OFFLINE AND ONLINE DISK SCHEDULING PROBLEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

N. EVREN AŞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
OPERATIONAL RESEARCH

DECEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Çağlar Güven
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Haldun Süral
Supervisor

Examining Committee Members

Assoc. Prof.Dr. Levent Kandiller (METU, IE) _____

Assoc. Prof.Dr. Haldun Süral (METU, IE) _____

Asst. Prof.Dr. Z. Pelin Bayındır (METU, IE) _____

Asst. Prof.Dr. Esra Karasakal (METU, IE) _____

Asst. Prof.Dr. Osman Alp (Bilkent Unv., IE) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : N. Evren Aşan

Signature :

ABSTRACT

OFFLINE AND ONLINE DISK SCHEDULING PROBLEMS

Aşan, N. Evren
M.Sc., Operational Research
Supervisor: Assoc. Prof.Dr. Haldun Süral

December 2006, 126 pages

This thesis considers the disk scheduling problem. The problem is investigated in two types of settings: offline and online. We first adopt the traveling salesman problem with time windows in the scheduling literature for solving the offline problem. Then we develop a decision epoch scheme in which offline problems are iteratively used in solving the online problem. We perform an experimental study for our approach and two well-known disk scheduling algorithms, and compare them according to several performance criteria.

Keywords: Disk Scheduling, Online Problem, Offline Problem, Traveling Salesman Problem with Time Windows

ÖZ

ÇEVRİMİÇİ VE ÇEVRİMDIŞI DİSK ÇİZELGELEME PROBLEMLERİ

Aşan, N. Evren
Yüksek Lisans, Yöneylem Araştırması
Tez Yöneticisi: Doç. Dr. Haldun Süral

Aralık 2006, 126 sayfa

Bu tezde disk çizelgeleme problemi ele alınmaktadır. Problem, çevrimiçi ve çevrimdışı olmak üzere iki kurulum tipinde incelenmiştir. Çalışmada öncelikle, sıralama literatüründe yer alan zaman pencereli gezgin satıcı problemi, çevrimdışı problemin çözümü için adapte edilmiştir. Çevrimiçi problemin çözümü için adım adım çevrimdışı problemlerin çözüldüğü bir karar anı şeması geliştirilmiştir. Ayrıca, bizim yaklaşımımız ve iki adet iyi bilinen disk çizelgeleme algoritması için deneysel çalışmalar yapılmış ve bu algoritmaların çeşitli performans kriterleri için karşılaştırılması yapılmıştır.

Anahtar Kelimeler: Disk Çizelgeleme, Çevrimiçi Problem, Çevrimdışı Problem, Zaman Pencereli Gezgin Satıcı Problemi

To My Mother

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Haldun Süral for his positive attitude, guidance and encouragement. He is the best advisor one can experience the opportunity to work with and the best academician that I have ever known in my entire academic life. I also wish to thank my previous co-supervisor Adnan Şahin, who had guided me into this relatively less known area of study.

Thereto I would like to render special thanks to my dear friend Onur Aktuğ for his great support especially in code debugging phase, and my dear fellows Ekin Açıkgöz and Aysun Özen for their kind support and encouragements.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

This study is concerned with the computer hard disk workload scheduling problem. The motivation is to make better scheduling decisions so as to increase performance of the disk system while guaranteeing service stability. The disk system's nature is stochastic, where the read and/or write requests come randomly for service. Therefore, their arrival times and addresses are not known until requests are realized. The online workload scheduling problem handles the real system working principles, whereas the offline problem is an adaptation of that in which all the requests that will come to the system are assumed to be known at the beginning with their corresponding attributes; arrival time and request address on disk. Hence, the offline problem is a deterministic problem. In this study, both online and offline variants of the problem are considered by concentrating on "read" operation in disk scheduling.

