Master Thesis

Application of Proxels to Queuing Simulation with Attributed Jobs

Author: Wenjing Xu
Master of Computational Visualistics

Supervisors: Prof. Dr. Graham Horton
Claudia Krull

Work Period: October 1st, 2007 – March 1st, 2008

Abstract

Queuing Theory is a branch of simulation which strives to provide analytical solutions to a number of queuing problems. If there is not analytical solution available, discrete event simulation is the commonly used method when facing queuing problems, but it has the drawback of being stochastic and only being able to solve the single specific parameterized problem. Compared to DES, using Proxels can provide deterministic result and make queuing simulation more efficient. But the current implementation does not support attributed customers, and therefore most queuing discipline are not included yet.

The goals of this thesis are expanding the application of Proxels to queuing simulation by adding the attributes to the jobs, and presenting the effect on the system performance by several planned experiments, finally proving the Proxel-based queuing system simulator is suitable to handle job's attribute with some accepetable restrictions.

Acknowledgments

I would like to use this opportunity and express my gratitude to all the people who gave me help and their support.

I am particularly thankful to:

- Professor Graham Horton for his guidance, support and his constant encouragement throughout the whole process of this thesis work.
- Claudia Krull who walked me through all the stages of writing this paper and gave me timely help to solve any kinds of problems during the work although she was in the vocation of having a baby.
- My father Xu Genying and my mother Ma Hongjie who gave me all their love, support for my study and everything in my life. My grandma's love and expectance is also the energy for my effort.
- My uncles Sha Hong and Sun Jianan who give me their help and loving care from the first day when I arrived in Germany.
- My boyfriend Yang Yue who always holding my hands whenever good or bad moments.
- My friend Wang Yao who always gives me her encouragement and pray for my success.

Self-Work Statements

Here with I declared that I have completed this work by myself and only with help of stated references.

Xu Wenjing

Matrikelnummer: 173683

Magdeburg, 10, February 2008

Contents

Chapter 1

Introduction

As the first chapter, the introduction firstly presents the background of this thesis and then states the motivation of choosing this theme in Section 1.2. The contents of tasks are specified in Section 1.3. Finally comes the structure of this thesis in a whole scene.

1.1 Background

Computer simulation is the way to modeling a real-life or a hypothetical situation on a digital computer, and analyzing the execution output, studying its behavior. In our daily life, computer simulation becomes more and more useful as a part of modeling many natural systems in physics, biology, machinery, economics and so on. Discrete event simulation is an operation that represents a system as a chronological sequence of events. Each event occurs at a point in time and marks a change of state in the system. A queuing system is one typical kind of discrete event simulation.

The goal of the analysis of a queuing system is finding analytical expressions for such performance measures as queue length, throughput and utilization. Queuing is an aspect of modern life that we may encounter any place and any time in our daily life. In the banks, people stand in a line in front of the counters and wait for the service. In the supermarkets, people wait in the queue to pay

for the goods. Although there are no people stand any line in the barber shop, the customers will be served in the sequence they arrive. In Figure 1.1, it is a picture of a real queue we can see everywhere. Obviously the basic components of a queuing system are queues, servers and customers. The study of queuing is able to provide both a theoretical background to the kind of system and the way in which the system itself may be designed to provide some specified grade of service to its customers. In queuing theory, a queuing model is used to imitate a real queue situation. The queuing behaviors can be analyzed mathematically by the performance measurements calculation. The performance measures are very important since they are often relevant with the work efficiency or economical losses of a real queuing system, such as an assembly line design. Therefore more accurate and deterministic results are expected in queuing system analysis.

Figure1.1 Queue (http://en.wikipedia.org/wiki/Queue_area)

Sometimes when there is no analytical expression can generate deterministic performance measurements for queuing models, discrete event simulation is commonly used as an alternative. But it seems not a perfect way since it has the drawback of being stochastic and only being able to solve the single specific parameterized problem. Compared to DES, using Proxels can provide deterministic result so that it is more suitable for the queuing simulation (more

details is in [2]).

The Proxel-based method is one simulation way which can be seen as the hybrid of the experimental and the numerical method because it contains the features of both classes of the methods. It is a state space-based simulation method, which generates the state space on-the-fly. The approach actually is the process that turns a non-Markovian model into a discrete-time Markov Chain (DTMC). This method does not require replications and produces deterministic result with an arbitrary accuracy for a discrete stochastic simulation model. “Proxel” is a term as an analogy of pixel which is constructed from “probability element”. A Proxel contains enough information for determining the transition probabilities in the next possible states, using so called instantaneous rate function.

1.2 Motivation

In the paper [2] wrote by Claudia and Horton, it describes the general approach and a tool implementing the Proxel-based simulation of queuing systems. And there are experiments validate the method and show the range of applicability. Still, the current implementation does not include all possible elements. Therefore the current tool can support the queuing system simulation easily and basically but the functions are not complete. Most queuing disciplines are not included yet because the current work does not support attributed customers. The simulation tool is restricted to process the queue in the most basic sequence FIFO. However queuing strategy is one of the essential elements of queuing system, it’s the rule for ordering the jobs in the queue.

Each queuing system of different strategy is modeled for different goals. Their goals can be some of the following: fairness, maximize throughput, minimize overhead, minimize waiting time, avoid infinite postponement, graceful degradation and enforce priorities. In order to achieve different goals, a queue may call for different strategies. The commonly used queuing strategies in a queue model are First In First Out, Last In First Out, Shortest Job First, Round Robin, Upper Time Limit, and Shortest Remaining Time, sometimes the

queue strategies are also mixed. What kind of queuing strategy one chooses

depends on what kind of goal one wants to achieve.

Therefore adding attributes of jobs in the queue becomes the chief future work of applying Proxel-based simulation of queuing systems.

In the Proxel-based simulation algorithm, the initial Proxel is created and for every simulation time step the successive Proxels are created. Following the algorithm, the implementation was structured modularly, so that an extension to attributed jobs is possible. However, adding attributes to the jobs will significantly increase the state space of the resulting model. Therefore the storage of attributes and the method to add the attributes into Proxels will be the greatest challenge.

1.3 Goals of the Thesis

1) The first goal: The Proxel-based Queuing System simulator can handle attributes successfully.

The current simulator is a basic queue simulator which uses the Proxel-based method. After several tests, the simulator was verified that all the elements work well. The future work of it is upgrading the simulator's functions and improving the simulator's capacity. Until now the simulator treated each job as the same, so in each discrete state we only need to consider the number of the jobs in the queue. Due to the reason of no queue strategies in the simulator, the main task of the future work is adding attributes to the jobs. This motion involved by many factors, such as the Proxel structure, the storage of the attributes, the performance calculation etc. Therefore successfully add attributes to the jobs in this simulator is the main goal of this thesis.

2) The second goal: Show the influence of adding attributes to the queuing system

The method of adding attributes will change the structure of the queue Proxel.

The size of the Proxel will be enlarged. Besides the structure of the Proxel, the main effect of adding attributes is expanding the state space significantly. Therefore the result including the performance measures, the amount of the processed Proxels and the computational time will be different from the no attributes situation. In order to control the attributes adding properly, such as the data type of the attributes, the storage or the algorithm adaption, we must make it clear that what kind of influence the attributes adding will bring and how the extent of the influence is. Some negative influence may be avoided or limited in a small range by adjusting the attributes data type, value set or the storage data structure. All in all, presenting the situation after adding attributes, reducing the negative effect to the lowest level and get a suitable way of attributes adding to the existing simulator is the other goal of this thesis.

3) The third goal: Display the suitability of adding attributes

The original Proxel-based queuing system simulator has only one queuing strategy FIFO. After successfully adding attributes, the adapted simulator can analyze more than one different queuing system. Therefore the range of queuing systems extends which the simulator is able to handle, although in this thesis only three attributes are planned to be added. Through the experiments of comparing the results which are generated by the attributed job system and non-attributed job system, the system can be proved that it still keep steadiness and accuracy of the performance measures. The influences of adding attributes also help to figure out that the approach described in this thesis is on a correct direction on the way of adding attributes.
Displaying the suitability of adding attributes following the process which stated in the thesis is the last goal of the project.

1.4 Structure of Thesis

The concept of this thesis will be organized into five chapters. The first chapter is the introduction. It will give an overview of the background knowledge and

the current method in this research field. The motivation of doing such a research theme is the important point in the part of introduction and the goals of the thesis are clearly stated. The second chapter is the description of the existed methods, the application of Proxel-based method into queuing system. The simulator is also showed in this part and including the evaluation of it, the advantages and disadvantages are discussed in this section. The future work of this simulator which is relevant of the thesis is finally described in the end of second part. The core content of this thesis is the third chapter. It describes the process of adding attributes specifically. The design of the approach and implementation of the plan are explained in this section. In the fourth chapter the experiments and the results presentation are shown. Each experiment and the results explanation are designed according to the performance criteria required to the analysis and proposed approach. The last chapter gives a brief summary of the results to conclude the thesis. Of course it also provides some suggestion of the improvement and a prospect of future work.

Chapter 2

Relevant Fundamentals and Existing Method: Proxel-based Method and Queue Theory

This chapter provides an introduction to fundamental knowledge along with the thesis and the existing approach which is advanced later in Chapter 3. In Section 2.1 we present the foundations and some commonly used terms of queuing theory and Proxel-based method. Furthermore in Section 2.2, the existing approach of using Proxel-based method to simulate queuing system is established, and also a discussion of its advantages, drawbacks and the future work.

2.1 Fundamentals of Queuing Theory

This section addresses the issues, which we classified as preliminary or necessary for understanding the basic principles of the Proxel-based queuing system simulator. The queuing theory and Proxel-based method which combined in the existing simulator are interpreted separately here; firstly is the theory of queuing system, a classical discrete stochastic model.

2.1.1 Preliminaries of Queuing System

Many real systems can be modeled as networks of queues, such as the waiting line in a bank, a bus stop. A queuing system can be described as customers arriving for service, waiting for service if the servers are occupied, and leaving the system after own service being finished.

One can begin the understanding queue theory with Little's Theorem. Little's Theorem states that: The average number of customers N can be determined from the following equation:

$$N=\lambda * T$$

Here λ is the average customer arrival rate and T is the average service time for a customer (detail can be found in [4]).

With Little's Theorem, the basic understanding of a queuing system was developed. A queuing system has three essential characteristics: arrival process, service process and number of customers.

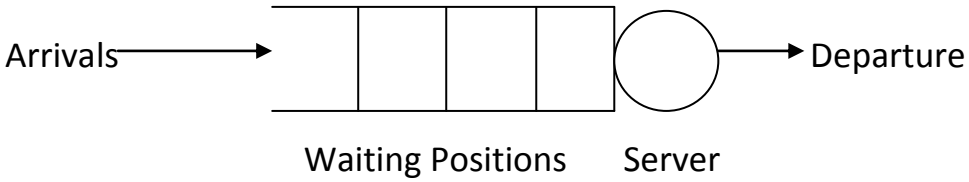

Figure 2.1 a Standard Graphical Notation for Queues

In Figure 2.1, it is a standard graphical notation for queues. The open rectangles with slots represent queues, and circles represent servers. The path is expressed as lines with arrows. Although the graphical notation expressive depicts the process of queuing system, it doesn't distinguish between tokens. As a result queuing strategies cannot be represented.

Until now, queuing systems have been well studied. As the shorthand for describing queuing processes, Kendall's notation was evolved. A thorough description of it can be seen in reference [4]. The notation describes a single

process as a series of symbols *A/B/X/Y/Z*,

Where

A: is the inter-arrival distribution of the customers
B: is the service time distribution
X: the number of parallel servers
Y: the maximum number of customers allowed in the system
Z: scheduling discipline/queuing strategy

In many situations only the first three symbols *A/B/X* are required because they are the three most important characteristics. Y is omitted if no restriction (default $Y=\infty$) and the queuing strategy is FIFO (FCFS).

When analyzing a system with a single queue, the most common performance measures are obtained as following:

ρ: *Server utilization* describes the fraction of time that the server is busy, or the mean fraction of active servers, in the case of multiple servers.
λ: *Throughput* describes the number of jobs, whose processing is completed in a single unit of time.
Q: Queue length is the number of jobs waiting in the queue at a given time.
W: *Waiting time* is the time that the jobs spend in the queue waiting to be served.
K: The number of jobs in the system at a given time.
π_i: The probability of a given number of jobs *i* in the system.

In queuing theory, almost every queuing system can be represented by Kendall's notation and analyzed as the above performances measures.

2.1.2 Queuing Strategies

Since one task of this thesis is adding attributes to the jobs in the queue, it means the queue in the whole system can be ordered according to different strategies. The following paragraphs present several basic queuing strategies definitions, more introductions can be found in [6].

