions of the parameter space based on the their contribution to the mapped probability.

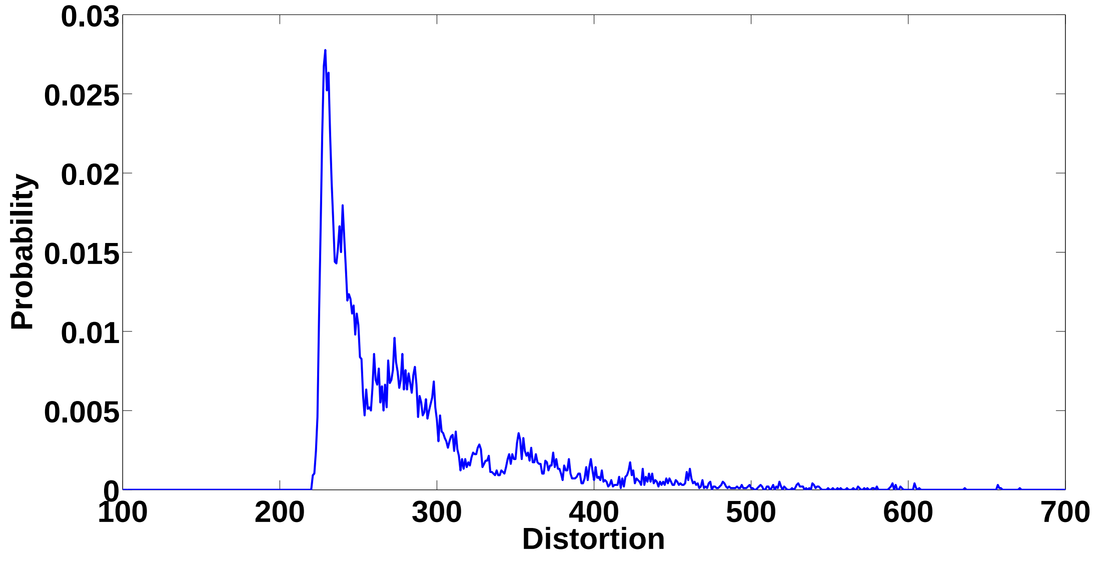

Figure 2.4. Simulation result with reducing temperature, Uniform distribution, 10 packets. The probability is the histogram of different distortion value divided by number of samples.

Figure (2.4) is pdf of Slice Sampling with using the mapping and temperature function of Simulated Annealing. After each new sample is generated, program reduces temperature by 0.1 starting from 1000. Figure (2.4) is very different form Figure (2.3). The samples are squeezed on the left side. Since the temperature is decreasing with time passes by, the samples with small distortion values have higher probability to be picked up in order to find global optimal value. Hence, the histogram is attracted to the left side of $x$ axis so as to escape from local best set to optimal set. Also on the figure, we can see the minimal distortion in this trial is aroud 220.

### 2.4.2 Case of 32 packets

In 32 package case, the method work well also, except the time performance. The complexity of computation is greatly increased in this case, because of a lot more calculation needed for neighbor space checking of each $r_i \in \mathcal{R}$. In the case of 32 packets, Uniform distribution $f_X(x)$ is also assumed.

| Input rate | Distortion |
|---|---|
| 99999988888877765555444444332211 | 49.71207 |
| 99999988888877766655554444332211 | 49.43509 |
| 99888888887777766655555544333321 | 48.06369 |
| 99999998888887777666555554443331 | 46.64615 |
| 99998888888877777666665554444431 | 46.55235 |
| 99988877777776666666555444444333 | 46.49173 |
| 99999998888887777666665555544443 | 45.31980 |
| 99999988877766666655555555554443 | 45.28976 |
| 99999988888877776666655555554444 | 45.19252 |
| 99999988888877776665555555554443 | 45.19155 |

Table 2.4. Table of Minimum: Uniform / 32 packets / $SNR = [0, 4]dB$

This trial runs for couple minutes in simulation. However the brute force may take billion years to do the exactly same thing, since there are $9^{32}$ different possible $\mathcal{R}$ exist. And unfortunately, it is still not the worst case, since a huge message can be easily divided into 64, 128, 256 or even more packets. So we are not able to calculate a table such as Table (2.2) to put frequency.

One idea can be drawn so far, by looking at the minimum table (2.1) and (2.4) is that each rate$r_i \in \mathcal{R}$ is not equally important. It has been proved by hundreds trials so far and is easy to be figure out. Due to the fact that an early package lost will cause discard of packets starting from the error occurs, the packet loss in the beginning is very critical to the whole performance. These first couple packets should be considered more important than the last several packets and are deserved higher protection level. Also related this aspect to the result in the trials of Table (2.1) and (2.4), the best results found are like following:

9 9 9 9 8 7 6 5 5 4 (10 bits);

9 9 9 9 9 9 9 9 9 9 9 9 9 8 8 8 8 8 7 7 6 6 5 5 5 5 5 5 4 4 4 3 (32 bits);

Obviously, these result all start with a sequence of "9", which is 4/12, the highest rate between protection code and data code, and representing the highest error protection.

### 2.4.3 Compare with mean value design SNR channel

In this section, simulation based on varying $SNR$ and Mean $SNR$ are compared, results are showed in the Table 2.5, 2.6, 2.7, and 2.8. Three different probability density functions are assumed, which are $N(2, 0.5)$, $N(2, 1)$ and Uniform distribution. In order to compare varying $SNR$ channel and mean value $SNR$ channel, considering varying $SNR = [0, 4]$dB, 2dB is assigned to mean $SNR$ channel. In the mean $SNR$ case, the optimal solution $\mathcal{R}_{mean}$ is generated by MCMC simulation. Then $\mathcal{R}_{mean}$ is put back into the $SNR = [0, 4]dB$ condition to find out the MSE and PSNR, since the channel of varying $SNR$ condition is more likely to channel of real world than the channel of mean $SNR$ condition.

Table 2.5. Performance based on different *bpp* on range of $SNR = [0, 4]dB \sim N(2, 0.5)$

| Packets number | Optimal solution | Overall bpp | Source code bpp | Protect code bpp |
|---|---|---|---|---|
| 10 | 9876555544 | 0.1563 | 0.0732 | 0.0831 |
| 16 | 9998877655 555444 | 0.25 | 0.1123 | 0.1377 |
| 32 | 9(6) 8(6) 7(4) 6(3) 5(9) 4(3) 3(1) | 0.50 | 0.2187 | 0.2813 |
| 48 | 9(10) 8(7) 7(9) 6(5) 5(10) 4(6) 3(1) | 0.75 | 0.3249 | 0.4251 |
| 64 | 9(21) 8(10) 7(5) 6(11) 5(10) 4(6) 3(1) | 1.00 | 0.4159 | 0.5841 |

Table 2.5 shows the source code bpp and protection code bpp under condition of different overall bpp. The table also contain the optimal result generated by MCMC.

Table 2.6, 2.7, 2.8 shows the comparison between varying channel and mean SNR channel. In each table, the 2nd and 3rd columns are the varying SNR channel and last two columns are mean SNR channel. Apparently, simulation proved that MCMC based on

Table 2.6. Performance based on channel of $SNR = [0,4]dB \sim N(2,0.5)$ and channel of Mean $SNR = 2dB$

| Packets number | MSE $[0,4]dB$ | PSNR $[0,4]dB$ | MSE $[2]dB$ | PSNR $[2]dB$ |
|---|---|---|---|---|
| 10 | 104.07093 | 27.9575 | 160.6278 | 26.0726 |
| 16 | 73.98707 | 29.4392 | 88.4346 | 28.6646 |
| 32 | 45.19155 | 31.5802 | 48.3620 | 31.2858 |
| 48 | 35.13393 | 32.6735 | 37.6239 | 32.3762 |
| 64 | 29.95420 | 33.3662 | 30.9719 | 33.2211 |

Table 2.7. Performance based on channel of $SNR = [0,4]dB \sim N(2,1)$ and channel of Mean $SNR = 2dB$

| Packets number | MSE $[0,4]dB$ | PSNR $[0,4]dB$ | MSE $[2]dB$ | PSNR $[2]dB$ |
|---|---|---|---|---|
| 10 | 135.9264 | 26.7978 | 234.1765 | 24.4354 |
| 16 | 105.3338 | 27.9051 | 136.1534 | 26.7905 |
| 32 | 76.27627 | 29.3069 | 82.2316 | 28.9804 |
| 48 | 66.13733 | 29.9263 | 71.5604 | 29.5841 |
| 64 | 61.02982 | 30.2754 | 62.7087 | 30.1575 |

varying $SNR = [0,4]dB$ channel can always obtain better PSNR and lower MSE than mean value $SNR$ channel.