The disk physical operation system consists of two main elements, the disk head and the platter (single or multiple). The head operates over the platter without touching it and make the read/write operations. Head moves in linear directions, as inward and outward movements on a single line (disk radius), while the platter rotates continuously without stopping and changing its direction. When a new job is to be serviced, the head first finds new job's track (circular line on disk or platter surface), then waits

the platter rotation until the starting location of the job address comes under the head. Head's movement is called **seek**, and once it finds the job's track, the time passed until the job's starting location comes under the head is called rotational **latency**. Because the disk rotates in single direction, the (time) distances between two job addresses (seek and latency) are not compatible with the Euclidean distances. The time distances are asymmetric and can be computed by considering the physical structure of hard disk. Seek and latency are realized as sequence dependent (setup) times for servicing two consecutive requests. When a job comes under the head after seek and latency, the time passed for processing (reading from or writing over track) of the job by the head is called **transfer time**, which is relatively short with respect to sequence dependent setup times. Seek, latency and transfer times constitute together **service time**.

Although the disk scheduling problem belongs to the computer science literature, its link with the machine scheduling is considered in this study. Hence, throughout the study, the machine scheduling terminology is used simultaneously, instead of related computer science terminology only. For instance, "flow time", "makespan", "job", etc. are occasionaly used instead of "response time", "service completion time", "request", etc., respectively.

The aim of the thesis is to develop a solution approach based on the solution of a variant of the traveling salesman problem (TSP) that could make further improvements in solving online problem over the proposed algorithms in literature. Despite the difficulties regarding solving TSP, such an approach is meaningful for the offline disk scheduling problem. A pure TSP application developed

for the offline version is used for solving online problem so that a benchmarking framework is provided to analyze the performances of online solution procedures. Incorporation of the TSP application into the online problem can be briefly explained as follows:

> At every decision time, we send the new queue information to the offline TSP model. Then, the offline application sends the scheduled queue information to the online application.

This pure TSP application for the online problem is analyzed and compared with two well-known algorithms, chosen for benchmarking. We also consider a heuristic approach to incorporate the TSP into the online problem.

Most of the well-known disk scheduling algorithms in literature focus on seek in the optimization. This study takes seek and latency together as the (sequence dependent) "**setup time**" and try to minimize the time needed to serve a set of requests. The main decision in disk scheduling is to decide the order (sequence) of jobs to be serviced in the system. There are two questions regarding main decision: "how to decide" and "when to decide". The first question is about the scheduling rule, which is that much of the literature is concentrate on. The second question is concerned with the instant that scheduling rule should be applied. The conventional approach for answering "when to decide" works as follows:

> Every time when the content of queue in the disk system changes with arrival of a new job or a timeout, the new queue information is sent to the decision maker (DM). Then, DM decides on the servicing sequence and sends back the scheduled queue to the online application.

The approach that we propose in this study, called as Deterministic Decision Epoch (DE), works as follows:

Queuing decisions are not made at every arrival and/or timeout cases. Instead, a decision epoch is defined as the time interval bounded by the completion time of the job in service up to the completion time of the next job. Hence, DM decides the new sequence with the information gained up to the completion time of the job under service.

Both approaches are taken into consideration in our study.

1.1 Motivation

Although the computer hardware have showed great evolution in little time, magnetic hard disks have not kept pace with them because of the physical constraints imposed by the system's structure. However, we believe that there is room in software improvement direction in hard disks, namely, the strategy in servicing the jobs in disk scheduling. It should be noted that there is always a tradeoff between hard disk performance and production cost. Although the marketing strategy decides always which one will win and how much the other will be ignored, we also believe that the software performance improvement increases the flexibility of the decision maker and affects production cost.

1.2 Problem Definition

In online disk scheduling, non-preemptive jobs coming randomly are directed to a queue having a capacity of holding **n** jobs. In addition to the queue (Q) of n jobs, there is a distinct queue just before the service having a capacity of two jobs, namely the run queue (RQ). RQ has a substantial effect on the disk performance

because once a job enters RQ, it becomes certain and the two jobs waiting in RQ are not subject to scheduling decisions anymore. Since the setup times of the jobs are sequence dependent, while a scheduling decision on Q is being made to determine the processing order of jobs, the address of the second job in RQ has an impact on the scheduling performance.

While deciding on the job processing sequence in Q, disk scheduling should also provide service stability by not allowing timeouts and queue depth (queue capacity) violations. Timeout refers the violation of maximum allowable time a job can stay in system before being serviced.