A queue strategy determines the discipline for ordering jobs in a queue. It defines the order in which they are served and the way in which resources are divided between the customers. There are two main categories of strategies: Static or dynamic priorities and Pre-emptive or Non-pre-emptive

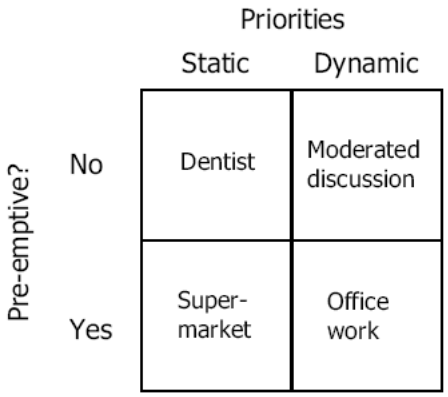

Figure 2.2 categories of queuing strategies (see [6])

In this thesis only the static and non-pre-emptive strategies are considered, because they are representative and most commonly used in queuing systems. Here are details of several queuing strategies:

First In First Out (FIFO): Jobs are ordered according to their arrival times. The job that has been waiting the longest is served first. This kind of queue is the most commonly used model because there is no differences between jobs. It is a fair queuing strategy and low overhead. There is no infinite postponement in such a queue. But long jobs may let the short jobs wait for a long time.

Priority Queue: Priorities are a measure of urgency or importance. Each job is assigned a priority and the jobs are sorted in the queue by their priorities. Low values represent high priorities. If some jobs have the equal priorities, the secondary criterion is used to sort jobs, such as FIFO.

Shortest Job First: Jobs are ordered by smallest estimated processing time. The shortest job is served first. So the waiting time of short job is small. But the latent danger is that the longer jobs may be infinitely postponed.

Deadline Scheduling: Each job has a deadline by which the job must be completed. Run the job whose deadline is the closet. This queuing strategy can

also be seems as a priority queue, because the deadline is a measure of urgency. The most urgent job can be served first and each job can be completed on time. But it may be not fair to the jobs whose deadline is a long term.

Mixed Strategies: Queuing model is always used for simulating and analyzing real world situation. Therefore the queuing strategies are often combined. Such as in a computer system, the jobs can be sorted by urgency priorities and also can be sorted by some non-pre-emptive strategy.

In this thesis, after the approach analysis and the attributes choosing, we will focus on the priority queue, shortest job first and deadline scheduling these three queuing strategies. The specific process for each queuing strategy will be described in later part (Section 3.2).

2.2 Proxel-based Method

This section provides an introduction to the basic terms of Proxel-based method and the algorithm description which are used throughout the thesis.

2.2.1 Foundations of Proxel-based Method

The Proxel-based simulation is a numerical approach for analyzing stochastic discrete models， it contains the features of both experimental and numerical approaches. Compare to the experimental approach, discrete event method, the Proxel-based method does not use random numbers. On the other hand, compare to the existing numerical approach, it avoids using partial differential equations by implementing the method of supplementary variables. In a very intuitive manner, the Proxel-based method follows the flow of probabilities corresponding to the behaviors of the model. These features enable the Proxel-based method can provide deterministic results and be obtained to an arbitrary accuracy. At the moment when no closed analytical expressions can be derived for some queuing system models, discrete event simulation might

be used as an alternative in previous time, but now Proxel-based method provides another choice.

This method operates directly on the model's state space whose generation is on-the-fly. The idea behind the Proxel method is to approximate the stochastic discrete process in a continuous approach using a discrete time step *dt*. This yields a computational model which consists of a set of discrete states at each discrete time step with corresponding probabilities. Proxel, which as a term is an abbreviation of the phrase "***pro****bability* ***el****ement*". It is the core computation unit of the Proxel-based method and it carries the adequate amount of information for generating its successor Proxels. The necessary elements and the information that each Proxel contains are the following:

- The discrete state which the model is in
- The relevant age information, also called age intensity vector, which track the time that each of the possible state changes has been pending
- The global simulation time which represent the total time from the start of the simulation
- The probability that the system is in the current discrete state with the corresponding age intensities.

In a formal, each Proxel can be represented as follow:

Proxel = (State, Age intensity, Time, Probability)

Proxel, as the component element of Proxel-based method, is structured as the above description and it stores the necessary information for depicting any probabilities configuration of a model. One can see Sanja's paper [12] for more thorough description. Next how are the Proxels implemented in the whole simulation process is introduced.

2.2.2 Proxel-based Method Description

In any system, at the time step 0, the initial Proxel is added.

Initial Proxel = (Initial State, 0, 0, 1.0)

Once the initial Proxel is added, the simulator generated its successor based on the possible state changes, update the age intensity vector and calculate the probabilities. If the successor's state is different from the initial one, the age intensity variable is reset to zero or set as irrelevant. Otherwise if the successor Proxel stays in the same state of the initial one in the next time step, then the age intensity variable is incremented by the size of time step *dt.* Once the second generation of Proxels is computed, the initial Proxel can be thrown away. The next generation is computed in the same way with the second one. This procedure is repeated during the whole simulation process until reach the end of the simulation time.

Let's see an example to demonstrate how the Proxel-based method works. In Figure 2.4, it is a two-state model which has two possible state changes that switch between the two discrete states, S (sunny) and R (rainy).

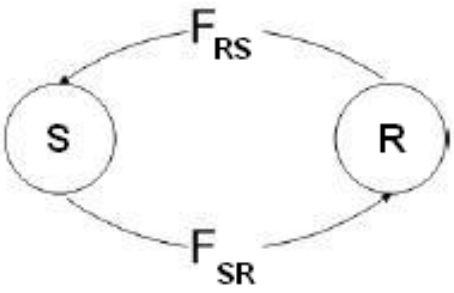

Figure 2.4 Example State Diagrams

At time step 0, let Sunny be the initial discrete state and the time step is *dt*. In state S, it is the age intensity vector of the state change from Sunny to Rainy, distributed according to the distribution function F_{SR}. And in R, it is the opposite situation, tracking the age intensity of change from R to S, distributed according to function F_{RS}. After generating the initial Proxel ((S, 0), 0, 1.0), at the second step t=2dt, the model can either stay in the state S or change to state R. The age intensity is incremented by *dt* for the former situation or is reset to zero when the state change happens. The probability is calculated by the

instantaneous rate function μ (τ), integrated along the time step, where τ is the age intensity vector for the active state change.

$$Probability = \int_0^{dt} \mu\,(x)dx,$$

The IRF (instantaneous rate function) is generated from the distribution functions CDF and PDF (details see [12] Section 2.1.1). So the second proxels are following: ((S, dt), dt, *) and ((R, 0), dt, *), the probabilities are omitted for the reason of simplicity. At each step, proxel based approach generates a number of proxels for all possible discrete states. The whole process can be described as a *tree* stucture. This tree structure is named *Proxel tree*. In Figure 2.5, it is a Proxel tree of the Sunny and Rainy model, the first four steps.

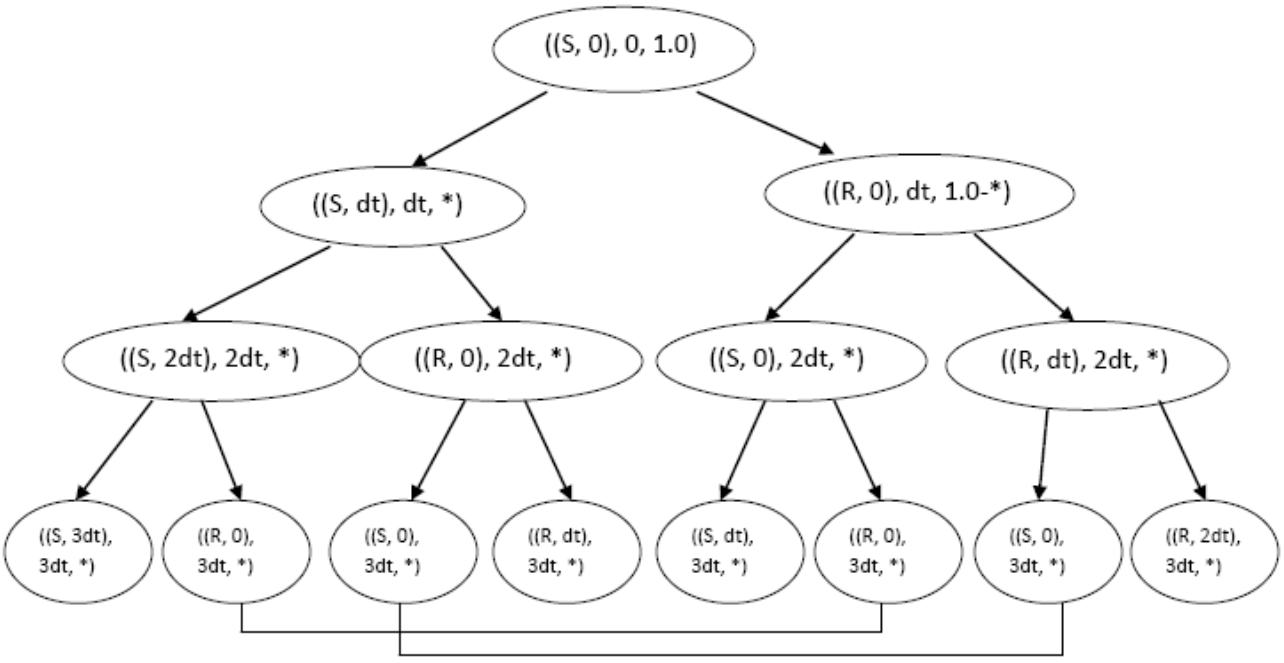

Figure 2.5 a Proxel Tree of The First Four Steps ForThe Example of Figure 2.4

From step by step, more and more Proxels are generated. From Figure 2.5, we can find in the fourth step, there are two pairs of Proxels which are connected by straight lines, their discrete states and age intensity are the same. We define the Proxels have this situation are the same Proxels and add their probabilities together. This reduces the state space storage by keeping one Proxel for one

kind. Therefore in the example, at fourth step, we only need to consider six Proxels.

The theory and a simple example presented in this section show the basic operation disciplines of Proxel-based method. Further, in Section 2.3, applying Proxel-based method for simulation queuing system is stated in detail.

2.3 Existing Approach: Application of Proxel-based Method in Queuing System

After describing the fundamentals of the queue theory and Proxel-based method, in this section, we will present the existing approach which combines the queue theory and Proxel-based method together. Furthermore we demonstrate how this simulator works and give some evaluations for it.

2.3.1 Implementation

The Proxel-based method can be applied to any discrete event system. A queuing system is one of the classical discrete event systems. Because stable queuing systems that can serve jobs faster than they come, have a steady state solution, this approach only consider these systems. The implementation starts from an initial system state and determines the possible next states and the transition probability within the discretization time step. Based on the queue theory and characteristics of a queue system, the state parameter of the queuing system is stored into the Proxel data structure directly as its elements:

$$P_x = (q,\ \vec{s},\ \tau_q,\ \vec{\tau}_s,\ p)$$

Here with the elements meaning:

q: the number of jobs in the queue,

$\vec{s}$: the occupation status of the servers,

τ_q: the age of the arrival process,

$\vec{\tau}_s$: the ages of the different service processes,

p: the probability of the combination

With the Proxel data structure, after the initial Proxel created, the generation of the next states in every simulation time step is performed. The remaining number of jobs in the calling source is encoded if specified. There are three states for such a queuing system, the job arrivals, the job's service is finished and stay the same status. In the algorithm, for each Proxel, the three kinds of probabilities are calculated. If the arrival probability is more than zero, the arrival Proxel will be generated. If the service finished has a positive probability, the service finished Proxel is created. The stay Proxel is generated when the stay probability is above zero. At each time step, the same loop is performed. The simulation time is controlled through the user interface.

After the simulation of the model, the performance measures are calculated from the simulation result to obtain the relevant information. For each implementation, we get the following result of both transient and average:

Transient server utilization and average utilization,

Transient queue length and average queue length,

Average job waiting time,

Transient number of jobs in the system and average number,

Transient probability for a defined number of jobs

The above description presents the algorithm underlying the Proxel-based queuing system simulator. Next an example is used to show the implementation process of the simulator which operated by users.

2.3.2 An Example

The user interface is also designed in order to analyze and parameterize a queuing system clearly. A user can specify the queuing system, execute the simulation, plot the transient results and read the performance measures via this graphical user interface. The following example demonstrates how this simulation tool works. It is a queuing system M/G/1 which has a Markovian arrival and general service process and two servers.