The cost performance can be calculated and has been put in the Table (2.9). For any $r_i \in \mathcal{R}$, the space of $r_i$ is $\{4/4, 4/5, \ldots, 4/12\}$, there are 9 possible numbers are assumed. Apparently for $\mathcal{R} = \{r_1, r_2, \ldots, r_N\}$, there are $9^N$ different $\mathcal{R}$, since each $r_i$ has 9 possible value. By counting the time of calculation of MCMC and compared with the number of possible inputs based on each bpp, cost performance table could be generated like

Table 2.8. Performance based on channel of $SNR = [0,4]dB \sim Uniform$ and channel of Mean $SNR = 2dB$

| Packets number | MSE $[0,4]dB$ | PSNR $[0,4]dB$ | MSE $[2]dB$ | PSNR $[2]dB$ |
|---|---|---|---|---|
| 10 | 220.76209 | 24.6916 | 515.7425 | 21.0065 |
| 16 | 191.32468 | 25.3131 | 255.8785 | 24.0505 |
| 32 | 163.88001 | 25.9855 | 176.9067 | 25.6534 |
| 48 | 154.31311 | 26.2468 | 164.3646 | 25.9727 |
| 64 | 149.55694 | 26.3827 | 151.9254 | 26.3145 |

Table 2.9. Cost based on different bpp with $N(2, 1)$ and $SNR = [0, 4]dB$

| bpp | 0.1563 | 0.25 | 0.50 | 0.75 | 1.00 |
|---|---|---|---|---|---|
| packet num | 10 | 16 | 32 | 48 | 64 |
| possible input | $9^{10}$ | $9^{16}$ | $9^{32}$ | $9^{48}$ | $9^{64}$ |
| cost | 1.01e-005 | 8.07e-12 | 5.68e-27 | 3.24e-42 | 2.34e-57 |

Table2.9. Compared with the exponential increasing of the number of possible input, the cost dramatically decreasing, because the calculation time of MCMC increase linearly.

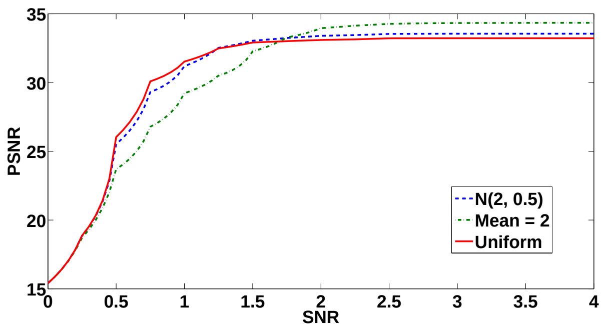

Figure 2.5. PSNR of optimal $\mathcal{R}$: 32 packets / $SNR = [0, 4]dB$

Figure 2.5 shows the performance of optimal $\mathcal{R}$ generated by MCMC method based on different channel conditions. Optimal solution $\mathcal{R}$ usually has the greatest average PSNR or lowest MSE in order to make it the best solution. In the figure 2.5, $\mathcal{R}_{mean}$ results has lower PSNR than others in range 0 to 1.75 approximately, but in range 1.75 it has better PSNR. However what we looking for is the average PSNR, in the case $N(2, 0.5)$ and uniform distribution, the area of PSNR are bigger than Mean channel. So we say optimal $\mathcal{R}$ based on a real distribution density function has better performance than Mean SNR channel.

# CHAPTER 3

# APPLICATION OF LOCATION PLANNING IN INTERMODAL TRANSPORTATION

In this chapter, we consider the planning of terminal locations for intermodal transportation systems. With a given number of potential locations, we aim to find the most appropriate number of those as terminals to provide the economically most efficient operation when multiple service pairs are needed simultaneously. The problem also has an inherent task to determine the optimal route paths for each service pair. For this NP-hard problem, we present a Markov Chain Monte Carlo (MCMC)-based two-layer method to find a suboptimal solution. In the lower-layer, the routing for all service pairs given a particular location planning is solved through a table-based heuristic method that considers both efficiency and fairness. In the upper-layer, by mapping the cost function into a stationary distribution, the optimal planning is solved based on a MCMC method that integrates advantages of both simulated annealing and slice sampling. Finally, the effectiveness of this Heuristic MCMC-based method is demonstrated through computer experiments.

## 3.1 Introduction

In 2010, the US President announced that the US Federal Government will achieve a reduction of emission of greenhouse gas (GHG) pollution by 28% by year 2020. As the largest consumer of US, Federal government spent at least 24.5 billion on energy and gasoline in year 2008. If the goal achieved, as stated in (Office, 2010) the federal government could reduce about 205 million barrels of oil, which also means taking at least 17 million cars off road for one year. . According to (UIRR, 2009), (UNECE, 2009), the transportation system plays a key role of polluting environment and is responsible for 30% of the total greenhouse gas emission. Intermodal transportation is considered as one key part of the solution for this problem. With reduced delay, congestion and operational cost compared to conventional transportation systems, intermodal transportation can significantly improve economic competitiveness, as well as help to sustain a more amiable environment by significantly reducing the total $CO_2$ emissions.

Considering the interconnection of the road mode and other modes such as railway transportation via terminals so that the road mode focuses more on many low-flow localized services and the railway mode acts as the major backbone for high capacity and large range services, one specific problem is to decide on the number of terminals and their locations given a set of potential terminals and to determine the route paths of difference services.

The current research in this area has considered different intermodal representation models, such as (Macharis and Bontekoning, 2004; Meinert et al., 1998; Ishfaq and Sox, 2011), and different heuristic optimization methods, such as (Bornstein and Azlan, 1998; Jaramillo et al., 2002; Filho and Galvao, 1998; Sorensen et al., 2012). In (Macharis and Bontekoning, 2004), an overview of the most prominent research efforts within operational research in the intermodal transportation has been provided. Compared with simulation-based techniques in (Meinert et al., 1998), using network models becomes more popular in research. In (Ishfaq and Sox, 2011) an overview is provided on several network models based on which the optimization process is then carried out. As proven in (Sorensen et al., 2012), this location planning problem is an NP-hard problem and hence the deterministic methods are impractical when the size of the network grows. As a result, when the number of nodes increases, an optimization method based on the heuristic search has to be applied, such as using simulated annealing in (Bornstein and Azlan, 1998), genetic algorithms (GA) in (Jaramillo et al., 2002), Tabu search in (Filho and Galvao, 1998) and greedy randomized adaptive search procedure and attribute-based hill climber in (Sorensen et al., 2012).

In this chapter, based on the terminal location modal proposed by Arnold (Arnold P, 2001), we present a new method using Markov Chain Monte Carlo (MCMC). MCMC is a Bayesian inference method that is commonly used in numerical generation of complex probability distributions with a high dimensional variable space. MCMC has also been widely applied to optimal system design problems such as (Shahrzad Faghih-Roohi and Ng, 2014; Xiaobin Wu and Goggans, 2012; Polasek, 2014). In MCMC-based optimization, the basic concept is to construct a Markov Chain with an equilibrium distribution that is appropriately mapped from the cost function of the problem to be solved. Therefore, samples generated based on this distribution represent different solutions of the original problem. It is known that heuristic methods have been well used for complex optimization problems to find sub-optimal solutions. For example, In (Marichelvam and Prabaharan, 2015; Mousavipour and Hojjati, 2014; Manimaran and Selladurai, 2014) the Particle Swarm Optimization (PSO) algorithms have been used to solve various practical problems including industrial scheduling, vehicle routing and supply chain related problems. In (D.K. Jana and Roy, 2013), GA was used to solve a fuzzy rough expected value multi-objective decision making model concerning a production inventory problem. In (J. Rezaeian and Arab, 2015), Multinode resource-constrained project scheduling problem (MRCPSP) is targeted with a GA-based meta-heuristic method. It can be observed that a common process in these methods is generating possible decisions and examining whether these decisions are appropriate. Similar to these heuristic methods, a MCMC method also generates samples that correspond to different solutions. However, the advantage of a MCMC-based method is that the sample (i.e. decision) generation and acceptance are based on the equilibrium probability distribution. Therefore, it has a better guidance for samples to evolve out of local optima such as in (Xiaobin Wu and Goggans, 2012). In this chapter, we design a two-layer MCMC-based method for the terminal location planning problem. We will show the effectiveness of the proposed MCMC-based method as compared to the Brute-force search in cases with manageable scales. We will also demonstrate that the method indeed provides a better solution as compared to an existing state-of-the-art method for the same problem (Sorensen et al., 2012) in the case of large random networks.

The rest of chapter is organized as follows. In section II, the intermodal network model and the problem formulation are described. In section III, the MCMC-based method is addressed. It consists of two layers of algorithms, where the lower-layer is a table-based heuristic routing method and the upper-layer is a MCMC-based location planning method. Section IV provides the simulation results for the performance evaluation and comparison. Finally, section V concludes the chapter.

## 3.2 Problem Description

We consider an intermodal transportation system consisting of two types of nodes, i.e., service nodes in "circle" and terminal nodes in "square", as shown in Fig. 3.1. For simplicity, they can be perceived as road and railway transportation modes, respectively. In such a system, services are offered between a certain number of origin-destination locations, such as represented by service nodes 1 to 9. In general, the direct delivery service using a specific carrier is very expensive and the number of direct deliveries is often limited. In order to improve the economic competitiveness, low-volume demands can be moved to a consolidation terminal, such as represented by nodes $A, B, C$, via the road mode. In these terminals, a large number of low-volume freight will be consolidated into high-volume flows that will be routed to other terminals through high-frequency, high-capacity services that could be operated in the railway mode. A large number of lower frequency services, often operated with smaller vehicles, are used between the terminals and the origin/destination nodes. Without the loss of generality, we also allow the customized direct service using paths between an origin node and a destination node that does not go through any consolidation terminals. Allowing these uni-modal services makes the terminal location planning problem different from the body of research in the hub location planning in (Crainic et al., 1989) where any transportation strictly follows "hub-and-spoke" organization and must go through hub terminals.