The online problem can be seen as a complex stochastic asymmetric traveling salesman problem with time windows. It follows that, the visiting sequence of all the available jobs in Q by the disk head must be scheduled considering stochastic problem nature, since the attributes, i.e. arrival times, processing times and addresses of the jobs are not known until they come. In offline setting, however, all the jobs are assumed to be known at the beginning with their corresponding attributes. Hence, the offline problem can immediately be reduced to an asymmetric traveling salesman problem with time windows. The visiting sequence of all the available jobs in Q by the disk head is scheduled subject to the time window of each job; starts with the arrival of job and lasts according to the timeout value of job.

Unfortunately, the TSP problem is shown to be NP-complete, so cannot be solved to optimality in “reasonable times”. Therefore, heuristics as approximation methods are needed to find good and fast solutions to the disk scheduling problems.

1.3 Outline of the Chapters

In Chapter 2, a brief literature review on related subjects is done. In the first section online problems and algorithms are reviewed. In the second section, a brief literature survey of routing problems under online setting in addition to more general offline setting is done. Third and fourth sections are about the disk scheduling. In the third section, the classical algorithms for disk scheduling are given with their variations. In the fourth section, further adaptations/modifications on those traditional methods and studies other than those are considered.

In Chapter 3, the computer hard disk system is explained in detail in the first section. After that, the disk scheduling problem is defined with several performance measures, and the possible objectives are briefly discussed. In the third section, a generic mathematical model developed for the offline problem for several objective functions is introduced. The generic offline problem formulation is a variation of the formulation of the TSP with time windows. In the fourth section, the computational difficulty of solving the model with a general purpose optimization package is illustrated with an experiment.

The online problem is examined in Chapter 4. The decision making issues and the policy developed are discussed in the separate sections. Conventional approach and deterministic decision epoch scheme developed for solving the online problem are detailed in these sections. After that, several reschedule procedures are discussed, including the TSP-based approach described in Chapter 3. In addition to the exact solution of the sequence dependent

makespan minimization problem, a heuristic reschedule procedure based on the Nearest Neighbor is also provided at the end of the chapter.

Chapter 5 covers the implementation of simulation under conventional and decision epoch settings, and the experimental analysis of the disk scheduling algorithms under both settings. In the first section, the nature of the simulation with basic properties is mentioned. The code frameworks of simulation are given for both settings in the second section. The parameter settings of the experiments, and the test bed generation methods are discussed in the third section. The experimental results for five policies on our test bed are provided and several comparisons are presented in the last section.

The last chapter, in addition to conclusion of the study done, provides further study subjects. Also a challenging future research subject is mentioned in Appendix K.

CHAPTER 2

LITERATURE REVIEW

Computer hard disk scheduling problems have been studied both in Operational Research and Computer Science literatures. Besides having direct application on computer industry, it has been widely studied by the hardware companies. Therefore, one can find numerous technical reports, although much of them being stayed secret because of competitive concerns.

This chapter is structured as follows. In Section 2.1, our literature review and analysis of online algorithms are presented. In Section 2.2, a brief review on traveling salesman problem and its derivative vehicle routing problem will be given both in offline and online settings. Classical disk scheduling algorithms are presented in Section 2.3. In Section 2.4, some adaptations from traditional policies, aiming improvements in several disk scheduling performance measures are discussed. One special remark about third and fourth sections is that, since the studies in literature usually make the experimentation with their own simulation framework including the disk geometry (necessitating some probabilistic assumptions as miss probability etc.), their results are valid within themselves. The results from one study for a certain algorithm cannot be compared directly with the results in the other study. Hardware differences in experimentations also support that remark.

2.1 Online Problems and Algorithms

In an online problem, decision maker (DM) has to deal with limitations on information instead of limitations on computational power in offline problems (Correa and Wagner, 2005). DM has to decide with incomplete information which continues to be available in time as increments. DM has a short time to decide at every new event (for example, in present disk scheduling literature, a new event is completion of a job or arrival of a new job coming for service with its corresponding attributes) with the information at hand at the time of the event without knowledge of future information.