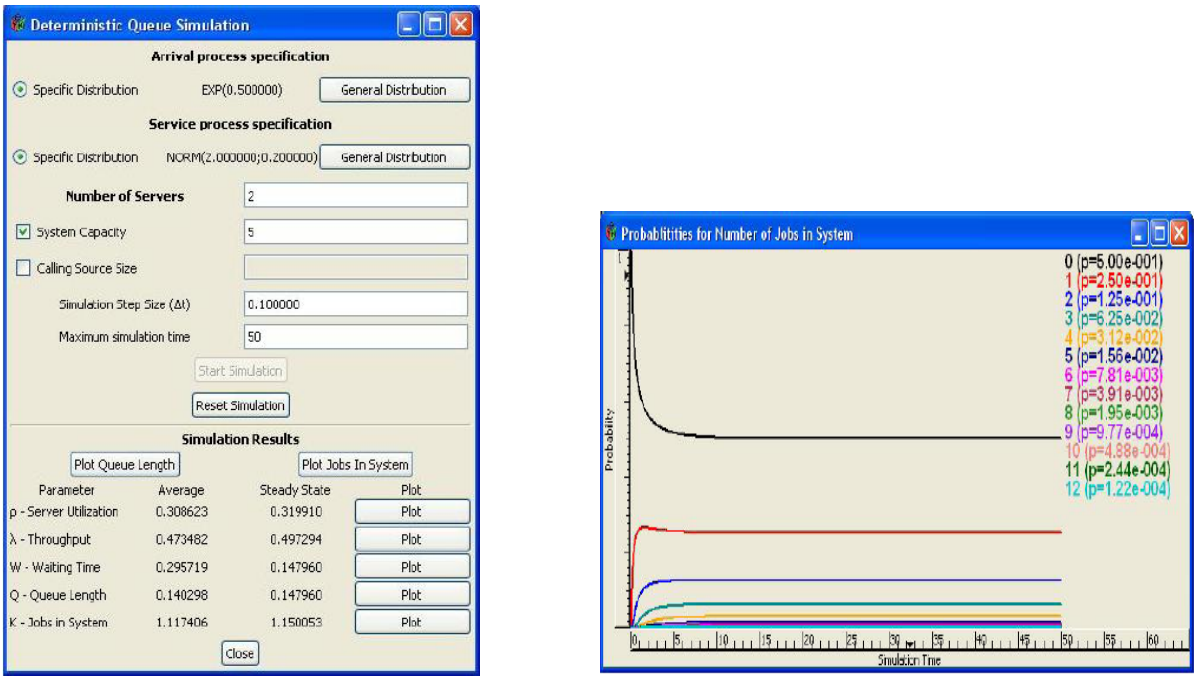

Figure 2.6 User Interface for Queue Simulation Using Proxels (left) and Probabilities of Jobs' Number in System (right)

In left image of Figure 2.6, the arrival rate is set as μ_1 = 0.5, and service parameters are N (2.0; 2.0) in the upper part of the GUI. In the middle part of the interface the time step, number of servers, system capacity, calling source size and simulation time can be modified. When all options are ready, user clicks the "start simulation" button. The lower part shows the simulation output. The average and steady state values of five performance measures are

represented after the simulation. The plot buttons are used for the graphs of transient values of these measures. The two upper buttons display plots of queue length and jobs in system, the five lower plot buttons are for transient values of each performance measures. When any plot button is clicked, the plot graph for the corresponding measures is represented as the right image in Figure 2.6.

2.3.3 Evaluation of the Simulator

This application of Proxel-based method in queuing system can be seen as an alternative to discrete event simulation of queuing models, since it provides deterministic both transient and steady state results for the performance measures. It can solve some stiff problems more quickly than using discrete event simulation even the discrete event simulation can't solve. In paper [2], there are more details of different benchmark experiments. They showed the results of different parameterization of the queuing system.

Until now, this simulator is treating the queue system with the no-attribute jobs. It means there is only one queuing discipline in the queuing system, FIFO. Although the state space will be significant increased after adding job attributes, and this is the bottleneck of the Proxel-based method, if we control the attributes value in a small set and structure the algorithm properly, adding attributes to the jobs in the queue is can be implemented. This is what we introduce in next chapter and the core concept of this thesis.

Chapter 3

Adding Attributed Jobs in Proxel-Queuing System

In this chapter we provide a specific description for the process of adding attributes to the existing simulator. In Section 3.1, the selection of attributes and the reasons are discussed. Then in Section 3.2, the method of adding attributes to the algorithm, programming complexity are established and also the effect to the original algorithm in theory. Finally, in the last section, the improvement of the user interface is showed.

3.1 Choosing Job Attributes

This section is the description for the job attributes selection. First portion presents the essential factors to be considered. In the second portion, the selected attributes and values are shown specifically.

3.1.1 Criterion of Attributes Selection

How to solve the problem that the existing simulator is not able to process the queuing system models without queuing strategies? The root cause is that the jobs in the queuing system are without attributes. Jobs attributes represent

instead job specific information and specify in some way actions that have to be performed for scheduling the jobs. Choosing proper types and values for the attributes of the jobs in the queue is the essential preparation before attributes adding. On the one hand, the attributes in the different queuing strategies are the samples for selecting. On the other hand, the features of Proxel-based method must be considered.

In Section 2.2.2 several representative queuing strategies are introduced. For the job without attributes, the basic strategy FIFO is used in the queuing model. If the jobs are attached different levels of priority, the queue of jobs is the priority queue. Each job can also be attached a processing time or a deadline as their attributes. Since the attributes adding is the first try to the existing simulator, it is better to choose the common and classical queuing strategies. Therefore at the first glance, the following three attributes are focused: priority, processing time and deadline.

The idea behind the Proxel-based method is predicting all the things might happen in the next time step by considering all the information at the current moment. So at each time step, it generates all possible success Proxels for each current Proxel. Actually the Proxel-based method is a state space-based simulation, which generates the state space on-the-fly. Adding attributes will significantly increase the state space. For this reason avoiding state space explosion is the chief goal when selecting attributes. As the introduction in Section 2.2.2, there are two main categories of queuing strategies: dynamic or static and pre-emptive or non-pre-emptive. Dynamical queuing strategy means the attributes might be change in any point of time. In the Proxel-based queuing simulator, at each time step, the probabilities of all possible states at next time step are calculated, so that the possible states should be foreknown. Dynamical attributes may raise the difficulty of predicting the possible states and the probabilities calculation of them. In the other hand, it is difficult to control the state space increasing rate. If the attributes levels change to a large number and the arrival probability is more than zero, then the number of success Proxels will be a large number accordingly. Use static attributes, the principle of ordering jobs is clear and the calculation of state probabilities can use a proper and steady method. Compare dynamic and static attributes, the static attributes are obviously easy to implement. Consider the pre-emptive

and non-pre-emptive queuing strategies. Pre-emptive queue means when a job with higher priority comes, the service for a lower priority job will be interrupted. In this way, the service Proxel is difficult to structure, and must find another way to record the age intensity variable because the state might be interrupted in the middle. Relatively, non-pre-emptive queuing system can ensure each state has a complete process. The Proxels do not need to store the interrupted jobs additionally. After the above thinking, the static attributes: priority, processing time and deadline with non-pre-emptive discipline is the final decision of job attributes choosing.

3.1.2 The Detail of Three Attributes

After the concepts of the objective attributes are determined, the next step is discussing the details of attributes, such as the type, values.

1) Priority: a measure of urgency or importance

Besides the FIFO queuing strategy, the priority queue is the most commonly applied in queuing system modeling. For example, in a hospital, the customers with emergent state of an illness should be handled first. When simulate such a queuing system, the priority as the attributes of jobs in the queue is the best way to measure the urgency or important extent.

To represent the levels of priority, the integer data type is the most suitable. Integers are not only able to measure the different priorities, but also easier than other data type for the following treatment in the whole algorithm, such as id calculation, jobs ordering. Furthermore, the storage of integers does not need a large space. In the arrival process, when a new job comes into the queue, it will be assigned an integer as its priority. Then the queue of jobs is reordered according to their priority values. The value is lower, the priority of job is higher, and the job is arranged at the more forward position in the queue. The situation must happen that some jobs are assigned the same priority values, at this moment the queue need a secondary ordering criterion, generally the FIFO is applied.

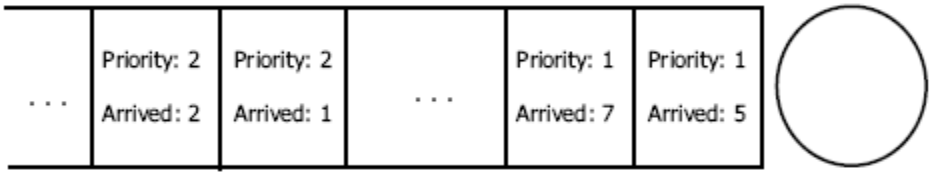

Figure 3.1 Example of a Priority Queue
(image from the slide of [6])

In Figure 3.1, the rectangle represents a queue of jobs with priorities, and the open side is the access for new jobs coming. The circle is the server which the jobs are waiting for. In the queue, the jobs with priority 1 are in the front of the jobs with priority 2. In the sub-queue of jobs with priority 1, the arrival time point is the secondary attributes. The job that arrived at time point 5 stands in front of the job came at time point 7. Therefore the job which has the highest priority and earliest arrival time will wait for the least time to be served.

2) Processing Time: a measure of jobs' length

Shortest Job First queuing strategy is sometimes applied in the internet dispatch system. This kind of algorithm can let the shortest jobs be finished quickly, furthermore can suffer a traffic congestion in a queuing system model.

In the Shortest Job First queuing system, processing time of the jobs is the information to be focused. When a new job arrives, it will be attached a processing time which can measure the length of this new job. After new job getting into the queue, the jobs in the waiting queue is reordered. As the name of the queuing strategy, the shortest job is put in the front of the queue in order to be served first. The same as priority queue, if there are several jobs with the equal processing time, they are ordered according to their arrival time points. Generally in the common queuing system, the processing time of the jobs should be represented by double data type. But in the Proxel-based queuing system whose algorithm is structured step by step although the simulation time and the *dt* are still double data type. Therefore instead of a double data type, the processing time is decided to be represented using an integer which measures the amount steps of each processed job. Compare to the priority queue, there is a problem of Shortest Job First that it requires a

priori knowledge of processing time. The priority can be set definitely before simulation, but the processing time is relative to the service process so that it needs to find a proper way to calculate the processing time out before the job coming. Additionally, to avoid the state space explosion problem, the values of the processing time must be controlled in a small set. And the processing time is according to the service distribution functions, so the calculation is more complex. In Section 3.2.2 the specific calculation method is described.

3) Deadline: a measure of urgency

The jobs are attached a time limit in some conditions, for example the work in a memorandum. In such a queuing system, each job must be completed by a specific time. Deadline is another expression for a time limit, and it's also a measure of job's urgency.

Similar with the above two attributes, a deadline is added to each job when it arrives into the queue. Then reorder the queue according to the jobs' deadline. After the arrangement, the new coming job is put behind the jobs whose deadlines are equals to it and in front of the job whose deadline is farther than it. From the meaning of the word "deadline", it should be a time point. But in a model of queuing system, setting the deadline as a real time point will make the following treatments too complicated. The "closest" deadline is actually mean the time interval between the deadline and current time is the shortest. Therefore using a time interval instead of deadline will significantly reduce the complexity. Imitating the processing time, the time period can be represented by an integer whose value is *t=deadline-current step.* Currently the most complex problem is how to know the deadline in advance. The deadline has no relation with both arrival and service process, so a special distribution function is necessary for generating the random values of deadline. Similar with the ones of the arrival and service, the kind of deadline distribution is specified by users through user interface. Properly control the values of deadlines in a small group is the essential work for avoiding the state space explosion which is the hidden danger of Proxel-based method.

Totally, the tasks for adding deadline to the jobs are adding a special distribution function for deadline, generating random deadline values for each job when it arrives, and reordering the queue to make sure the job with the closest deadline is served first. In addition, an interface for user to choose deadline distribution is required. The advantage of adding deadline is that the system can ensure that all the jobs are completed on time. The specific description of the adding process is in the next part (Section 3.2).

3.2 Adding Jobs' Attributes

After the attributes selection and their additional details determination, the plan of how to adding these attributes is described. In this section, the process of implementation for adding the attributes into the existing simulator is described particularly.

3.2.1 Preparation of Adding Attributes

Finding a suitable data structure for storing jobs attributes is the main task in the preparation before all the operations. The data structure of array is an easy and common choice for attributes storage. As experiments, three different data structures based on array have been tried.