Since the terminal operation and the initial and final drayage move between the terminals and service nodes contribute significantly to the operational and transportation

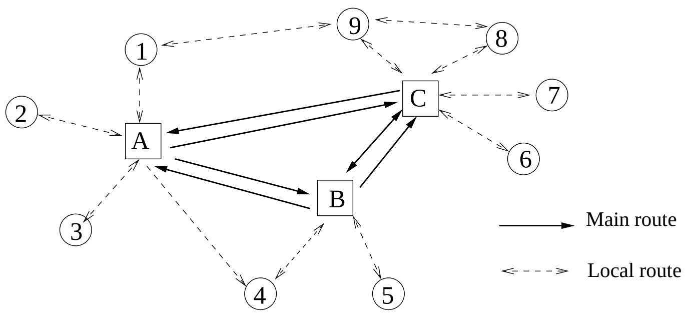

Figure 3.1. Intermodal transportation network with consolidation terminals.

cost, terminal location planning is critical for the success of an intermodal transportation system. The objective of the intermodal terminal location planning is to determine which terminals from a set of potential terminal locations will be used, and how to route the supply and demand of a set of customers through the network, via both uni- and intermodal transportation, so as to minimize the total cost.

To illustrate the research problem, we adopt the basic model proposed in (Arnold et al., 2004) and describe the transportation with a graph network. Let $I$ be the set of all origin-destination service pairs and $K$ the set of all potential terminal locations in the network. Each origin-destination pair $(i, j)$ has associated with it a positive and fixed amount $q_{ij}$ of goods that need to be transported (clearly $q_{ii} = 0$). The variable $x_{ij}$ represents the portion of the demand $q_{ij}$ transported uni-modally, whereas the set of variables $x_{ij}^{gk}$ relate to the portion of the demand $q_{ij}$ transported inter-modally using terminals $g, k \in K$. Let $c_{ij}^{gk}$ be the unit cost of transporting demand between $i$ and $j$ through terminals $g$ and $k$ and $c_{ij}$ be the unit cost of transporting demand directly from $i$ to $j$ without any intermediate intermodal operations. $c_{ij}^{gk}$ is generally much less than $c_{ij}$ due to = the high cost of direct transportation. For each potential terminal location $k \in K$, it has been associated with a positive and fixed capacity $C_k$, a fixed cost $F_k$ proportional to the capacity, and a decision variable $y_k$ which is "1" when terminal $k$ is open (i.e., selected in the planning) and "0" otherwise. Then the overall cost function is given as

$$J = \sum_{i,j:(i,j)\in I} \sum_{g,k\in K} c_{ij}^{gk} x_{ij}^{gk} + \sum_{i,j:(i,j)\in I} c_{ij}x_{ij} + \sum_{k\in K} F_k y_k. \tag{3.1}$$

There are three parts in equation (3.1). The first part is the cost from the path through intermediate terminals; the second part represents the direct transportation cost between two service nodes; and the third part is the operational cost for the open terminals including the cost of the construction, equipment and manpower. In equation (3.1), we can notice that the path through exactly two intermediate terminals is the only choice for intermodal transportation. In practice, terminal transportation can occur through multiple terminals. However, given a specific transportation network, the cost along the multiple terminals can always be summed as the cost between the initial and ending terminals.

This objective function represents the total transportation cost associated with all transportation flows within the network. Therefore, the terminal planning problem becomes to find the decision variables $y_k$, $x_{ij}$ and $x_{ij}^{gk}$ that minimize the function of $J$, subject to the following constraints:

$$x_{ij}^{gk} \le q_{ij}y_g, \qquad \forall g,k \in K, \forall (i,j) \in I \tag{3.2}$$

$$x_{ij}^{gk} \le q_{ij}y_k, \qquad \forall g,k \in K, \forall (i,j) \in I \tag{3.3}$$

$$\sum_{g,k\in K} x_{ij}^{gk} + x_{ij} = q_{ij}, \qquad \forall (i,j) \in I \tag{3.4}$$

$$\sum_{i,j:(i,j)\in I} \sum_{g\in K} x_{ij}^{gk} + \sum_{i,j:(i,j)\in I} \sum_{g\in K} x_{ij}^{kg} \le C_k, \quad \forall k \in K \tag{3.5}$$

$$x_{i,j} \ge 0,\ x_{ij}^{gk} \ge 0,\ x_{ij}^{kk} = 0,\ \forall (i,j) \in I, \forall g,k \in K \tag{3.6}$$

$$y_k \in \{0,1\} \quad \forall k \in K \tag{3.7}$$

Constraints (3.2.2) and (3.2.3) ensure that one flow can only go through those opened terminals. Constraint (3.2.4) shows that the sum of flows transported from intermodal and unimodal network must be equal to the overall demand associated with each origin-destination pair. Constraint (3.2.5) means that the overall flow going through a terminal cannot exceed the capacity of the terminal. Constraint (3.2.6) ensures that the amount of flow is non-negative and one flow cannot go through one terminal only. Constraint (3.2.7) means that one terminal is either selected (open) or not selected (closed).

Note that the above formulation also involves an underlying operation that simultaneously determines the optimal route paths for each service pair $(i, j)$, given a set of selected terminal locations. That is, $x_{ij}^{gk}$ and $x_{ij}$ are in fact related to the route paths associated with $y_k$ and other variables including $q_{ij}, F_k, C_k$, for $k \in K$. Therefore, two issues are involved in this problem:

- For a particular terminal planning, i.e., the open/close status of potential terminals, determine the optimal route paths for all service pairs simultaneously.
- Determine the set of the most appropriate terminals that minimizes the overall cost function.

These two issues are related to each other. Opening more terminals may provide more options to choose route paths, but causes additional infrastructure and operational cost for the terminals. The first issue appears similar to a routing problem such as finding the shortest path between a particular source and destination pair. However, when multiple service pairs exist simultaneously, finding the shortest path for one service pair could prevent some other service pairs from using certain paths and hence lead to high cost for these service pairs. In addition, when the number of terminals and service pairs grows, greedy algorithms, such as (Thomas H. Cormen and Stein, 1990), become less and less efficient. Let $b = |K|$ be the number of potential terminals. There are $2^b$ different possible planning. As $b$ increases, it is computational demanding to enumerate all the possible terminal planning cases and for each case to find the best route paths for all service pairs. To solve this complex problem, we design a heuristic MCMC-based method.

## 3.3 MCMC-based suboptimization methodology

MCMC (Gregory, 2005) is a method for sampling from probability distributions. It uses the "detailed balance" mechanism to construct a Markov chain that has a desired equilibrium distribution. After a short "burn-in" stage, the states of the Markov chain will represent the samples from the desired distribution. One merit of this method is that the samples visit all regions in the parameter space based on their contribution to the equilibrium distribution, rather than sticking with a specific attraction region.

To apply MCMC to the location planning problem, we first need to map the cost function into a probability function. This is represented as

$$P_S(\mathcal{D}|\Omega) = \frac{1}{Z_S} \exp\left[-\frac{J(\mathcal{D}|\Omega)}{S}\right] \tag{3.8}$$

where $S$ is a scaler used to adjust the shape of the probability function. $Z_S$ is the normalization constant to ensure that the function is a probability distribution. $J(\mathcal{D}|\Omega)$ is the cost function calculated with equation (1) for one set of decision variables $\mathcal{D}$, given system setup parameters $\Omega$. $\mathcal{D}$ includes a binary bit sequence $\boldsymbol{y} = \{y_k, k \in K\}$ indicating whether one potential terminal is open or closed, and the best route paths, i.e., $x_{ij}^{gk}$ and $x_{ij}, (i,j) \in I, g, k \in K$, for all service pairs. $\Omega$ is the set of known network parameters including the capacity and operation expense of each potential terminal, and the distance and unit cost of each path. Clearly a small system cost gives a large probability value in this inference/optimization problem.

The purpose is to find a specific decision set $\mathcal{D}$ that maximizes $P_S(\mathcal{D}|\Omega)$. The ability of MCMC to generate samples conforming with a well defined probability density function can be applied to solve the optimization problem by finding the chain state giving the highest probability value. We solve this problem by design a two-layer algorithm. The first is to find $x_{ij}, x_{ij}^{gk},\ (i,j) \in I,\ g, k \in K$ given $\boldsymbol{y}$, i.e., finding the route paths for all services pairs simultaneously, given a particular terminal location planning. The second is to find the optimal terminal location planning $\boldsymbol{y}$ that maximizes $P(D|\Omega)$.

### 3.3.1 Lower-layer: Table-based Heuristic Routing Method

This method is designed to find the shipping paths $x_{ij}, x_{ij}^{gk}$, given a specific location planning $\{y_k, k \in K\}$. For each service pair $(i, j)$, we first construct a routing information table $M_{ij}$, and then based on all these tables we find the final routing result for each service pair and store the information in a table $R_{ij}$

#### 3.3.1.1 Routing Information Table $M_{ij}$ and Final Routing Table $R_{ij}$

We assume that for each service pair $\{i, j\}$, any indirect path only uses two terminals, one connected to $i$ (denoted as $T_1$) and the other connected to $j$ (denoted as $T_2$). Given total $|I|$ service pairs, the routing information for each service pair $\{i, j\}$ is created individually and stored in a table $M_{ij}$ with the structure shown in Table 3.1.