In late eighties and early nineties, there were three basic online problem types studied extensively: the paging problem, the k-server problem and metrical task systems.

- *"The paging problem is to maintain a two-level memory system consisting of a small fast memory and a large slow memory, to serve a sequence of requests to memory pages so as to minimize the number of page faults incurred."* (Albers and Leonardi, 1999). The most well known deterministic online strategies for paging are Least Recently Used, First-In-First-Out, and Least Frequently Used (Winter and Zimmerman, 1998).
- *"The k-server problem generalizes paging as well as more general caching problems. The problem consists in scheduling the motion of k mobile servers that reside on points of a metric space S. Requests are issued at points in S and, in response to each request, one of the servers must be sent to that point.*

The goal is to minimize the total distance traveled by all the servers."

- *"Metrical task systems can model a wide class of online problems. A metrical task system consists of a pair (S; d), where S is a set of n states and d is a cost matrix satisfying the triangle inequality. Entry d(i; j) is the cost of changing from state i to state j. A task system must serve a sequence of tasks with low total cost."* (Albers and Leonardi, 1999).

After early nineties, the online problems appeared in a wide range of application areas such as distributed data management, scheduling and load balancing, routing, robotics, financial games, graph theory, and a number of problems arising in computer systems like disk scheduling (Albers and Leonardi, 1999).

To cope with online problems, online algorithms are developed. In general, an online algorithm takes input at increments one by one. Then it generates output for each of the input piece taken without the knowledge of the future input. For testing the performance of an online algorithm, a useful tool, called competitive analysis suggested by Sleator and Tarjan (1985) has been widely used. The main idea of the competitive analysis is to compare the computational cost incurred by an online algorithm with that of the offline algorithm in which the full input knowledge is available at the beginning. Considering a minimization problem and letting ALG(I) be the cost incurred by online algorithm ALG on instance I, the competitive ratio of ALG is defined as follows:

$$\sup_{I} \frac{ALG(I)}{OPT(I)}$$

where OPT(I) is the optimal solution found for instance I by an offline algorithm. Lower that worst case ratio is better the online

algorithm (Bonifaci, 2005). This analysis makes the offline variant of the online problem valuable.

2.2 Offline and Online Routing Problems

Easy to describe and hard to solve Traveling Salesman Problem (TSP), has become popular since a method for solving it is published in 1954. Basically the problem is to find a least costly tour for a salesman visiting each of the n customer locations once and returning to the starting point, where the cost of traveling from one location to another is given. TSP belongs to the class of combinatorial optimization problems known as NP-complete. In this case no one can expect to develop an algorithm that guarantees to find the optimal solution to TSP in polynomial time. Hence, to tackle that difficulty, applying heuristics for finding good feasible solutions in reasonable times is a common approach. Readers are referred to Reinelt (1994) for a TSP review and its practical solution procedures.

Although the most approaches in the routing area assume an offline setting in which the input is entirely known beforehand, several real life cases necessitate an online point of view. For instance, the advancement in information technology makes the just-in-time management more important. Express transshipment necessity fed also by rapid growth of e-commerce led to a sudden increase in real time routing problems, where problem size and parameters change after the vehicle routes are constructed (Chang et al., 2003). Similarly, in a logistics system, the mobile service provider cannot exactly know the costs at the time of travel and even could not know where the next location s/he must visit.

Therefore, in the online version the locations to visit are told to the salesperson while s/he is traveling. So that, every request has a release time r_i, which indicates the time when the location is available to visit and/or also a due date d_i, where i is the location or customer identity. The objective function is given by the time at which all the requests are served. It is the same as the makespan criterion in machine scheduling terminology. In online setting of TSP, if the salesperson is not required to return its starting point, this version is called as Nomadic Online TSP. The version in which the tour is required to be closed is called as Homing Online TSP (Bonifaci, 2005).