1) Three Arrays Three Attributes

As the subtitle described above, for each kind of attribute build one independent array for storage. So there will be three arrays defined as priority array, processing time array and deadline array. In the arrival process, a new job is attached all the three attributes. The attributes will be input into three particular arrays according to their categories. Since each job has three attributes, all the arrays are inserted into each Proxel to represent the jobs queue. If the system is specified to simulate the jobs with priority, the array of priority will be sorted as higher priority job first principle at each simulation step. The other arrays keep the input and out attributes movements when jobs

arrive and leave, but they do not need to be ordered. The work in this thesis currently only supports one attribute per simulation. That means at each implementation three arrays are waiting to be chosen, but at most one attribute array will be used, even when the simulator process no attribute jobs, the three arrays will lie idle. Actually at least 2/3 storages are wasted during each simulation and the simulator must spend time on input and out the attributes to the arrays which are set aside. The most unsatisfactory point is that three arrays per Proxel must need a large space when the jobs number is big. Furthermore the discrete states will be increased along with the adding attributes, and this must significantly extend the state space. Such a three arrays data structure may cause the space explosion or the simulator extremely slow speed problems. Three Array Three Attribute seems not a smart choice.

2) One Array Three Attributes

The first data structure actually makes three different formats copies for each job so that the simulator can use one of them according to the user's requirement. For every job, each array does the input and output movements. In another word, the same operation is implemented three times per job. To save the time wasted in duplicated process, combining three arrays into one is a solution.

On the basis of principle of single "Proxel", each job with different attributes can be represented by a unit. In order to keep the convenience for adding other categories attributes in the future work, it is necessary to build a structure for the job unit which contains the priority, processing time and deadline currently. The total structure for the queue is still an array, but the members in the array become the units of jobs. Certainly other data structure can be tried, such as linked list. If other attributes are added, there may be other better data structure. This is a subject of future research. In each Proxel, an array of jobs unit is inserted. All the attributes management focuses on one objective so that the duplicated movements in the first method can be omitted. In a contrast, saving time is the advantage of this data structure. The jobs units will be reordered as the selected queuing strategy at each arrival time step, however still only one attribute is focused at each run. Another benefit of using jobs unit is that it is convenient to check the situations of the other unselected

attributes during the simulation. User may be interested in the other information of the jobs, if so the first data structure is hard to provide a quick solution, but the array of job units is able to track each job quickly. The disadvantage of this data structure is the same as the first one, the storage space problem. The size of Proxel definitely grows if three even more attributes are inserted into as members.

3) One Array One Attribute

For general queuing system simulator, last data structure that represents each job as a unit seems a proper one. But to the simulator which using a Proxel-based method, storing all the information of the jobs in each Proxel is not appropriate way. In order to saving storage space, besides controlling Proxels number, the size of each Proxel should not be too big. For this reason adding essential attributes which users need is more suitable than including all the information of jobs.

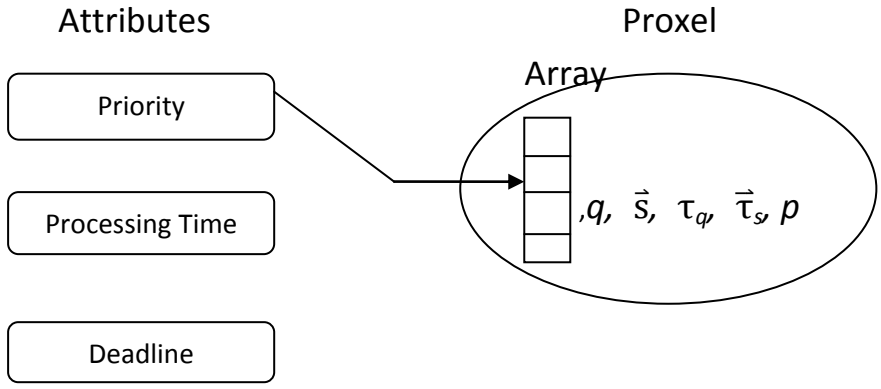

Figure 3.2 the Attributes Storage with a Triple Switch

A triple switch is the key in this method. One empty array is built for the attributes coming. The triple switch is used to control which category of the attributes comes. The array with the selected attributes is added into each Proxel as the new element. Figure 3.2 shows the structure expressively. In the image the switch is turn to priority, the array in each Proxel will store priorities and ordered as priority queue principle. This method significantly reduces the storage space than the above two data structures and also saves the computing

time. It is also the most convenient structure for increasing the categories of attributes among three ways. Compare to the second structure, the only lack is that it does not support the simulator to track the other attributes of the jobs. But the situation of users have interested on the other information may not happen frequently. Weigh up the advantages and disadvantages, this regret can be accepted or put into the future improvements. Obviously, One Array One attribute structure is the most suitable one for the current simulator, and it is selected to be applied later in this thesis.

3.2.2 Adding Priorities

In the algorithm of existing Proxel-based queuing simulator, the Proxel is structured as $P_x = (q, \vec{s}, \tau_q, \vec{\tau}_s, p)$, the meaning of these symbols is described in Section 2.3.1. The only element in a Proxel for representing the jobs in the queue is the number of the jobs *q*, because each job is seen as a unit without additional information. For this reason the servers only have two status “busy” and “idle” which can represented by a Boolean variable. Adding priority to each job signifies the current Proxel element *q* is not enough to depict the complete substance of the jobs. It is necessary to add another element to each Proxel in order to represent the priority of the jobs in the queue. The data structure of array is an easy and common choice for storing priority. The storage design has been described specifically in Section 3.2.1. An array with a triple switch for choosing attributes is applied here. Priority will be added when the switch open the access of the array for it. The servers’ state is also the part of discrete state of a Proxel. As a result of attributes adding, the Boolean servers array is changed to integer array corresponding to the attributes storage. The original Proxel is upgraded to the following, where “*jobs”* is the array for the attribute storage:

$$P_x = (jobs, q, \vec{s}, \tau_q, \vec{\tau}_s, p)$$

The priority should at least have two values for describing the different levels of job’s importance. To show the least influence of the original system, each job has a priority either “1” or “2” which “1” represents the higher priority, and “0” means no job.

In the algorithm, at each time step, if the probability of arrival is more than 0, an arrival Proxel is generated. Now the new job is possibly to have either priority 1 or priority 2. According to the idea of Proxel-based method that generates the Proxels of all the situations which possibly happen at the current time step, the original arrival Proxel is split into two Proxels. One is for the priority 1- job coming, and the other is for priority 2. The probability of each such Proxel is set the half value of the arrival probability. Certainly the probabilities can be set in other different way, such as one is 40% of the arrival probability and the other is 60%, but the amount of the both probability must equals to the arrival probably. The probabilities will be set in different ways in the experiments of next chapter to observe the influence of the result.

With the restructured Proxel structures, the adapted algorithm works as follows: At time step 0 of each simulation, the initial Proxel P_x *= (0, 0,* $\vec{0}$*, 0,* $\vec{0}$*, 1.0)* is created and for every simulation time step the following loop is performed.

/* Algorithm 3.1: Adding Priority */

```
01 FOR every Proxel P_x
02          p_arr = P (arrival);
03          FOR EACH occupied server s_i
04                  p_serv_i = P (service i is finished)
05          ENDFOR
06          p_stay = 1 – (sum (p_serv_i) + p_arr)
07          normalize_probabilites ();

08          IF (p_arr > 0 && priority_set == true)
09                  create_arrival_proxel_1 ();
10                  create_arrival_proxel_2 ();
11          ENDIF
12          IF (p_arr > 0 && priority_set ==false) create_arrival_proxel ();
13          FOR EACH occupied server s_i
14                  IF (p_serv_i > 0) create_service_finished_proxel ();
15          ENDFOR
```

```
16          IF (p_stay > 0) creat_stay_proxel();
17 ENDFOR
```

The first part (02 - 07) calculates the probabilities of possible state changes. The detail description of can be found in page 5 of [2]. The second part (08 - 16) generates the Proxels for next simulate time step of the system. The part (08 - 11) is the operation of generating arrival Proxels with attributed jobs when the switch of priority is on.

In each arrival Proxel, when a new job with a priority comes, the queue is reordered immediately. Contrast the steadiness and algorithm complexity of several sorting methods. The commonly used insert sorting whose time complexity is $O(n^2)$ and space complexity is O(1) is applied as the method of reordering the queue. So that the new job can be inserted into the array of the queue according to its priority directly, the jobs after it will backward for one position. After sorting the queue, if there is any free server, then get the job which is at the head of the array immediately. The current server is occupied and set the same priority with the processed job, meanwhile this job is removed from the queue, the jobs behind are forward by one position. The service finished and stay Proxels keep the same operation as before, but what needs to mention is that the servers status should be updated to integer ones.

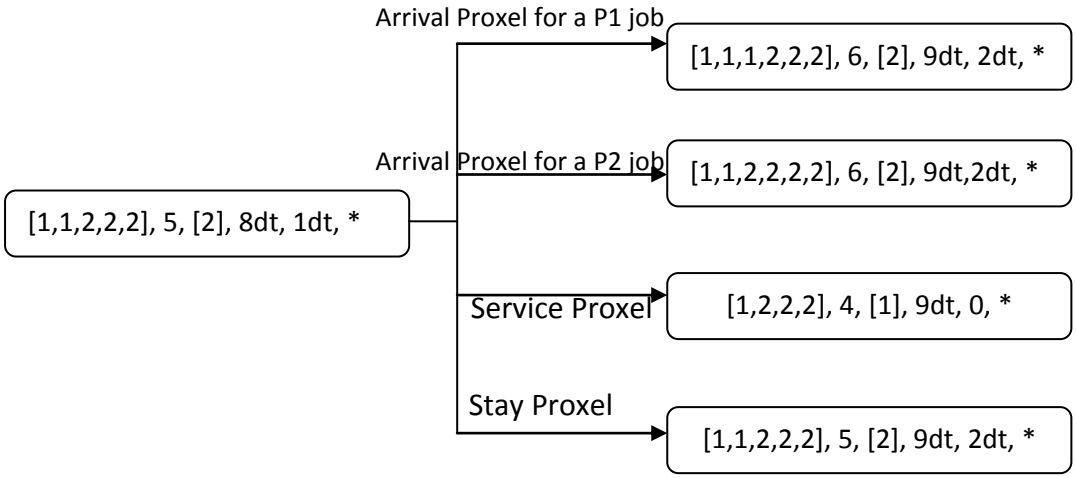

Figure 3.3 a Branch of the Proxel Tree for Modeling the Queuing System with Priority Jobs

In Figure 3.3, it depicts a branch of the Proxel tree which simulates the queuing system with the priority-jobs from the 8^{th} to 9^{th} time step. The left rectangle is one of the Proxels at 8^{th} time step. It contains an array with priorities [1, 1, 2, 2, 2], the number of jobs in the queue 5, the server which is occupied by a priority 2 job, the age of simulation 8dt, the age of service 1dt, and the probability which is omitted as a star. The probabilities of arrival, service, stay are all more than zero. At the 9^{th} time step, the most above Proxel represents the job of priority 1 arrives, the new job is inserted into the third position of the array. The simulation age and the service age are added by one time step. The second above Proxel is the priority 2 job arrival Proxel. The Proxel below it is the service finished Proxel and the lowest Proxel is the stay Proxel.

Since the jobs are only attached priorities without effect on other variables in the model, the performance measures of the whole system will not change much. In order to observing the individual performance measures for each priority class, the analysis results for priority 1 and priority 2 jobs are calculated separately. The performance measures calculation formulas are adapted as following. The simulation time step is denoted as *dt*, *i* the priority class, m the total number of servers in the system, m_i the number of servers occupied by priority *i* jobs, k the current time step, *tmax* the maximum simulation time and *kmax = tmax / dt* the maximum number of simulation time steps, *s* state.

Transient server utilization:

$$\rho_i[k] = \sum m_i/m * P(s)[k]$$

Transient throughput of the system:

$$\lambda_i[k] = throughput_i[k]/dt$$

Transient queue length:

$$Q_i[k] = \sum queued\_jobs_i(s_i) * P(s)[k]$$

Average job waiting time:

$$\overline{W}_i = \sum Q_i[k] * dt / \lambda_i$$

Transient number of jobs in the system:

$$K_i[k] = \sum(s_i + queued\_jobs_i(s_i)) * P(s)[k]$$

The average values are calculated similar as the performance values calculation of the whole system. Although these separate analysis results are not shown in the user interface, they are recorded in additional files and drawn out afterwards as an observation compare to the total performance measures.

3.2.3 Adding Processing Time

The information contained by processing time is a measure of jobs' length. When the data structure switch for the processing time is turned on, the array for storing processing time is inserted into each Proxel as an element. Since the data type of processing time is also integer, the array is the same with the one which stores the priorities. In order to ensure the shortest job be served first, after each job arrives, the array will be reordered using insert sorting. The value of processing time is lower; the job is at the more front position in the array.