Table 3.1. Data structure of Table $M_{ij}$ for service pair $\{i, j\}$.

| $i$ | $T_1$ | $T_2$ | $j$ | Unit Cost | Path Capacity | Prediction factor |
|---|---|---|---|---|---|---|

Each row of this table represents a possible path between $\{i, j\}$. If the values of $T_1$ and $T_2$ are "0"s, it represents the direct path. For a terminal planning with $b$ open terminals, there are $\binom{b}{2} = b(b-1)/2$ possible indirect paths. These $b(b-1)/2+1$ rows are listed in the ascending order based on the "Unit Cost" which denotes the cost to transport one unit of load along this path. The "Path Capacity" shows the maximum load that can be transported through this path. For an indirect path, this is upper-bounded by the minimum capacity value of the starting and the ending terminals.

One $M_{ij}$ is created independently without the knowledge of other service pairs. When these service pairs coexist, simultaneously routing is needed. The final route paths for each service pair $(i, j)$ are stored in a table $R_{ij}$, with the same structure information as in $M_{ij}$ except that the "Path Capacity" now shows the amount of load that will be transported through that path for service pair $(i, j)$. Apparently, finding $R_{ij}$ should consider $M_{ij}$ of all service pairs. The method to obtain $R_{ij}$ is as follows.

#### 3.3.1.2 Route-Assignment Method with a Prediction Factor

Assume there are $|I|$ number of service pairs. Some of which may compete for the use of certain terminals. A natural question is which service pair has the priority to use a particular path. The key issue in the routing design is that we need to avoid the situation that one service pair occupies most capacity of one terminal that is also demanded by other service pairs. To handle this problem, we propose to make use of a prediction factor $Pred_{(ij)}$ that is defined as the inverse of the ratio of unit cost of the current path (row) and the next path (row) for service pair $(i,j)$ in $M_{ij}$. For example, after sorting the $M_{ij}$ table by the "Unit Cost" in ascending order, i.e., the path with the lowest "Unit Cost" is in the first row and the cost is denoted as $c_{ij}^{1st}$ while the second lowest is $c_{ij}^{2nd}$, then $Pred_{(ij)} = c_{ij}^{2nd}/c_{ij}^{1st}$ for the first row. This factor captures the information that if we have missed assigning the current path best for service $(i,j)$, how much difference in scaling it would be if the service has to use the next best path for transportation. As the path assigning procedure continues, this prediction factor for each service pair also varies as different paths will be considered.

The heuristic routing method is as follows. For each service pair $(i,j)$, starting from the top row of $M_{ij}$, the process proceeds by continuously assigning a path and the number of unit loads transported through that path to a specific service pair. For a particular service pair $(i,j)$, the probability of being chosen for the current path is determined by $\frac{Pred_{(i,j)}}{\sum Pred_{(i,j)}}$. The denominator is the sum of the prediction factors of all the service pairs that may also use the same path for transportation. To further improve the fairness, the number of units of load in one assignment for a particular path may be up-bounded. For example, in one assignment, the selected service $(i,j)$ can only use the minimum value of its remaining service demands and one-tenth of the maximum capacity of the current path. Intuitively, services with more urgency to use a particular path are likely to be assigned with more portion of that path. Once one assignment is completed for service $(i,j)$, the routing information, such as the path index $m$ and the number of loads, will be stored in $R_{ij}$ in a row. In the meantime, the path information will be updated for all $M_{ij}$. That is, the capacity of path $m$ in all $M_{ij}$ should be reduced accordingly. A row with a path capacity reduced to zero is referred to as an empty row, meaning that the path is no longer available. Routing process for each service pair $(i, j)$ ends when the sum of loads along different paths in $R_{ij}$ equals $q_{ij}$.

Using the prediction variable helps to solve the route competition problem. If a service pair with a high prediction factor has no conflict with other service, it can occupy the path alone. If one path is need by multiple service pairs, every service pair has a chance to use this path due to the probabilistic selection and the granularity in path assignment. The pseudo-code for one-pass of route assignment and table-update is given in Algorithm 1.

**Algorithm 1 Lower-layer: Heuristic Routing design**

```
Input:
$b$ (the number of total potential terminals)
$I$ (the set of the total service pairs)
$C$ (the set of terminal capacity)
Output: $R_{ij}$ for each $(i, j, q_{ij})$ in $I$
While $(i, j, q_{ij}) \in I$ do
    Generate table $M_{ij}$ (with ordered rows);
    Calculate prediction value $Pred_{(i,j)}$ ;
end
While not all $(i, j, q_{ij})$ are routed do
    1. Find the first non empty row $m$ of $M_{ij}$ ;
    2. Obtain $Pred_{(i,j)}$ from $M_{ij}$;
    3. Select $(i, j, q_{ij}) \in I$ with probability $Pred_{(i,j)} / \sum Pred_{(i,j)}$;
    4. Calculate current ship load $= Min(0.1 * C_m^{max}, q_{ij})$;
    5. Record $m$, current ship load in $R_{ij}$;
    6. Reduce $q_{ij}$ by current ship load;
    7. Update path capacity in all $M_{ij}$ ;
end
```

$q_{ij}$ in $(i, j, q_{ij})$ is the quantity of load needs to be shipped between $(i, j)$. The first "while" loop is to generate $M_{ij}$ that stores all routing information and path costs between terminal $(i, j)$. For a path with intermediate terminals, such as $i \rightarrow g \rightarrow k \rightarrow j$, "Unit Cost" is calculated as

$$\begin{aligned} c_{ij}^{gk} &= c_{ig} + c_{gk} + c_{kj} \\ &= Dist(i,g)Z_1 + Dist(g,k)Z_2 + Dist(k,j)Z_1 \end{aligned} \tag{3.9}$$

where $Z_1, Z_2$ are values in "cost per unit load per mile" for each mode of transportation. Due to the nature of intermodal transportation, $c_{ij}^{gk}$ is much smaller than the unit cost $c_{ij}$ used in a direct transportation that is calculated as $Dist(i,j)Z_1$.

The second loop in Algorithm 1 is to find the route paths for all services simultaneously. Once a single assignment is completed, the algorithm iterates for next service pair. In each assignment, it always starts from the first non-empty row in a $M_{ij}$ table. For the fairness of using particular path, one service should not occupy all the transportation capacity of a path that might be important to other services too. In step 3 of Algorithm 1 we set the upper limit for each route assignment as one-tenth of the path's maximum capacity. The actual load assignment is $Min(q_{ij}, 0.1C_m^{max})$, $m$ is the current best path of $(i,j)$. This upper limit can be set by the user, while noting that the smaller the limit is, the more time the process needs. In step 6, after one path and load assignment, the available capacity of intermediate terminals needs to be updated. This is critical because any terminal that runs out of capacity will be no longer used. As a result, some paths in $M_{ij}$ associated with this terminal will no longer be valid and need to be eliminated. When updating all $M_{ij}$ tables in step 7, these rows are considered as empty.

### 3.3.2 Upper-layer: MCMC-based Optimal Planning

Given the service and network parameters $\Omega$ and the terminal open/close status $\boldsymbol{y} \in \mathcal{D}$, we can find the routing information and then calculate the cost function $J(\mathcal{D}|\Omega)$ in the lower-layer. The problem remains to find the best $\boldsymbol{y}$ that gives the minimum overall cost. Assume that there are total $b = |K|$ possible terminals. Therefore, there are a total of $2^b$ possible ways of planning. Exhaustive search is possible only for a very small number of terminals. When $b$ increases, heuristic methods must be used. In this work, this problem is solved in the upper-layer with MCMC.

#### 3.3.2.1 Simulated Annealing

Equation (3.8) shows the mapping from the cost to a probability distribution. Let $\boldsymbol{y}_1$ be the current location planning, i.e., the open/close status of the $b$ terminals. Let $J(\boldsymbol{y}_1|\Omega)$ be the total cost based on this location planning and the routing from Algorithm 1. Suppose that $\boldsymbol{y}_2$ is a new $b$-bit-sequence (i.e., a new location planning) generated randomly. Then,

$$\frac{P_S(\boldsymbol{y}_2|\Omega)}{P_S(\boldsymbol{y}_1|\Omega)} = \exp\left[-\frac{J(\boldsymbol{y}_2|\Omega) - J(\boldsymbol{y}_1|\Omega)}{S}\right]. \tag{3.10}$$

Whether this vector $\boldsymbol{y}_2$ will be selected as a new sample of the stationary probability of MCMC is determined by the probability

$$Min\left\{1, \exp\left[-\frac{J(\boldsymbol{y}_2|\Omega) - J(\boldsymbol{y}_1|\Omega)}{S}\right]\right\}. \tag{3.11}$$

This process is termed as the Metropolis process in (Metropolis et al., 1953) in MCMC. It is well known that with this process, the samples of the Markov Chain follow the distribution function of equation (3.8). With the mapping between the distribution function and the cost function, we can also find that

1. If $J(\boldsymbol{y}_2|\Omega) \to \infty$, then equation (3.10) goes to 0. In such a case, the acceptable probability for $\boldsymbol{y}_2$ tends to be 0, which means that $\boldsymbol{y}_2$ has a low chance to be accepted as a new sample in MCMC.