Vehicle Routing Problem (VRP) is a generalization of TSP allowing more than one server (salesperson) and bringing the vehicle capacity and demands of the locations into the problem. Because of the suitability of the online problem setting to logistics systems, and the central importance of VRP to logistics systems, the online VRP is also a widely studied subject in the literature. In online VRP, the order of visit to known customer locations are decided on real time without knowing the possibilities of demand changes in locations and even the new customer locations that may come into the picture while the vehicles keep serving the available locations at that time. Hence, online VRP has applications on to dial-a-ride systems, such as controlling a taxi station or an elevator set of a building, or planning routes for a set of couriers (Bonifaci, 2005). The disk scheduling problem therefore can also be seen as a dynamic, stochastic online routing problem having asymmetric characteristics.

2.3 Classical Disk Scheduling Algorithms and Their Variations

After the first commercial hard disks were introduced into the market by IBM (IBM 350 RAMAC disk drive, 5 megabyte) in 1956, the magnetic hard disks have had little evolution comparing with the processors and other hardware components since the main logic of disk system is physically the same with the very first hard disk. The hard disks are also in their technological limits in storage capacity. When the platter density is held at the reached level of today's technology, space requirement increases for greater data capacity, whereas the difficulty of creating smaller heads arises for higher data density platters. Decreasing the magnetic domain makes hard to sustain the disk stability. However, the job service decisions have been always extremely important for overall hard disk performance. In the beginning there was one simple scheduling policy for job service decisions: First-Come-First-Served (FCFS). Although this is a simple scheduling method, it is highly resistible to starvation, since a new coming job has to wait only for the jobs that came before it. While FCFS (or FIFO) scheduling policy had been applied in early hard disks, when the end of sixties came, several intelligent algorithms for hard disk scheduling had already found application in the field.

Shortest Seek Time First (SSTF) is a form of simple Shortest Time First (STF) algorithm. Since the seek time (rather than latency) constitutes greater part of the access time, looking for the shortest seek time job generally makes considerable improvement over FCFS job scheduling. However, SSTF is very susceptible to starvation. Especially in high workload situations, some jobs could wait to be serviced for an unacceptably long time.

SCAN algorithm which is first proposed by Denning (1967), is a clever modification of SSTF, which tries to overcome the starvation illness. It works as such that the head starts to scan the disk surface from the outermost cylinder (cylinder 0) and serves the jobs inward. When it comes to the innermost cylinder (cylinder n), it starts to scan outwards and makes the same process outward. In this case, the middle parts of the disk are serviced well, while the innermost and outermost parts are serviced relatively poorly. By this method, the maximum waiting time of a new coming job, at the worst case, onto the just passed cylinder 0 or n, is the time equivalent of two times the *cylinder 0 – cylinder n* distance.

In SSTF and SCAN algorithms, there is a slight possibility of starvation. An example from Stallings (2001) shows that "*if one or a few processes have high access rates to one track, they can monopolize the entire device by repeated requests to that track. High-density multisurface disks are more likely to be affected by this characteristic than lower-density disks and/or disks with only one or two surfaces.*" To overcome this possibility a combination of FCFS with any of the algorithms above can be applied like that:
The jobs waiting for service are segmented in queues of defined length, say N. After the chosen algorithm is used for the N jobs in first queue, the same thing is done for the second whether its length is higher or lower than N. While N gets higher and higher, N-step-SCAN approaches SCAN in performance (Stallings, 2001).

SCAN is further modified to overcome the service anomaly mentioned above and to decrease the maximum waiting time of a new coming job. In C-SCAN, the scan direction is always the same, meaning that when the disk head finishes the scanning (i.e. inward), it comes back to the outermost cylinder and starts to the

same process again, and so on. In this case, as opposed to SCAN, all the parts of the disk are equally serviced. By this method, the maximum waiting time of a new job, at the worst case, onto the just passed cylinder 0 or n, is the time equivalent of the *cylinder 0 – cylinder n* distance (half of that of SCAN algorithm).

Another modification of the SCAN algorithm is called LOOK. The only difference of LOOK is that, while the disk head scans the disk surface from one end to the other, it turns scanning direction instead of going to the end when there is no job at the current direction forehead. Although the worst case maximum waiting time of a job is the same, the average waiting time of a job is shorter in LOOK than that of C-SCAN.