There is a different point between adding priority and processing time. The priority values are determined before each simulation, but the values of processing time are unknown in advance. How to generate a processing time for a new coming job? The key of solving this problem is requiring a priori knowledge of processing time. Reviewing the description of shortest job first queuing strategy, the processing time of each job is actually the prediction of how long the job to be served. In another word, the processing time is related to the service process and the value depends on the service distribution function. When a new job arrives, all possible values of its processing time must be generated according to the service distribution which is selected by user through the simulator interface. This means in a specific service distribution function, the processing time values whose probability is more than 0 must be chosen. Supposing y = F(x) is the service distribution function, x is simulation time, y is the probabilities, the values of x that makes F(x) > 0 is the possible processing time. In the existing simulator, the functions of service process have included most commonly used probability distributions. Three types of these distributions are easier for processing time generation: Deterministic, Uniform and Triangular. The reason is that the value of Deterministic distribution is

definite, and the x values are limited in a finite interval of Uniform and Triangular distribution. But the other four distributions (Exponential, Normal, Lognormal, and Weibull) don't have a limitation of x values. This is not only difficult to calculate the processing time but also may cause the potential danger of Proxel-based method: the state space explosion. If there are a large number of possible values for the processing time, each arrival event will be divided into a large number of Proxels at the time step which the arrival probability is more than 0. The state space must explode after several simulation steps. Therefore it is necessary to find a proper way for controlling the processing time's value set.

In the original algorithm of Proxel-based queuing system simulation, the applied distribution function is instantaneous rate function (IRF) which is computed from cumulative distribution function (CDF) and probability density function (PDF) (see detail in [12] Section 3.2). Although IRF can provide absolute probability values, for processing time generation, the CDF is sufficient because the focused values are integer service time steps but not the double probabilities value. In Figure 3.4, they are the line images of the four CDF which have infinite support of x-values. The y-coordinate represent probability, x-coordinate is simulation time step. The maximum y value is 1.0. The common ground of these four images is that the probability value tends to 1.0 in the pace of x value increasing. Begin from a certain x value; the probabilities are almost equal to 1.0. The state space explosion problem can be solved by define the "certain" x value a cutoff point to limit the processing time in a small set. This value is the maximum processing time, and is represented by *pt_max* in the following. To dumping the surplus x values whose probabilities are almost 1.0, the probability boundary can be set as 0.99. Then the x value which corresponding the probability 0.99 is chosen as the cutoff *pt_max*. All the possible values for jobs' processing time are *1/dt, 2/dt, …, pt_max/dt*. When an arrival event of a new job happens, *n= pt_max* Proxels are created, and each Proxel according to one processing time value. For Deterministic, Uniform and Triangular distributions, the finite interval of x values divided by *dt* are the possible processing times. A proper way of processing time calculation is found, the type of distribution and the parameters of the function are determined by user. Actually the processing time's values are decided by the user's choice. But if user gives a large range or

an unsuitable parameter to the service distribution function, it may also a state space explosion. This is the disadvantage of adding processing time and it restricts user's convenience of selection.

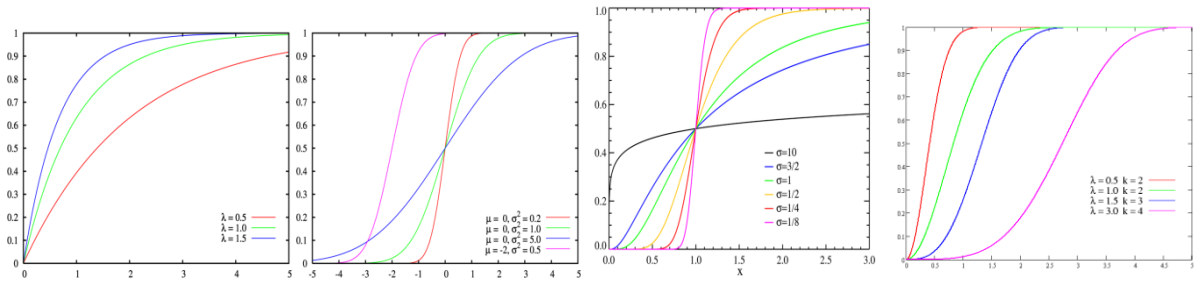

Figure3.4 left to right: CDF Images of Exponential, Normal, Lognormal, and Weibull Distributions
(Images from *http://en.wikipedia.org/wiki/Probability_distribution*)

After processing time generation, there is a little adaption for service finish probability calculation. In the original simulator, the service finish probability is the generated according to the service process distribution. Now the service time of each job is calculated out in advance, so the service finish probabilities are deterministic values. In service event Proxel, count down the processing time of the job which is in a service at each step. If the processing time is more than 0, it means that there is no possibilities for service finish happening. When the processing time is counted to 0 at certain time step, then the possibility of service finish is 1.0. The service process is much concise after adding processing time to the jobs in the queue.

The adapted algorithm is as following:

/* Algorithm 3.2: Adding Processing Time */

```
01 FOR every Proxel P_x
02          p_arr = P (arrival);
03          FOR EACH occupied server s_i
04                  IF ( pro_t > 0) p_serv_i = 0;    (service i is finished)
05                  IF ( pro_t == 0) p_serv_i = 1;
06          ENDFOR
07          p_stay = 1 – (sum (p_serv_i) + p_arr)
```

```
08          normalize_probabilites ();

09          IF (p_arr > 0 && pro_t_set == true)
10                  create_arrival_proxel_1 ();
                        …
n                   create_arrival_proxel_pt_max ();
n+1         ENDIF
n+2         IF (p_arr > 0 && pro_t_set ==false) create_arrival_proxel ();
n+3            FOR EACH occupied server s_i
n+4               IF  (p_serv_i  =1)  create_service_finished_proxel  ();     // count
                                                          down the processing time in
                                                          this proxel
n+5            ENDFOR
n+6         IF (p_stay > 0) creat_stay_proxel();
n+7      ENDFOR
```

Follow the above process and the depicted algorithm, the processing time can be successfully added into each job in the queue. The deadline, as the third attribute is described next.

3.2.4 Adding Deadline

When the triple switch is turned on to add deadline, each coming job is attached a deadline attribute, actually a time period for the job staying in the queuing system. The job with closer deadline is placed in a more front position of the queue. Since the deadline is converted into time period in our plan, the approach of adding it has similar way as adding processing time. Both of processing time and deadline focus on an interval of “time steps”, so their data types are all integers. And the storages data structure are integer arrays which is selected by the triple switch.

Deadline is the time limitation for a job staying in the system, and it is given by users without the relation to arrival and service rates. Therefore the deadline generation isn’t according to any element of the whole system. The solution of inputting the deadline value to each job can imitate the arrival and service

process, generating random numbers by a separate distribution which can be added especially for deadline. The distribution function and the parameters are specified by users, and then all possible values for deadline can be calculated out. If a separate distribution part is available with generating the ending points, then a particular distribution part is also able to directly compute the time steps interval of each job staying in the system. From the above analysis, the deadline generation is much similar as the processing time one. The distinction is that the deadline distribution part is not relative to the arrival and service processes.

After building a separate channel for deadline, the following steps can imitate the adding processing time process. In order to prevent the overflow of state space in Proxel-based simulator, controlling the size of the deadline's value set is an essential operation. The same solution in the process of adding processing time can be used here. Collect the values whose probabilities are more than 0 and less than 0.99 for Exponential, Normal, Lognormal, Weibull distributions and divided by single time step *dt*. And when users select Deterministic, Uniform and Triangular distributions, just use the x-interval's boundary values divided by *dt* to calculate the upper and lower values for the deadline values group. Contrast to the first method, the third one is much more feasible to implement and the deadline generation is in a separate channel without complex connection. Furthermore the third method costs less computing time than the second one. All in all, in these enumerated ways it is the most suitable one for adding deadline to the jobs in the queue.

The subsequence steps are on the basis of original Proxel-based queuing simulation. When the arrival probability is more than 0, for each deadline value generate one Proxel at next time step. But the sum of all success arrival Proxels' probabilities must be ensured equal to the current arrival probability. After each job arrives, reorder the queue to let the jobs whose deadline is closer stand more forward in the queue. The service and stay Proxels keep the same as in the original algorithm. The performance measures of each deadline level should be calculated separated from the total ones. It is better to merge about 5 time steps into a group since in this way the result is enough to observe the influence after adding deadline in different experiments and can save the result storage and calculation time. Similarly the adding deadline keeps the danger of

state space explosion if inappropriate parameters of the distribution function are set by users. The range of the parameters of each distribution can be found through experiments.

Until now, the implementations for adding three kinds of attributes are described completely. The priority seems the easiest attribute to add since it has not any relation with the simulation process. Although the other two attributes is more complex than priority, following the above adding process statements, they can also be added successfully. Certainly, the user interface needs some adjustment which is stated next.

3.3 User Interface

Besides the algorithm, the user interface is the other essential aspect which needs some corresponding alterations. After the alteration, the user interface is able to support users to conveniently choose the jobs with or without attributes and select the type of attribute as their demand. This concept of this section is the statement of user interface adjustment.

3.3.1 Preprocess of User Interface Adjustment

The original user interface has been shown in the example of Section 2.3.2. It contains three main parts, the option part for users to specify the queuing system and result part for reading transient and steady performance measures are in the main window, the image for plotting the transient results is represented in a separate window.

The implantation of adding attributes to jobs in the queue actually is adding an attachment to each job but not modify the structure of the whole queuing system model. Therefore the user interface does not need to be restructured. A small piece of place especially for attributes selection is enough. Adding attributes is a long-term work, and the current work in this thesis is just a start and an identification of adding attributes to the queuing system. In the future

work, the field about attributes must be improved and the types of attributes might be increased. For this reason, designing a separate window for choosing attributes is convenient for the future work and easy to adjust. Observing the original main window, since it contains both options part and results part, the main window already contains many elements. If directly add new elements of attributes to the main window, it will either become too crowded or the size will be enlarged too big. Therefore building a new window especially for specifying the job's attribute is an indispensable operation. However there must be a small element connecting the main window and the attributes window.

The elements of attributes window should be according to the types and implementations of attributes. Additionally the design for attributes window also should consider the options and convenience for users. On the basis of implementations of adding attributes which are discussed in Section 3.2, the manipulations that belong to users include adding attributes to jobs in the queue or not, selection one of the three attributes to add, furthermore if the deadline is chosen, the distribution function and parameters for deadline also should be specified by users. The adjustments to the original user interface and how the new elements work are shown in next section.

3.3.2 The User Interface Adjustments

Standing in the perspective of users, during the specifying the options of queuing system, the first step about attributes implementations is deciding whether adding attributes to jobs or not. So an element for this function should be added firstly.

The part which is marked by a red rectangular in Figure 3.5 is the new elements for users to decide the jobs with or without attributes in the specification queuing system step before executing each simulation. In the initial state, the check button in the left is empty and the button in the right side is unavailable. Users are able to select adding attributes to jobs by marking a hook into the check element so that the right hand "job attributes" button becomes available to click. The attributes window that shown in Figure 3.6 will jump out if the

button on the right side is clicked. The button is actually the connection between main window and attributes window.