2. On the other hand, that $J(\boldsymbol{y}_2|\Omega) \to 0$ means that the cost of $\boldsymbol{y}_2$ is negligible. Since $\exp[\frac{J(\mathcal{D}_1|\Omega)}{S}]$ is always larger than 1, $\boldsymbol{y}_2$ will always be selected.

To apply this MCMC to heuristic optimization problems, the scaler $S$ can often be modeled as a function of time $\tau(t)$ to speed up the optimization convergence, and this process is termed as simulated annealing in (Bertsimas and Tsitsiklis, 1993). Hence, equation (3.11) becomes

$$Min\left\{1, \exp\left[-\frac{J(\boldsymbol{y}_2, t|\Omega) - J(\boldsymbol{y}_1, t|\Omega)}{\tau(t)}\right]\right\}. \tag{3.12}$$

The use of parameter $\tau(t)$ came from physics mimicking the behavior of atoms that move more rapidly when the temperature is high, and slowly when the temperature is low. This strategy is designed to let a state easily jump out of a local optimum when the temperature is high, but focus on the neighborhood of the found optimal point when the temperature is low. The temperature function is a non-increasing function, which represents a cooling-down process. $\tau(0)$ is usually set as a high value in the beginning of simulation to make sure that a new state has high probability to be accepted. In this stage, system may reach more diverse states based on the probability contribution. When $\tau(t)$ approaches 0, the probability becomes very small and the system is reluctant to accept a new state $j$, which is often described as a "Frozen" stage. The speed of decreasing of temperature is usually slow to give the system enough time to be "trapped" in the low cost area. In our simulation, we reduce the temperature after each new sample is accepted.

#### 3.3.2.2 Slice Sampling

Simulated annealing in equation (3.12) determines whether to accept a new sample but does not concern how to generate the new samples used for the acceptance test. In this work, we integrate simulated annealing with slice sampling. Slice sampling is based on the idea that to sample a random variable from a distribution, one can sample uniformly from the region under the graph of its density function (Kirkpatrick et al., 1983). Process of slice sampling with one variable is shown in Fig. 3.2 and described as follows.

1. evaluate $P[x(t) = i]$

2. draw a vertical coordinate,

   $u \sim Uniform(0, P[x(t) = i])$

```
3. create a neighbor space enclosing current state $i$,
       $i \in (I_{left}, I_{right})$
4. start loop {
5.     draw $j$ from neighbor space, $j \in (I_{left}, I_{right})$
6.     evaluate $P[x(t+1) = j]$
7.     if $P[x(t+1) = j] \geq u$, break out loop,
           the next state $x(t+1) = j$ accepted
8.     else modify the neighbor space and repeat loop
9. end loop }
```

$P[x(t) = i]$ is the probability function. $x(t) = i$ represents that at time $t$, the state of system is $i$. The time is indexed as $t = \{1, 2, 3, \ldots\}$.

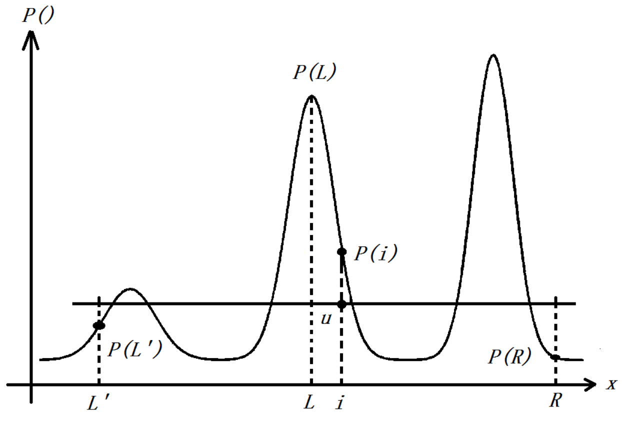

Figure 3.2. Slice sampling process. $[L, R]$ is the neighbor space enclosing current state $i$. Left end of the interval $P(L)$ is evaluated and is known to be greater than $u$, so a step to the left of size $w$ is made, which make left end to $L'$. Check $P(L')$, found $P(L')$ is smaller than $u$, so stop and step out. Also at the right end of the interval $P(R)$ is evaluated and is smaller than $u$, so no stepping out should be done. When the above process it done, $P$ of both ends should be smaller than $u$. Since the neighbor space is decided as $[L', R]$, then the next state candidate will be picked randomly in the interval $[L', R]$.

As a result, the proposed method is to use slice sampling to generate new samples while use simulated annealing to improve the convergence speed in optimization. Compared with a normal Metropolis process, slice sampling has variable step size in random walk and can better characterize the local property of the underneath probability function, and therefore promptly generate acceptable new samples. Slice sampling can be applied to wherever the Metropolis method can be use. The only requirement is that the target density function $P(\boldsymbol{y} \in \mathcal{D}, t|\Omega)$ can be evaluated. In this work, due to the binary representation of $\boldsymbol{y}$, the implementation idea of multidimensional slicing sampling in (Skilling and MacKay, 2003) can be directly modified and adopted here. The description of slice sampling to generate a new sample at time $t$ is as in Algorithm 2.

**Algorithm 2 Upper-layer: Generating a new terminal planning sample.**

```
Input: $b$, $\boldsymbol{y}$, $P(\boldsymbol{y}|\Omega)$
Output: $\boldsymbol{y}'$
$Y = uniform(0, P(\boldsymbol{y}, t|\Omega)$;
$U = randbits(2^b)$;
Set $L$ to a value $L \leq b$;
Randomly generate $\boldsymbol{y}'$;
While $(P(\boldsymbol{y}', t|\Omega) < Y)\&(L > 0)$ do
    1. $N = randbits(L)$ ;
    2. $\boldsymbol{y}' = ((\boldsymbol{y} - U) \oplus N) + U$ ;
    3. $L = L - 1$ ;
end
Increase $t$;
```

In the algorithm, steps within the "while" loop automatically take c

$P(\boldsymbol{y}, t|\Omega)$ is the desired distribution function based on $J(\boldsymbol{y}|\Omega)$ defined by Simulated Annealing. $P(y, t|\Omega) = \exp\left[-\frac{J(y|\Omega)}{\tau(t)}\right]$ (Goldstein and Waterman, 1988). $\tau(t)$ decreases by every new state $\boldsymbol{y}'$ found. Operations in binary sequences are all subjected to a final modular operation with $2^b$. "-", "+" and "$\oplus$" between two binary vectors are subtraction, sum and "exclusive-or" operations followed with *mod* of $2^b$.

## 3.4 Experiments and Performance Evaluation

We test the proposed method with computer simulations. We first verify the effectiveness of the routing algorithm in a simple network setup and then test the overall heuristic MCMC-based planning in randomly generated network scenarios. Comparison with one existing method is also provided.

### 3.4.1 Routing in an Equilateral Triangle Network

For simplicity, we design a special case to verify the effectiveness of the routing method with prediction factors. We assume there are three service nodes, denoted as nodes 1,2 and 3, and three terminals denoted by nodes 4, 5 and 6. There are three service pairs as shown in Table 3.2. The capacity of terminals are set as 50 each. As shown in Fig. 3.3, the three service nodes are in the form of equilateral triangles, and so are the three potential terminals.

Table 3.2. Service pairs

| origin ($i$) | destination ($j$) | units of load ($q_{ij}$) |
|---|---|---|
| 1 | 2 | 100 |
| 2 | 3 | 100 |
| 3 | 1 | 100 |

The routing results are shown in Fig. 3.3 and Table 3.3 with the use of prediction factor, and are shown in Fig. 3.4 and Table 3.4 without the use of prediction factor.

Table 3.3. With the predication factor. Routing path of Fig 3.3. Each service use 25 units on potential terminal. Potential terminals $4, 5, 6$ are fully used.

| Origin ($i$) | $T_1$ | $T_2$ | Dest. ($j$) | Units of Load |
|---|---|---|---|---|
| 1 | 4 | 5 | 2 | 25 |
| 1 | 0 | 0 | 2 | 75 |
| 2 | 5 | 6 | 3 | 25 |
| 2 | 0 | 0 | 3 | 75 |
| 3 | 6 | 4 | 1 | 25 |
| 3 | 0 | 0 | 1 | 75 |

Capacity of each of the three potential terminals has been set with 50 units of loads, however, each service has amount of 100 to be shipped. This means that we cannot ship

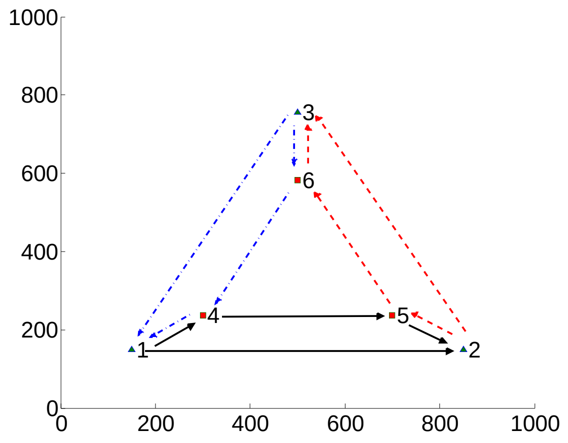

Figure 3.3. With the predication factors, the routing paths of service pairs $(1 \rightarrow 2)$, $(2 \rightarrow 3)$and $(3 \rightarrow 1)$. The three services have the same distance and have the same chance to use the potential terminals. Their routing paths are shown in Table 3.3.