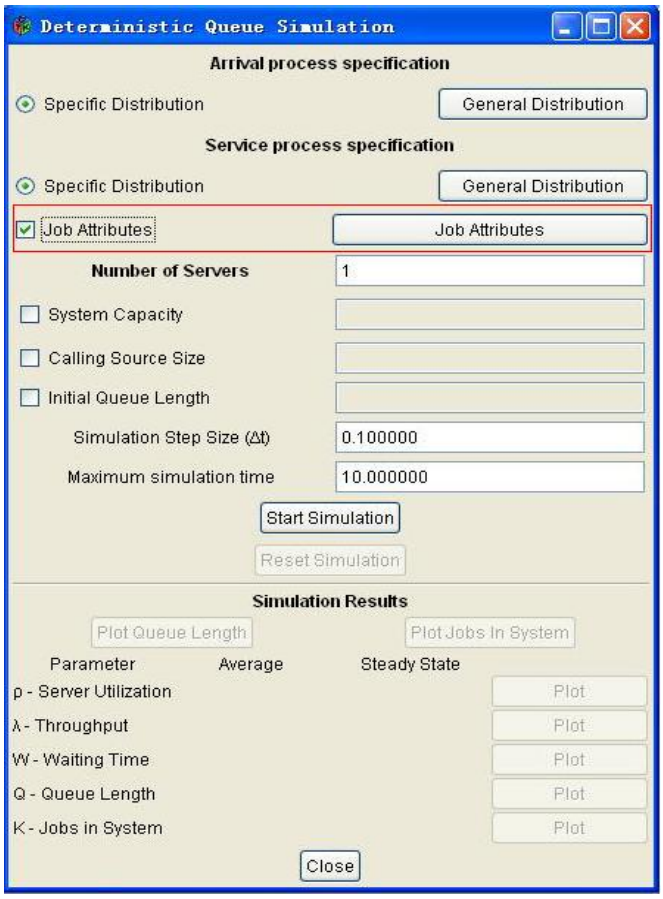

Figure 3.5 User Interface 1- New Elements in Main Window

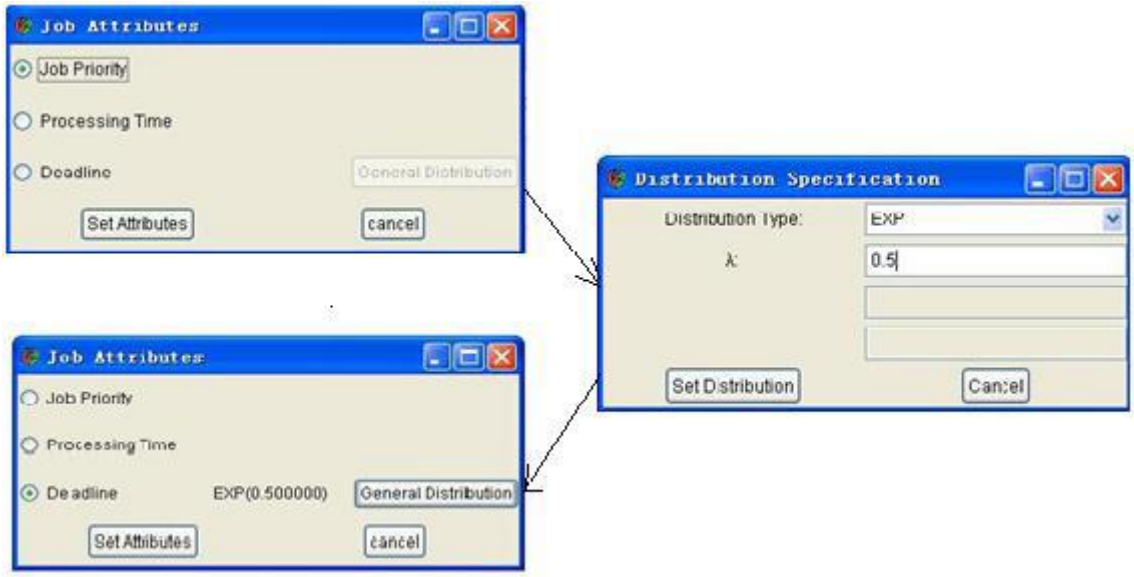

Figure 3.6 Attributes Window and Distribution Window for Deadline

The attributes window contains three elements to present the three attributes for users to choose: priority, processing time and deadline, which is shown in Figure 3.6. The priority one and processing time one are the same. Since the deadline contains a separate distribution, it is different from the other two. When users choose the jobs with deadline, the “General Distribution” button becomes available to be clicked so that the distribution window appears. After specification of the distribution and its parameters and confirming the settle by clicking “Set Distribution” button, the details of deadline distribution will be shown at the attributes window. The “Set Attributes” button is used for confirming the attribute selection at the final step of attributes arrangement. This paragraph of implementations instruction is according to steps which following the arrows in Figure 3.6. When the preparation of adding attributes is ready and other parameters of the queuing system are specified, the simulator can run for analyzing a queuing system with attributed jobs based on Proxel method.

So far, all alterations of the original algorithm and the user interface for adding attributes are depicted completely. Next chapter will prove the feasibility of the methods and the influence of adding attributes by making several different experiments.

Chapter 4

Experimental Verifications

In this chapter the experimentation results of adding attributes implementations are presented. A series of experiments are concentrated on adding priority to jobs in the queue, with the process of yielding analytical expressions for the performance measures and demonstrating the abilities of the tool. According to their focalizations, the experiments are classified as validation experiments and benchmark experiments. The satisfaction and dissatisfaction of adding job's attributes to Proxel-based queuing system are reflected by the results comparison and discussions.

The experiments are carried out with the following setup:

- *CPU Intel (R) Pentium (R) M Processor with 2.00 GHZ*
- *1.0 G RAM*
- *Microsoft (R) Windows (R) XP operation system*

The development tools are listed as following:

- *Dev C++ -Version 4.9.9.2* (Development Environment)
- *Glade -Version 2.12.1* (Interface Tool)
- *Microsoft Office Excel 2007* (Result Visualization)

4.1 Experiments Plan

Every theory is tested and verified by many different designed experiments before it is published. Especially in a simulation project, it is very easy to make errors of performance measures accuracy, assumptions for simulation and programming aspect. Since the existing Proxel-based queuing system simulator has been tested and its steady work ability has been verified, our work can be tested through directly comparing with original tool. It omits the process of verifying the validation of the conceptual queuing system by contrasting with real systems.

According to the goals of this thesis which are announced in Section 1.3, the experiments with different expectations are planed as following.

- Validation Experiment: Present the simulator is able to successfully provide a solution for the queuing system with attributed jobs.
- Benchmark Experiment 1: Variate the number of priority levels and observe the effect on the performance measures. Try to find a range of priority levels in which the simulator performs excellently.
- Benchmark Experiment 2: Variate the proportion of each priority's probability, present the results' variation and show the influence in different situations.

Because of the time limit of the thesis is close, it is very regretful that there is no enough time for the processing time and deadline experiments. But the methods of adding processing time and deadline have been described in detail, so the algorithms can be achieved definitely. Furthermore, since processing time, deadline and priority are parallel attributes of jobs in the queue; their implementations are similar with each other. The experiments of adding processing time and deadline also can be structured as priority ones

4.2 Validation Experiment: 2 Priority Levels

The expectation of this validation experiment is to prove priority can be successfully added into the jobs of the queue as one kind of attribute. The objective queuing system is the basic one M/M/1 which has a Markovian arrival and service process and a single server. The parameters of this queuing system are set as the list in Table 4.1. *Exp* is the abbreviation of Exponential distribution, the number inside the parentheses is the rate of the distribution function. This simulation will end until the maximum simulation time step 200=20/*dt* achieve.

Arrival process distribution	*Exp* (1.0)
Service process distribution	*Exp* (2.0)
Number of servers	1
Simulation step size (*dt*)	0.1
Maximum simulation time	20

Table 4.1 System Specification of Experiment 1

Figure 4.1 presents the both average and steady state performance measures results of two times simulation. The left image is the results of the jobs with no attributes and the right one is after adding 2 levels priority to the jobs. Observing from the images, the values of the both images are exactly the same. This is because what we altered is only the information inside each job unit but not the structure of the whole queuing system and the simulation way. The validation of non-attributed situation's results has been tested in the experiments part of paper [2]. Therefore performance measures of total jobs keep the same as before proves that after adding priority the simulator is able to provide correct analysis description for the queuing system. Adding priority causes no effect on the deterministic and accuracy of results.

Parameter	Average	Steady State
ρ - Server Utilization	0.477553	0.499758
λ - Throughput	0.950108	0.999503
W - Waiting Time	0.450407	0.498078
Q - Queue Length	0.427935	0.498078
K - Jobs in System	0.905488	0.997835

Parameter	Average	Steady State
ρ - Server Utilization	0.477553	0.499758
λ - Throughput	0.950108	0.999503
W - Waiting Time	0.450407	0.498078
Q - Queue Length	0.427935	0.498078
K - Jobs in System	0.905488	0.997836

Figure 4.1 Performance Measures of Experiment 1

Table 4.2 compares several statistics of before and after adding priority to the jobs. The total number of processed Proxels is significantly grows after adding priority. When jobs have no attributes, there is one job category. So only one arrival Proxel is created for the new job arrival event. The added priority classified the jobs into two categories: higher priority jobs and lower priority jobs. As a result, each arrival Proxel must split into two ones. The state space extends, correspondingly the total number of processed Proxels increases. The work load is forced to rise; therefore it costs more calculation time.

Statistics	no attributes	2 levels priority
Total number of Proxels processed	5163	72498
Max number of concurrent Proxels	49	561
Accumulated error	1.596e-010	4.742e-009
Computational time	0.031s	0.797s

Table 4.2 Statistics Comparison

Now, let's observe the especial results for different priority jobs which are shown in Figure 4.2. In order to describe conveniently, the jobs with higher priority are called P1-jobs, similarly P2-jobs represent lower priority jobs. The

values of each performance measures are presented by line graphs, with horizontal simulation time step coordinate and vertical probability coordinate.

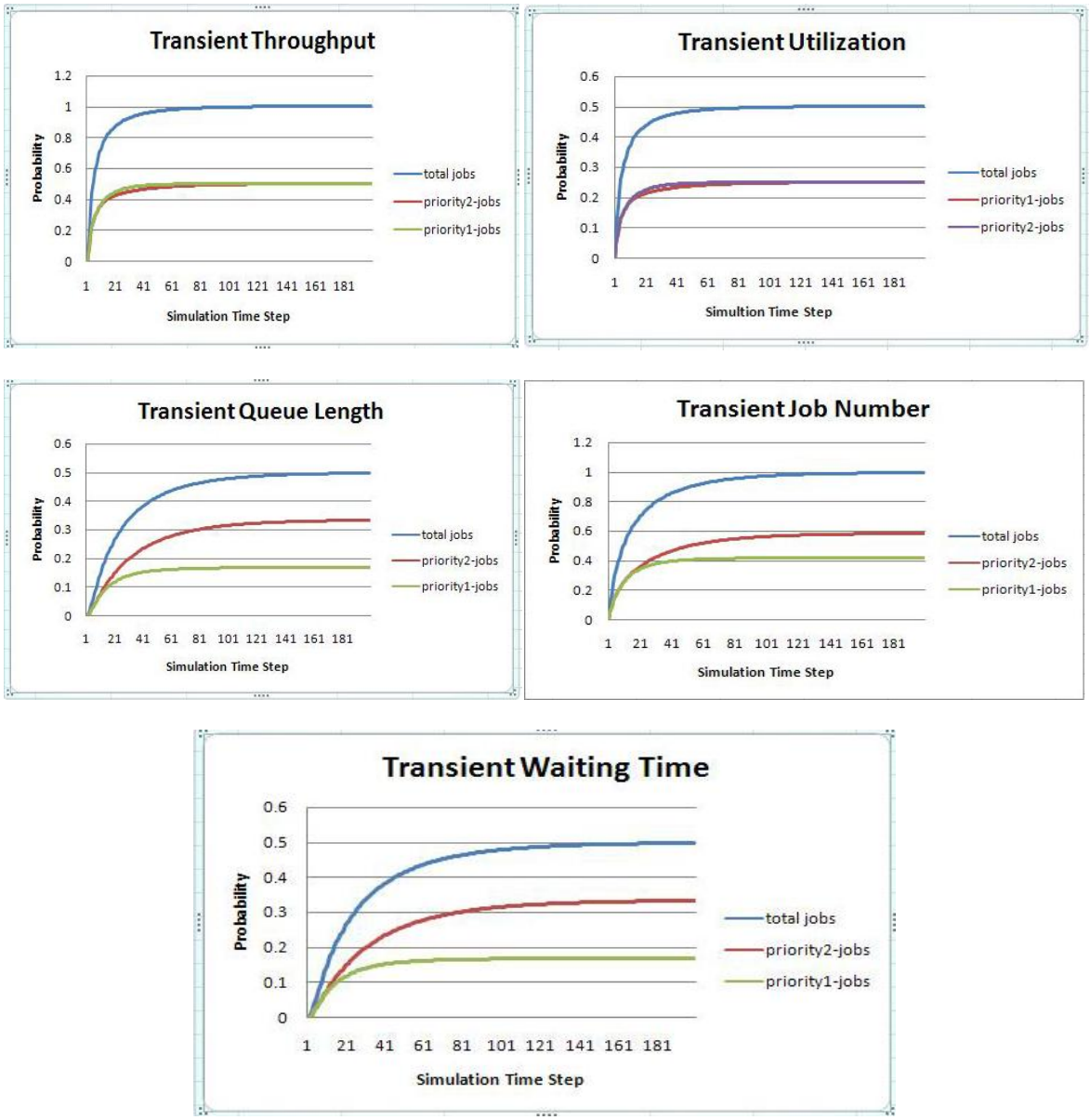

Figure 4.2 Performance Measures for Different Priority Jobs

Transient Throughput: Throughput is counted at the service finish moment, the sum of P1-jobs and P2-jobs equals to the total.

Transient Server Utilization: According to the throughput formula (detail can be found in paper [2]), the value is related to current busy servers' number and the probability value. Since this system has single server and both P1-jobs

P2-jobs share the arrival probability by 5:5, the transient values of server utilization are almost equal. The sum utilization of both two kinds of jobs equals to the total ones.

Transient Queue Length: Queue length is decided by the number of jobs in the queue. P1 jobs are always served before P2 jobs, so at every moment the P1 has fewer jobs than P2 in the queue.

Transient Jobs in System: This value is calculated as the sum of jobs in servers and the queue and in this experiment there is single server. P1 jobs are always finished first and leave the system. Obviously P1 has less value than P2.

Transient Waiting Time: Because P1 jobs are processed before P2 jobs, the waiting time of P1 is certainly less than P2. But the sum of the P1 and P2's waiting time still equals to total waiting time because of the time saved in P1 jobs is added into P2 jobs.

4.3 Benchmark Experiment 1: n Priority Levels

In this section we change the total levels of job's priority for observing the influence to the system performance. The system specification is as Table 4.3 describes. This is M/M/2 queuing system which is similar as Experiment 1 but with two servers. The level of priority is varied to 3,4,5,6,7,8,9 for test, and the results of performance measures are all the same which is represented in Figure 4.3. The correctness of analyzing the queuing system has not been influenced when the levels of priority increased. The Proxel-based queuing system simulator keeps its simulation ability as before.

Arrival process distribution	*Exp(1.0)*
Service process distribution	*Exp(2.0)*
Number of servers	2
Simulation step size (*dt*)	0.1
Maximum simulation time	20

Table 4.3 System Specification of Benchmark Experiment 1

Parameter	Average	Steady State
ρ - Server Utilization	0.096500	0.100000
λ - Throughput	0.973333	1.000000
W - Waiting Time	0.032287	0.033333
Q - Queue Length	0.031426	0.033333
K - Jobs in System	0.520592	0.533333

Figure 4.3 Performance Measures of Benchmark Experiment 1

Along with the levels of priority increase, the Proxel state space gets extension. The maximum Concurrent Proxels and calculation time are all increased. Although the accumulated error is added, it is still limited in an acceptable range. The most acute alteration is happened on the size of state space. Table 4.4 presents the extent caused by the change of priority levels. Figure 4.4 reveals a potential danger of state space explosion: when the level of priority is a large number or the simulation time is long, the system might go on strike because of the state storage problem. Therefore in a normal situation, it is better to set the job's priority in 10 levels. When the priority level is set to be more than 5, the simulation time around 20 is enough and good for this simulator to get into steady state.

Priority levels	Total number of Proxels	Maximum number of concurrent Proxels	Accumulated error	Computational Time
0	3915	32	2.283e-010	0.093s
2	73490	566	1.455e-008	0.656s
3	555419	4212	1.968e-007	5.156s
4	2413161	18153	1.367e-006	27.456s
5	7413928	54991	5.760e-006	95.437s
6	17836703	140755	1.806e-005	369.921s
7	39872912	381820	4.891e-005	846.711s
8	63783057	518677	9.283e-005	1553.828s
9	102721292	824124	1.687e-004	3097.547s

Table 4.4 Statistics Comparison with Different Priority Levels

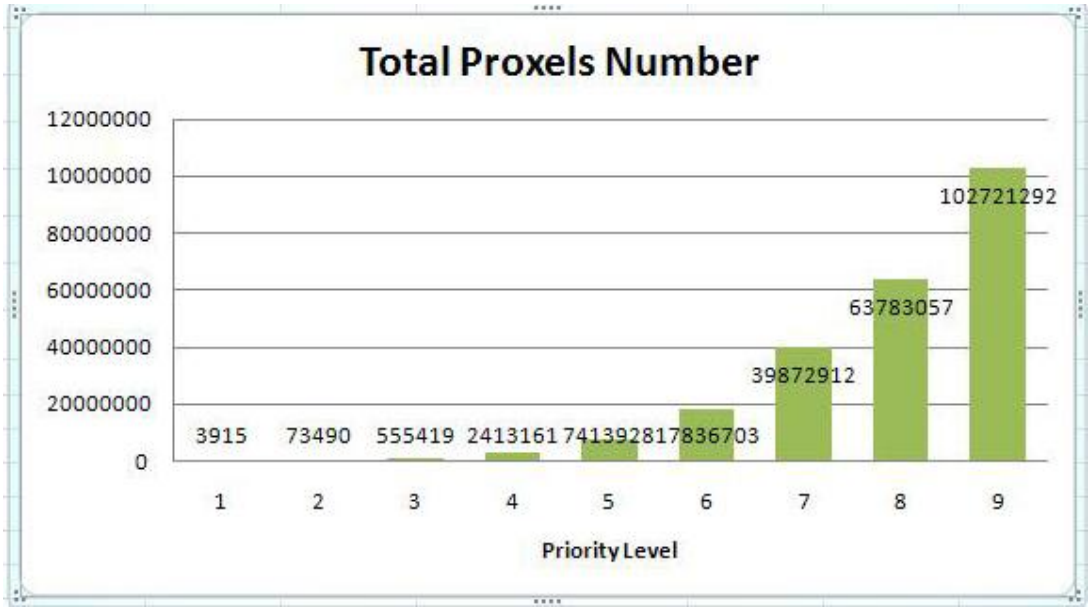

Figure 4.4 Comparison of Proxel Number

4.4 Benchmark Experiment 2: Variation of the Probability Proportion

Here the probabilities proportion of different priority Proxel in 2 levels priority queuing system is focused. In the validation experiment of Section 4.2, each arrival Proxel with a priority has the half value of an arrival event probability. In this section, the proportion of probability will be changed and show the influence on the queuing performance. The series experiments are implemented with the same queuing system G/M/c, which does not have an analytical solution in queuing theory. The parameters are listed in Table 4.5.

The probability proportion between P1 jobs and P2 jobs is varied from 1:9 to 9:1 and ensure the sum of the probability is 1.0. The results of performance measures are shown in Figure 4.5. Because after this series experiments, both the average and steady performance measures for total jobs are keep the same. Therefore the probability proportion also has no effect on the whole system

performance. The little variation of statistics which is shown in Table 4.6 is caused by the computational differences which are probably more cut off proxels, due to smaller probability values.

Arrival process distribution	*Norm*(1.2;0.1)
Service process distribution	*Exp* (0.5)
Number of servers	2
Simulation step size (*dt*)	0.1
Maximum simulation time	20

Table 4.5 System Specification of Benchmark Experiment 2

Parameter	Average	Steady State
ρ - Server Utilization	0.518361	0.671111
λ - Throughput	0.679860	0.810271
W - Waiting Time	0.623430	0.806797
Q - Queue Length	0.423845	0.806797
K - Jobs in System	1.798056	2.416726

Figure 4.5 Performance Measures of Benchmark Experiment 2

Priority levels	Total number of Proxels	Maximum number of concurrent Proxels	Accumulated error	Computational Time
1:9	373388	4897	4.286e-008	2.031s
2:8	422736	5665	4.246e-008	2.453s
3:7	448692	6530	4.328e-008	2.719s
4:6	463006	6759	4.231e-008	2.843s
5:5	468955	6719	4.138e-008	2.891s
6:4	467244	6648	4.175e-008	2.907s
7:3	457149	6907	4.293e-008	2.734s
8:2	435303	6223	4.365e-008	2.563s
9:1	389154	5269	4.449e-008	2.140s

Table 4.6 Statistics Comparison

Due to the non-Markovian arrival distribution, the results are not smooth. But the expected transient values can be captured by Proxel-based method. Two probability proportions are chosen as samples to observe the particular performance measures of P1 and P2 jobs. Every performance measures contrast between 3:7 and 7:3 probability proportions is compared in the following line graphs.

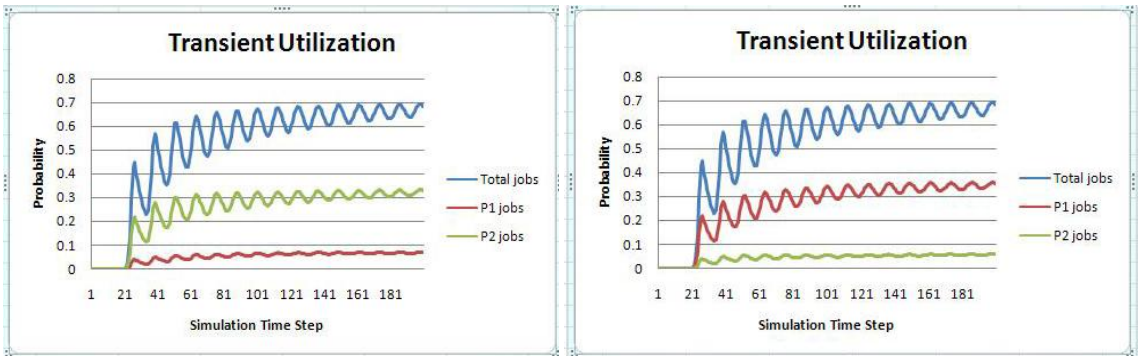

Figure 4.6 Transient Utilization (Left 3:7, Right 7:3)

Transient Server Utilization: Since the server utilization is computed as the busy servers multiply the probability of current simulation time step, when the probabilities of P1 and P2 jobs exchanged, the utilization value is exchanged correspondingly.

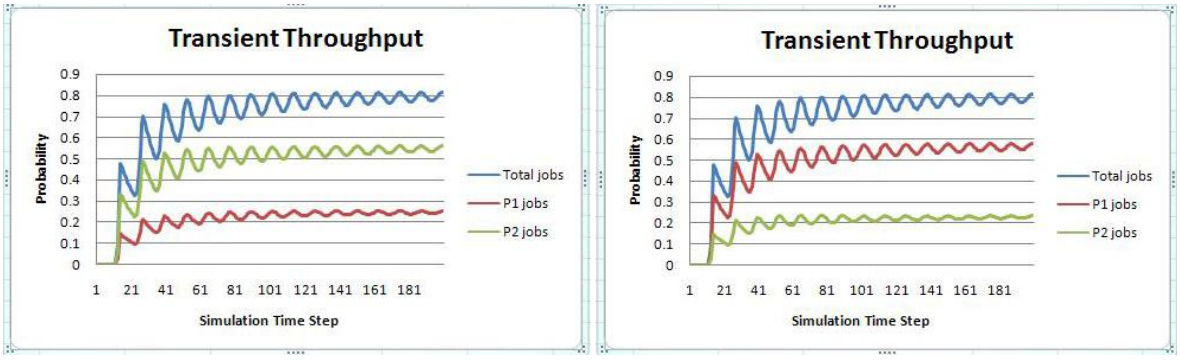

Figure 4.7 Transient Throughput (Left 3:7, Right 7:3)

Transient Throughput: The total throughput has no change however the proportion changes. But the particular throughput for P1 and P2 exchanged their values because it is only calculated at the service finished time point.

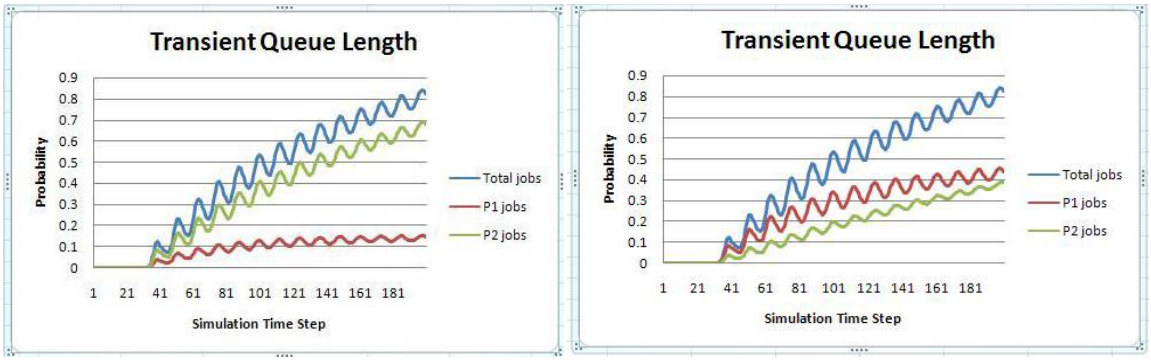

Figure 4.8 Transient Queue Length (Left 3:7, Right 7:3)

Transient Queue Length: When the proportion is 3:7, there is less probability for P1 jobs arrival. Moreover the P1 jobs are served before P2 jobs. Therefore the distance of queue lengths between P1 and P2 is much bigger than in the 7:3 situation.