Table 3.4. Without predication factor. The routing paths of Fig. 3.4. The first service pair $(1 \rightarrow 2)$ used all capacity of potential terminals 4 and 5. This results in terminal 6 not being used and large costs for other service pairs.

| Origin $(i)$ | $T_1$ | $T_2$ | Dest. $(j)$ | Units of Load |
|---|---|---|---|---|
| 1 | 4 | 5 | 2 | 50 |
| 1 | 0 | 0 | 2 | 50 |
| 2 | 0 | 0 | 3 | 100 |
| 3 | 0 | 0 | 1 | 100 |

all loads through the potential terminals, and there must be some loads shipped by direct paths. The result in Fig. 3.3 and Table 3.3 clearly matches the intuition that the intermodal transportation should be used and neither service pair can occupy a terminal capacity alone. It can be noted that since the positions of these nodes are in a symmetric pattern, one service pair should not use the terminal capacity more than other service pairs. While without the fairness consideration by using the prediction factors, supporting one service most efficiently could leave other service pairs in a pessimistic position as shown in Fig. 3.4 and in Table 3.4. The rows with $T_1 = T_2 = 0$ represent the direct paths.

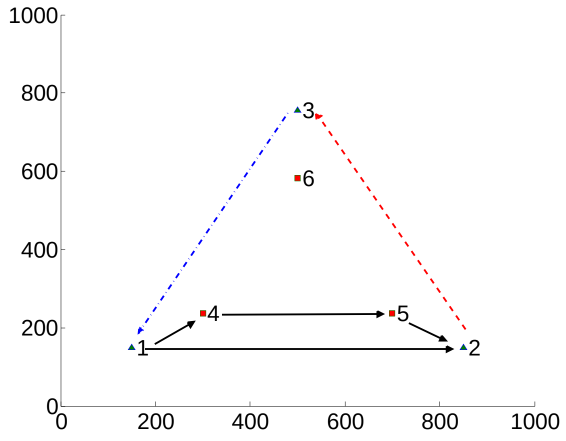

Figure 3.4. Without the predication factor. Terminal 6 has not been used. In this case, $(1 \rightarrow 2)$ uses all the capacity of terminals $4, 5$. Clearly this is not an efficient way to route the shipping paths.

### 3.4.2 Random Networks

In this case, we randomly generate the service nodes and potential terminal nodes in a two-dimensional plane, except that we set all potential terminals be in the central part of the map, and set the service nodes surrendering them. Service loads for each service pair and the capacity of each potential terminal are also generated randomly. The operational cost of a terminal is set proportional to its capacity. The transportation cost of a path between two nodes depends on the distance and transportation mode, on the premise that any two terminals are connected. The direct path is also included but with a much high path cost. Transportation distance between any two nodes is assumed to be the direct line distance between the two nodes.

Fig. 3.5 shows an example of the random map with $x, y$ axes ranging from 0 to 1000. The services nodes are denoted by the triangle shape while the terminal nodes are denoted by the square shape. All nodes are numbered. Each terminal is assigned with a capacity, operational cost together with its location coordinates. In the Fig. 3.5, we assume there are 4 service pairs, with the loads of service randomly generated in the range of 15 to 30 for each

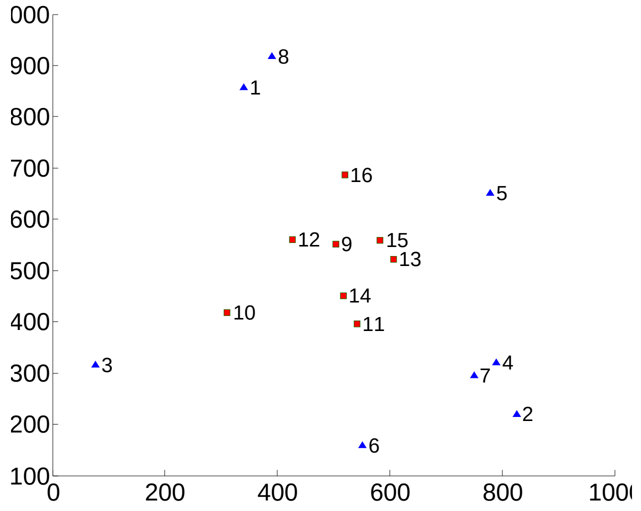

Figure 3.5. Map contains service nodes and potential terminals. 1-8 are service nodes and 9-16 are potential terminals

pair. The service pairs are shown in Table 3.5.

Table 3.5. Service pairs

| Origin $(i)$ | Destination $(j)$ | Units of Load |
|---|---|---|
| 01 | 02 | 35 |
| 03 | 04 | 19 |
| 05 | 06 | 19 |
| 07 | 08 | 18 |

Each routing path has an associate cost. The lower cost the better. Since paths using terminals are strongly suggested, we assume any path through two intermediate potential terminals would have less total cost. In simulation, we assume that it has the half cost compared with the cost that uses the direct path with the same distance. Apparently, other assumptions of the cost of indirect-direct cost can also be used, which may change the final results of the terminal planning. With the proposed heuristic MCMC method, the routing information and the final cost are shown in Table 3.6 for Fig. 3.5. As can be found, different service pairs may use different numbers of route paths. For example, service pair $(1, 2)$ uses both direct and indirect paths while pair $(3, 4)$ only uses the intermodal transportation. It can be noted that some potential terminals are used by different service pairs. Service pairs

$(1, 2)$ and $(7, 8)$ share terminal 16; service pairs $(7, 8)$ and $(3, 4)$ share terminal 13. This suggests some terminals are important to multiple service pairs. The route paths are also shown in Fig. 3.6.

Table 3.6. In the final planning, all potential terminals are all selected except terminal 11. The total cost is 7098, which includes the terminal operational cost 2930 and shipping cost 4168.

| Origin ($i$) | $T_1$ | $T_2$ | Dest. ($j$) | Amount | ship cost |
|---|---|---|---|---|---|
| 3 | 10 | 14 | 4 | 13 | 498.0 |
| 3 | 14 | 13 | 4 | 6 | 254.3 |
| 7 | 13 | 16 | 8 | 6 | 215.8 |
| 1 | 16 | 15 | 2 | 11 | 443.6 |
| 7 | 14 | 9 | 8 | 8 | 306.3 |
| 1 | 9 | 15 | 2 | 2 | 84.2 |
| 1 | 12 | 9 | 2 | 19 | 806.7 |
| 7 | 0 | 0 | 8 | 4 | 287.6 |
| 5 | 0 | 0 | 6 | 19 | 1031.2 |
| 1 | 0 | 0 | 2 | 3 | 240.2 |

We also studied the effect of changing of terminal operational cost $F_k$ on the planning results. Fig. 3.7 and Table 3.7 show the results when $F_k$ is increased by 80% compared to the previous example. With the increase of $F_k$, the number of terminals selected decreases, which matches intuition. In this situation, the total cost is slightly increased. Table 3.8 shows the open-and-close status of the terminals in two different cases.

### 3.4.3 Effectiveness Compared with Brute Force Method

In this part, we verify the effectiveness of the proposed MCMC method as compared to the brute-force method. Both methods use the same lower-layer routing algorithm. We denote the network setup as "$IXKY$" which stands for having $X$ service pairs and $Y$ potential terminals in the network. The brute-force method calculates all possible open/close status of terminals in order to find the global optimal solution. In previous simulation, the lower-layer uses the prediction factor which is implemented with a wheeling function (WF). Due to the randomness in routing and MCMC, the same terminal location planning may give

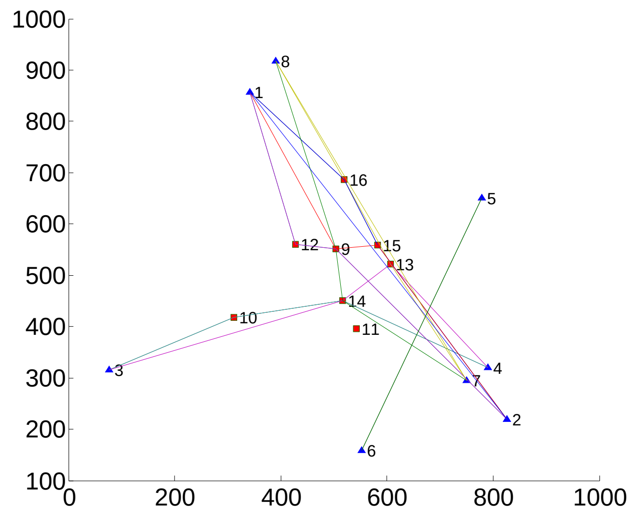

Figure 3.6. Paths and terminals after planning. Terminal $12, 13, 14, 15$ have been fully used. $F_k$ depends on terminal capacity. Terminals with large capacity associate with large operational cost.