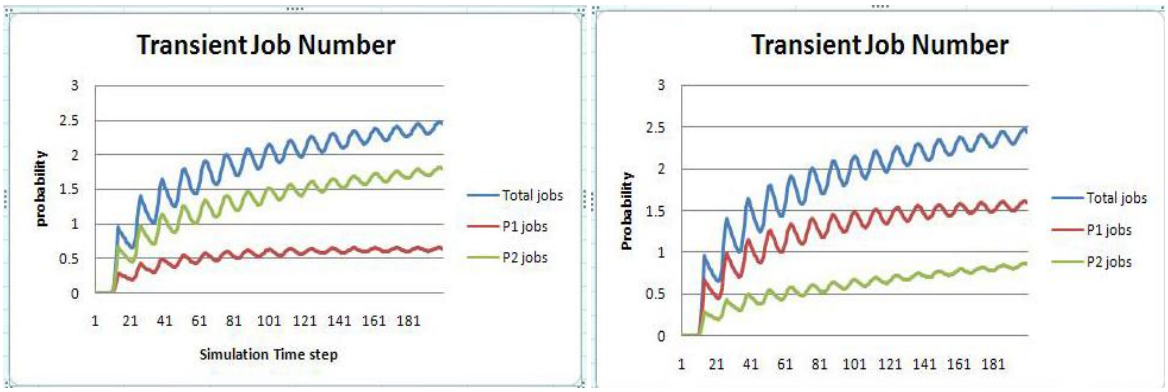

Figure 4.9 Transient Jobs in System (Left 3:7, Right 7:3)

Transient Jobs in System: When P1 has less arrival probability, fewer jobs arrival in the system. As a contrast, P1 occupies larger arrival probability and more jobs are contained in system.

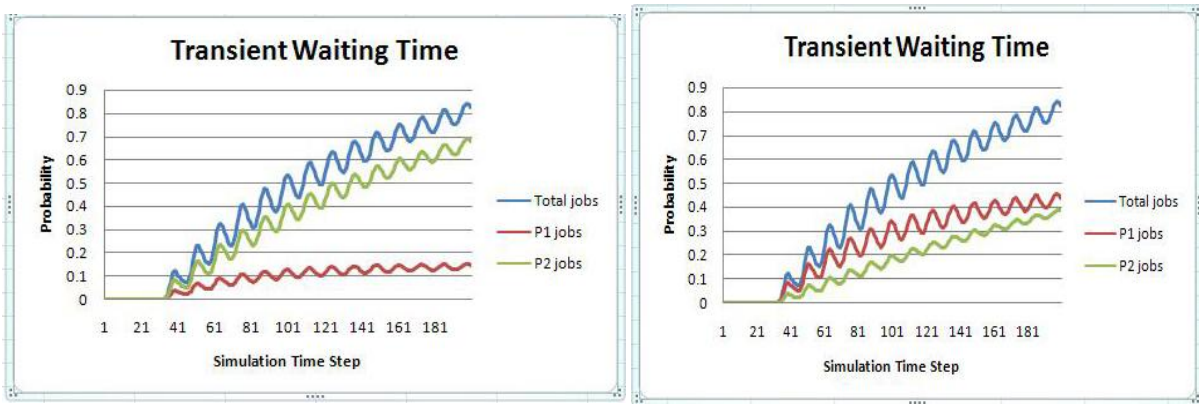

Figure 4.10 Transient Waiting Time (Left 3:7, Right 7:3)

Transient Waiting Time: In the 3:7 situation, less P1 jobs come and they are served more quickly than P2 jobs. P1 jobs cost much less time than P2 jobs. In the 7:3 situation, more P1 jobs come so they wait for longer time. Since P1 jobs cost more time, P2 jobs must wait for P1 to be finished, correspondingly they wait longer than before.

4.5 Discussion of the Results

Three kinds of experiments are carried out for their particular expectations. In validation experiment, the least priority levels are added in the most basic queuing system. The total results equal to the no attributes situation presents that the simulator is able to handle attributed jobs.

In benchmark experiment 1, the priority levels are varied from 2 to 9. Although the results keep the same, the statistics not regarding priority correspondingly

varied. The total processed Proxels number and the computational time increase significantly. When the priority level is more than 5, the Proxels number is huge and the computational time should be counted by minute but not second. When 9 levels priority are added, the system cost almost one hour in computation. As a suggestion for the simulator steady implementation, the priority level is better to be set below 10. Simulation time 20 is enough for the simulator to get a steady state and avoids the simulator to be immersed in a mass calculation when there are more than 5 priority levels.

When the priority level is determined, the probability proportion of each priority classification has no influence with the total performance measures. But the particular results of each priority classification changes according to the probability proportion.

The above results of experiments are valid for this implemented Proxel-based queuing system simulator, but not in general.

Chapter 5

Conclusions and Future Work

The summary of the entire thesis with contribution of the described work is stated in this chapter. This part is followed by the conclusions of the thesis, restrictions and subjects of future research in this direction.

5.1 Summary

Adding attributes is an essential extension part of Proxel-based queuing system simulator because it is the prerequisite of increasing number of queuing strategies that enable handled. Each Proxel presents the state of current time step and also carries adequate information of predicting the state of next time step. Including attributes information into Proxels in the main implementation of this thesis. Priority, processing time and deadline are the attributes what we intend to insert.

An array with a triple switch is the recommended data structure for storing job attributes. Adding attributes will extend both the size of each Proxel and number of Proxels. Choosing a suitable storage is one of the ways to handle state space explosion efficiently. After testing three different data structures, an array with a triple switch is the most suitable one because it is faster, smaller space occupation and usage flexibility for choosing attributes in

Proxel-based simulation. Therefore this data structure is selected to contribute its advantages to the attributes storage, insertion and deletion in our implementations.

Priority is one kind of jobs attributes and the value has no relation to the arrival and service process. Since the levels of priority are determined beforehand, the priority is the easiest one to be implemented among the three attributes. In each Proxel, jobs with different priorities are ordered in the array to make sure the job stand more forward whose priority is higher. Because the categories of jobs are increased according to their priority level, correspondingly for every job arrival event, Proxels are classified as the priority levels to simulate different priority jobs coming. Free servers get new job from the head of the array since the array is already sorted in an order. In every simulation time step, the Proxels for all possible states are generated until the maximum simulation time is achieved. Finally the simulator provides both transient and steady results of the system performance measures. Following this process the Proxel-based simulator completes a simulation of a Priority Queue and provides an analysis description for it.

The ways of adding processing time and deadline are also described in detail. Since they are similar with priority, the main implementations are similar with priority's. The distinguish is the calculation way of processing time and deadline values. Processing time is related to the service rate, so for each service distribution there is a particular value set for processing time. Deadline has a separate distribution for the value generation, and the calculation way is the same as processing time. But both of these two attributes have the danger of Proxel state space explosion which might be caused by the big value set. As a result of the feature of Proxel-based method, each value there should be one kind of Proxel. This problem can be solved by setting a cutoff point for each distribution, therefore the value sets of processing time and deadline are controlled suitable for this Proxel-based simulator. Finally the Shortest Job First and Deadline Scheduling are possibly added into the queuing strategies of the system successfully.

The validation of adding attributes is proved by the experiment of an M/M/1 queuing system with 2 priority level. Furthermore Benchmark Experiment 1 shows the statistics variation along with the priority level increase. The significant increase of total processed Proxel number reveals that the level of priority is better to be set below 10, otherwise the system may cost a long time at the computation or might cause a state space explosion. Benchmark Experiment 2 varies the probability proportion of 2 priority level jobs, the results of performance keeps the same presents that the proportion has no influence with the total result generation. The only change is the particular performance measures of jobs with each priority. The statistics change a little because of the computational difference.

The work of adding attributes to the Proxel-based queuing system simulator is organized as the data structure selection, algorithm description and speculated experiments. The complete substances of this thesis are depicted above as a summary.

5.2 Conclusions

5.2.1 Contributions

Jobs attributes can be successfully added into Proxel-based queuing simulation system. A refined approach of adding attributes and a flexible data structure for attributes storage have been developed. The selected attributes storage costs less time on the data implementations and smaller space for storing attributes than the other two data structures in such a Proxel-based queuing system, and it is also convenient for adding more attributes in the future research. The way of using integer data to represent priority, processing time and deadline makes the simulate operations and values calculation much more

concise. The probability of dangerous state space explosion is reduced by applying a cutoff point as the threshold in processing time and deadline values generation approach. A separate window is designed for jobs attributes specification. It is not only easy to implement but also flexible for alteration if more attributes are added in the future.

Compare to the results of simulation non-attributed jobs queuing system, the total number of processed Proxels and the computational time are significantly increased because of an extension of state space. But these shortcomings have been controlled in an acceptable extent by our designed ways of adding attributes. Through a series of experiments, the simulator is proved that it is able to provide analysis description to attributed jobs queuing system with Proxel-based method. Furthermore the values of performance measures in both transient and steady status still keep the deterministic and accuracy level as the original simulator. Generation of deterministic results with a higher accuracy is the major advantage of Proxel-based method compared to other methods; therefore this feature is maintained after adding attributes is the most powerful certification of the work's feasibility. The special performance measures for each job category with certain attribute present that the adapted Proxel-based tool simulates the queuing system with the desired queuing disciplines well. All in all, the approach of adding attributes described in this thesis is a suitable extension to the original Proxel-based queuing system simulator.

5.2.2 Restrictions

Now, the Proxel-based queuing system simulator is able to handle attributed jobs successfully and the influences after adding attributes are presented by several planned experiments. Currently all the phenomenon reveals that adding attributes is suitable improvement for the original simulator. But there are still some unsatisfied restrictions of adding attributes to jobs in the queue.

- **Restriction from state space problem**

The method of adding attributes which are depicted in this thesis can be implemented successfully, but the storage problem still can't be absolutely solved since it is the bottleneck of the Proxel-based method. To prevent the state space explosion, the levels of attribute can't be a huge number and this can be controlled by the developer of the tool. But there is some risk which depends on user's choice on the service and deadline distribution and parameters specification, for example if the interval between upper value and lower value of the Uniform distribution is too big; the system will face the danger of state space explosion. Since these restrictions are caused by the features of Proxel-based method, there is no absolute solution but the potential danger can be reduced by improvements in the future.

• **Restriction from objective environment**

The extension of state space must increase the work load of computer, so the computational time will definitely become longer. However, the cost time can also be reduced more or less by data structure improvement and computer setup advances. The experiments of processing time and deadline are the main regret; they are putting into the subjects of future work.

5.3 Outlook

This thesis can be seen as the beginning work in the queuing strategies development direction with the Proxel-based queuing system simulation. The current simulator still can't support all kinds of queuing systems and the implementations of adding attributes have lots of work which can be improved.

•**More queuing strategies:**

Currently only three attributes are considered to be added into the jobs in the queue so that the queuing system might be able to solve the queuing strategies according to the attributes. But in queuing theory, there are still more queuing disciplines have not been implemented corresponding to more different attributes. Therefore adding more attributes in parallel to the jobs is the essential subject on the way of extending the queuing system's range.

- **Improvement of current work:**

The processes of adding processing time and deadline are described to every detail; however the experiments verification of them should be put at the first position in the improvements. Without experiments, the unpredictable problems will not appear automatically. Additionally the attributes storage has the space for improvement, other data structures can be attempted in the future, such as an array with a hashing technology. A more suitable storage can handle the state space problem better and saving total processing time. The method of storing and calculating performance measures can also be improved in the future.

Using Proxel-based method in queuing system simulation is a new approach for both application of Proxel-based simulation method and queuing system performance analysis. Adding attributes significantly developed the basic simulator so that the tool supports more queuing systems simulation. Even though our work proves the feasibility and advantages of adding attributes, the implementations have limitations and potential risks. The direction of extend more queuing strategies need further studies and research work based on many experiments. Along with the development and improvement of the tool, the Proxel-based queuing simulator might occupy a more and more important role in the fields of analyzing queuing systems.

Reference

[1] Bergin Joseph. *Data Structure Programming: with the standard template in C++.* Springer, New York, 1998.

[2] Claudia Krull, Graham Horton. *Application of Proxels to Queuing Simulation* Otto-von-Guericke Universität Magdeburg

[3] C++ Programming. *http://www.cplusplus.com/doc/tutorial/*

[4] Donald Gross and Carl M.Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 3rd edition, 1998

[5] Dictionary of Algorithms and Data Structures: *http://www.nist.gov/dads/*

[6] Graham Horton. Slides of Lecture: *An Introduction to Simulation*. Otto-von-Guericke Universität Magdeburg

[7] Glade User Interface Design Manual. *http://glade.gnome.org/manual/index.html*

[8] Grunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. Queueing Networks and Markov Chains. John Wiley & Sons, New York, 1998.

[9] Ivo Adan and Jacques Resing. *Queueing Theory*. Eindhoven University of Technology. The Netherlands, 2001

[10] Probability Distribution. *http://en.wikipedia.org/wiki/Probability_distribution*

[11] Queueing Theory. *http://en.wikipedia.org/wiki/Queueing_theory*

[12] Sanja Lazarova-Molnar. *The Proxel-based Method: Formalisation, Analysis and Applications.* PhD thesis, Otto-von-Guericke Universität Magdeburg, November 2005