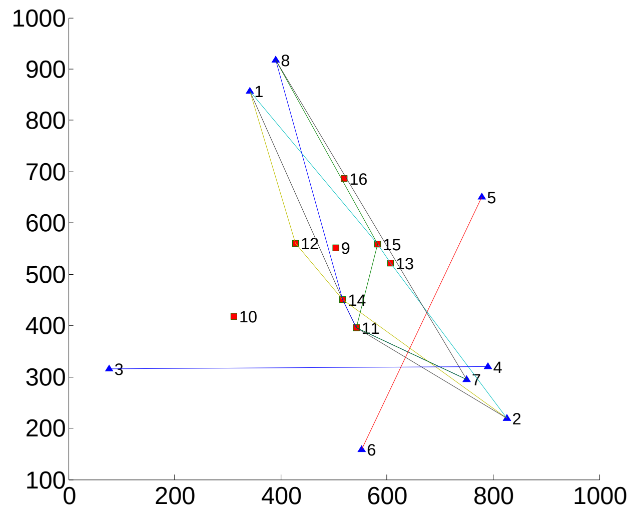

Figure 3.7. Increased 80% of each termial cost $F_k$. Since the high cost $F_k$, the terminal $9, 10, 16$ are closed in order to save operation cost. The terminals $11, 12, 13, 14, 15$ are open and shared by multiple services.

slightly different cost values in different runs of the program. Therefore, we test a number of randomly generated networks for one "$IXKY$" setup.

In table 3.9, for each of $I8K8$ and $I10K10$ cases, 10 random network scenarios are tested. It can be found that the MCMC solution is very close to optimal solution from the

Table 3.7. Final planning when $F_k$ are increased by 80%. 5 out of 8 terminals are open. The total cost is 9332.8, which includes the operation cost 4374 and transportation cost 4958.8. If we use the planning obtained in table 3.6, the total cost would be 9441, which includes the operation cost 5274 and transportation cost 4168.

| Origin ($i$) | $T_1$ | $T_2$ | Dest. ($j$) | Amount | ship cost |
|---|---|---|---|---|---|
| 1 | 15 | 13 | 2 | 12 | 480.6 |
| 1 | 12 | 14 | 2 | 19 | 794.7 |
| 1 | 14 | 11 | 2 | 4 | 167.4 |
| 7 | 11 | 14 | 8 | 4 | 155.1 |
| 7 | 11 | 15 | 8 | 1 | 40.3 |
| 5 | 0 | 0 | 6 | 19 | 1031.2 |
| 7 | 0 | 0 | 8 | 13 | 934.7 |
| 3 | 0 | 0 | 4 | 19 | 1354.7 |

Table 3.8. Terminal status comparison between table 3.6 and 3.7 based on different $F_k$. In the case of increased $F_k$, the number of closed terminals is also increased to avoid extra cost.

| Terminal index | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|
| $y_k$ of normal $F_k$ | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| $y_k$ of increased $F_k$ | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| Capacity | 30 | 13 | 10 | 19 | 12 | 27 | 13 | 17 |

brute-force method in each case. To further demonstrate the benefit of using prediction factor, the results of the Brute Force search without the use of prediction factor in the lower-layer is also included.

Table 3.9. Three methods are compared in order to verify if using the wheeling function (WF) could result better routing plan. In this case terminals would be share by multiple shipping flows based on predication factors.

| | | I8K8 | I10K10 |
|---|---|---|---|
| Brute Force w/ WF | Max | 10470 | 12165 |
| | Min | 7623 | 8511 |
| | Avg | 9112.7 | 10396.1 |
| Brute Force w/o WF | Max | 10639 | 12328 |
| | Min | 7676 | 8526 |
| | Avg | 9225.2 | 10502.1 |
| Avg Difference | | -1.2 % | -1.0 % |
| MCMC w/ WF | Max | 10473 | 12203 |
| | Min | 7623 | 8511 |
| | Avg | 9130.3 | 10420.9 |
| Avg Difference | | -0.19 % | -0.24 % |

### 3.4.4 Comparison with an Existing Method

Two recently developed methods for terminal location problems were reported in (Sorensen et al., 2012), termed as GRASP ( Greedy Randomized Adaptive Search Procedure) with Heuristic evaluation procedure (HEP), and ABHC (Attribute Based Hill Climber) with HEP). In both methods, HEP is used as the lower-layer algorithm and either GRASP or ABHC is used as the upper-layer algorithm. Since comparison already has been made between GRASP and ABHC that shows similar performance (Sorensen et al., 2012), we compare the performance of the MCM-based method with GRASP.

The GRASP is a meta-heuristic method appropriate for combinatorial optimization problem. A GRASP algorithm usually consists of two steps: the construction layer to generate a feasible greedy randomized solution and the local search layer to improve the solution. In the construction layer, a restricted candidate list (RCL) is usually set up to contain candidate. The candidates are ranked by order, and chosen by the construction layer. The local searching layer starts from the construction layer solution and finds a local optimal solution.

Table 3.10. Comparison for cases $I16K16$, $I20K20$, each with 10 independent network scenarios. GRASP and MCMC used the same under-layer method presented in the chapter, i.e., table-based heuristic method with prediction factor. The average improvement in percentage of cost is listed in for MCMC-based method.

| | | I16K16 | I20K20 |
|---|---|---|---|
| GRASP | Max | 15504 | 20319 |
| | Min | 12512 | 13462 |
| | Avg | 13542.3 | 16900.4 |
| MCMC | Max | 15967 | 20688 |
| | Min | 12140 | 13067 |
| | Avg | 13394.2 | 16771.2 |
| Avg Improvement | | 1.1 % | 0.8 % |

HEP in the lower-layer is similar to our Algorithm 1 but without the use of the prediction factor. Because HEP has little consideration of the terminal competition for all service pairs, we first use our lower-layer algorithm and test the performance of GRASP and

MCMC in the location planning. Table 3.10 compares MCMC-based method and GRASP-based method for randomly generated network for cases of $I16K16$, $I20K20$. 10 independent random network scenarios are solved for each case. It can be observed that the MCMC-based method has better performance in each example and hence demonstrates the advantage of using distribution guided search in heuristic optimization process.

Next, we compare GRASP-HEP with the proposed heuristic MCMC method for four different cases of $I16K16$, $I20K12$, $I28K12$, $I30K20$. In table 3.11, GRASP combined HEP is compared with MCMC combined the proposed routing algorithm. The heuristic MCMC-based method again shows better performance.

Table 3.11. Comparison for the cases of $I16K16$, $I20K12$, $I28K12$, $I30K20$. Both GRASP-HEP and heuristic MCMC are applied to each map to make comparison. The max, min, and average costs are listed.

| | | I16K16 | I20K12 |
|---|---|---|---|
| GRASP-HEP | Max | 19051 | 21649 |
| | Min | 11946 | 16597 |
| | Avg | 15591.4 | 18947.8 |
| MCMC w/ WF | Max | 16327 | 20912 |
| | Min | 10656 | 14767 |
| | Avg | 13897.9 | 17326.2 |
| Avg Improvement | | 12.2 % | 9.4 % |

| | | I28K12 | I30K20 |
|---|---|---|---|
| GRASP-HEP | Max | 30005 | 33597 |
| | Min | 18298 | 22503 |
| | Avg | 26348.1 | 28055.2 |
| MCMC w/ WF | Max | 28267 | 32467 |
| | Min | 17413 | 21248 |
| | Avg | 24592.2 | 25749.1 |
| Avg Improvement | | 7.1 % | 9.0 % |

## 3.5 Conclusion

The contribution of this chapter is development of a two-layer heuristic MCMC-based method to solve the terminal location planning problem in intermodal transportation systems. The lower-layer is a heuristic routing design for multiple service pairs given a particular planning, where the prediction factor is used for fairness in path allocation so that a much needed terminal could be fairly shared by multiple service pairs. In the upper-layer, a MCMC-based method was presented where simulated annealing and slice sampling are integrated for the search of the optimal location planning. While slice sampling can produce new samples according to the underlying stationary probability distribution of the Markov Chain, simulated annealing decides the acceptance of the new samples so that the optimization process could jump out of the local optima and converge speedily. The advantage of the proposed method has been demonstrated through comparison with both the brute-force search and an existing method.

The work of this chapter demonstrates that MCMC, as a probability inference method, could be used to solve complex optimization problems with high dimensional searching spaces. This success comes from the ability of MCMC to generate samples (i.e., solutions) based on a stationary distribution function mapped from the cost function of the problem. In the future, more efficient sampling methods in (Skilling, 2006), such as nested sampling, could be explored for further improvement in solving optimization problems.

# CHAPTER 4

# CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we studied the behavior of Markov Chain Monte Carlo method and its application in different field. In the first few chapters, we introduced general idea of Markov Chain Monte Carlo Method.

Then in the first application, image transmission over time varying channels is considered. The varying channel, characterized by its probability distribution, makes the optimization problem of finding the best rate allocation very complicated. A MCMC method is proposed to solve this problem. It has been shown that the method can generate near-optimal solutions with low complexity. It also provides an overall picture of the distribution of distortion versus the rate allocation. In addition, it has been shown that the design considering the channel distribution performs better than the design considering the mean value of the channel.

In the second application, we have developed a MCMC-based two-layer method for the location planning problem in multi-mode transportation systems. The upper-layer is MCMC-based method to deal with terminal location problem in the intermodal transportation. The lower-layer is a table-based optimal routing method in order to calculate cost of multiple service pairs given a particular planning. In the lower-layer, the prediction factor is used to let terminals shared by multiple service pairs. In the upper-layer, a MCMC-based method is using both simulated annealing and slice sampling in order to search the optimal location planning. The result is compared with the Brute force method to verify the correctness, and an existing method (GRASP-HEP) to verify the advantage of performance.

The work of this thesis has demonstrated that a MCMC method, based on both slice sampling and simulated annealing, can be successfully applied to practical optimization problems with high-dimensional searching space. There are works that can be further researched in the future. For example, more advanced sampling method, such as nested sampling, could be used to further improve the performance. In addition, mapping between the cost function and targeted stationary distribution in Markov chain could be further investigated.

# BIBLIOGRAPHY

# BIBLIOGRAPHY

Arnold, P., D. Peeters, and I. Thomas (2004), Modelling a rail/road intermodal transportation system, *Transportation Research Part E: Logistics and Transportation Review*, 40(3), 255–270.

Arnold P, T. I. M. H., Peeters D (2001), Pour une localisation optimale des centres de transbordement intermodaux entre rseaux de transport: formulation et extensions, *The Canadian Geographer*, 45, 427–436.

B.A., B. (2002), Robust image transmission using jpeg2000 and turbo-codes, *Signal Processing Letters, IEEE*, 9(6), 117–119.

Bertsimas, D., and J. Tsitsiklis (1993), Simulated annealing, *Statistical Science*, 8(1), 10–15.

Bornstein, C., and H. Azlan (1998), The use of reduction tests and simulated annealing for the capacitated plant location problem, *Location Science*, 6, 67–81.

Cao, L. (2007a), Joint Source and Channel Coding for Image Transmission over Time Varying Channels, in *Communications, 2007. ICC '07. IEEE International Conference on.*

Cao, L. (2007b), On the unequal error protection for progressive image transmission, *IEEE Transactions on Image Processing*, 16, 2384–2388.

Chande, V., and N. Farvardin (2000), Progressive transmission of image over memoryless noisy channels, *IEEE Journal on Selected Areas in Communications*, 18, 850–860.

Crainic, T., P. Dejax, and L. Delorme (1989), Models for multimode multicommodity location problems with interdepot balancing requirements, *Annals of Operations Research*, 18, 279–302.

D.K. Jana, K. M., and T. Roy (2013), Multi-objective imperfect production inventory model in fuzzy rough environment via genetic algorithm, *International Journal of Operational Research*, 18(4), 365–385.

Filho, V., and R. Galvao (1998), A tabu search heuristic for the concentrator location problem, *Location Science*, 6(2), 189–209.

Gilks, W., S. Richardson, and D. Spiegelhalter (2003), *Markov Chain Monte Carlo in practice*, Chapman & Hall.

Goldstein, L., and M. Waterman (1988), Neighborhood size in the simulated annealing algorithm, *American Journal of Mathematical and Management Sciences*, 8(3), 409–423.

Gregory, P. (2005), *Bayesian Logical Data Analysis for the Physical Sciences*, Cambridge University Press.

Hajek, B. (1988), Cooling schedules for optimal annealing, *Mathematics of Operations Research*, 1(2).

Ishfaq, R., and C. Sox (2011), Hub location-allocation in intermodal logistic networks, *European Journal of Operational Research*, 210(2), 213–230.

J. Rezaeian, S. M., Farham Soleimani, and A. Arab (2015), Using a meta-heuristic algorithm for solving the multi-mode resource-constrained project scheduling problem, *International Journal of Operational Research*, 24(1), 1–16.

Jaramillo, J., J. Bhadury, and R. Batta (2002), On the use of genetic algorithms to solve location problems, *Computers and Operations Research*, 29(6), 761–779.

Kirkpatrick, S., C. D. Gelatt, and J. M. P. Vecchi (1983), Optimization by simulated annealing, *Science*, 220(4598).

Macharis, C., and Y. Bontekoning (2004), Opportunities for OR in intermodal freight transport research: A review, *European Journal of Operational Research*, 153, 400–416.

MacKay, D. J. (2003), *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press.

Manimaran, P., and V. Selladurai (2014), Particle swarm optimisation for multi stage supply chain network associated with fixed charges, *International Journal of Operational Research*, 21(1), 100–122.

Marichelvam, M., and T. Prabaharan (2015), Solving realistic industrial scheduling problems using a multi-objective improved hybrid particle swarm optimisation algorithm, *International Journal of Operational Research*, 23(1), 94–129.

Meinert, T., A. Youngblood, G. Taylor, and H. Taha (1998), Simulation of the railway component of intermodal transportation, *Tech. rep.*, Arkansas University, Fayetteville, AK.

Metropolis, N., A. W. Rosenbluth, M. lN. Rosenbluth, A. H. Teller, and E. Teller (1953), Equation of state calculations by fast computing machines, *The Journal of Chemical Physics*, 21(6).

Mira, A., and L. Tierney (2002), Efficiency and convergence properties of slice samplers, *Scandinavian Journal of Statistics*, 29, 1–12.

Mohr, A. E., E. A. Riskin, and R. E. Ladner (2000), Unequal loss protection: graceful degradation of image quality over packet erasure channels through forward error correction, *IEEE Journal on Selected Areas in Communications*, 18, 819–828.

Mousavipour, S., and S. M. H. Hojjati (2014), A particle swarm optimisation for timedependent vehicle routing problem with an efficient travel time function, *International Journal of Operational Research*, 20(1), 109–120.

Neal, R. M. (2003), Slice sampling, *The Annals of Statistics*, 31(3), 705–767.

Nosratinia, A., J. Liu, and B. Aazhang (2003), Source-channel rate allocation for progressive transmission of images, *IEEE Transactions on Communications*, 51, 186–196.

Office, T. P. S. (2010), Target to drive energy cost reductions in federal operations, creating clean energy jobs.

Polasek, W. (2014), Estimating marketing response models by mcmc, *International Journal of Mathematical Modelling and Numerical Optimisation*, 5(1/2), 24–44.

Ross, S. M. (2010), *Introduction to Probability Models*, Elsevier Inc.

Rowitch, D. N., and L. B. Milstein (2000), On the performance of Hybrid FEC/ARQ systems using rate compatible punctured turbo (RCPT) codes, *IEEE Trans. Communications*, 48, 948–959.

Said, A., and W. A. Pearlman (1996), A new, fast, and efficient image codec based on set partitioning in hierarchical trees, *IEEE Transactions on Circuits and Systems for Video Technology*, 6, 243–250.

Shahrzad Faghih-Roohi, M. X., and K. M. Ng (2014), Accident risk assessment in marine transportation via markov modelling and markov chain monte carlo simulation, *Ocean Engineering*, 91, 363–370.

Sherwood, P. G., and K. Zeger (1997), Progressive image coding for noisy channels, *IEEE Signal Processing Letters*, 4, 189–191.

Sherwood, P. G., and K. Zeger (1998), Error protection for progressive image transmission over memoryless and fading channels, *IEEE Transactions on Communications*, 46, 1555–1559.

Skilling, J. (2006), Nested sampling for general bayesian computation, *Bayesian Analysis*, 1(4), 833–860.

Skilling, J., and D. J. MacKay (2003), Slice sampling - a binary implementation, *Annals of Statistics*.

Sorensen, K., C. Vanovermeire, and S. Busschaert (2012), Efficient metaheuristics to solve the intermodal terminal location problem, *Computers and Operations Research*, 39, 2079–2090.

Stankovic, V., R. Hamzaoui, and D. Saupe (2003), Fast algorithm for rate-based optimal error protection of embedded codes, *IEEE Transactions on Communications*, 51, 1788–1795.

Stankovic, V., R. Hamzaoui, and Z. Xiong (2004), Efficient channel code rates selection algorithms for forward error correction of packetized multimedia bitstreams in varying channel, *IEEE Transactions on Multimedia*, 6, 240–248.

Stankovic, V., R. Hamzaoui, and Z. Xiong (2005), Fast algorithm for distortion-based error protection of embedded image codes, *IEEE Transactions on Image Processing*, 14, 1417–1421.

Thomas H. Cormen, R. L. R., Charles E. Leiserson, and C. Stein (1990), *Introduction to Algorithms*, MIT press.

Thomos, N., N. V. Boulgouris, and M. G. Strintzis (2005), Wireless image transmission using turbo codes and optimal unequal error protections, *IEEE Transactions on Image Processing*, 14, 1890–1901.

Tierney, L. (1998), A note on metropolis-hastings kernels for general state spaces, *The Annals of Applied Probability*, 8(1), 1–9.

UIRR (2009), $CO_2$ reduction through combined transport, *Tech. rep.*, International Union of combined Road-Rail transport companies.

UNECE (2009), Illustrated glossary for transport statistics, *United Nations Economic Commission for Europe.*

Xiaobin Wu, L. C., and P. Goggans (2012), Optimization for image transmission over varying channel with mcmc, *EURASIP Journal onWireless Communications and Networking (doi:10.1186/1687-1499-2012-275, 2012).*

# VITA

Xiaobin Wu obtained his BS in Computer Science from Shanghai University in China. He is a MS student of Department of Electrical Engineering at the University of Mississippi. Currently he is also a patient administration specialist in the US Army. His research area is Markov Chain Monte Carlo and applications.