CLOOK is a combination of C-SCAN and LOOK algorithms. Only difference between CLOOK and LOOK is the single direction character of the first one. Although this character does not guarantee the equal service throughout the disk surface like it does in the case of C-SCAN algorithm, it increases the uniformity of service.

The disk scheduling algorithms other than FIFO can be examined under two main categories; seek delay reduction policies and positioning delay reduction policies. Under the first heading SSTF, SCAN, C-SCAN, LOOK and CLOOK algorithms take seek into consideration, while shortest-positioning-time-first (SPTF) algorithm takes combined seek and rotational latency into consideration. SPTF use the more complete information about the data blocks on disk and the current position of head and choose the job with minimum positioning delay. In Table 2.1, we present an overview of the disk scheduling algorithms.

Table 2.1 Overview of Disk Scheduling Algorithms

Algorithm	**Main Idea**	**Remarks**
FIFO	Jobs are taken into service in their coming order.	Poor throughput and long average waiting time. But highly resistable to starvation.
SSTF	Seek reducing algorithm. The next job to be processed is chosen as the nearest job in seek distance to the job under service.	Good performance on throughput and average waiting time. Very susceptible to starvation especially in high workload situations.
SCAN	A modification of SSTF. It applies SSTF in one direction of disk, when head comes to the edge it reverses the direction and makes a new SSTF application on reverse direction until the edge.	Inferior performance on throughput and average waiting time. But little starvation.
C-SCAN	Cyclic SCAN makes the head run only in single direction and when it comes to the edge it returns to the starting point and redo the run in same direction.	Inferior performance on throughput and average waiting time. Less starvation than SCAN.
LOOK	A modification of SCAN. While the head goes in a direction, if there is no job in that direction, it instantaneously reverses the direction instead of going to the disk surface's edge and apply SSTF in new direction.	Slightly better performance than C-SCAN on average waiting time. Almost same starvation with C-SCAN.
CLOOK	A modification of LOOK to apply it in single direction only.	Almost the same performance with LOOK on throughput and average waiting time. Increased service uniformity.
SPTF	Considering current position of head finds the next position with minimum seek and rotational latency.	Necessitates more complete information of data blocks on disk surface. Good performance on average waiting time especially.

Teorey and Pinkerton (1972) investigate and compare the performances of FIFO, three seek time optimization policies SSTF, SCAN and N-step scan, and one early rotational position optimization policy (Eschenbach scheme). They also consider LOOK and CLOOK. They made both analytical and computational performance comparisons in this study. The computational performance comparisons are done with the aid of a utility function combining system throughput with mean and variance of waiting time for individual requests. They found that under light workload the best performance was attained by LOOK. Under heavy workload CLOOK, combining the best characteristics of LOOK and the Eschenbach scheme, had maximum performance.

Worthington et al. (1994) is almost the most cited work in disk scheduling area, since they introduce prefetching (on-board cache utilization) concept into the basic scheduling policies. In prefetching, the head is allowed to make excess read on the track after completion of the read of a request at that track when the scheduling structure permits. That excess reads are stored in onboard cache for a time and they are called when a new request's part of the address overlaps the stored one. Hence, for that part of the address, head does not have to make read operation. They made simulations including prefetching and reduced inferences on quite good performance increase in certain situations. For example, while CLOOK shows slightly inferior performance than SSTF and LOOK for random workloads, it achieves the highest cache hit rates and lowest average response time for most of the real-world traces.

2.4 More Study on Disk Scheduling

Geist and Daniel (1987) have introduced VSCAN, a continuum of two disk scheduling algorithms SSTF and SCAN. "*A continuum of disk scheduling algorithms, V(R), is defined* where R is a variable having values between 0 and 1 which defines the algorithm's closeness to SSTF and SCAN. *V(R) has endpoints of V(0) = SSTF and V(1) = SCAN. V(R) maintains a current SCAN direction (in or out) and services next the request with the smallest effective distance. The effective distance of a request that lies in the current direction is its physical distance (in cylinders) from the head. The effective distance of a request in the opposite direction is its physical distance plus R x (total number of cylinders on the disk). This definitional continuum also provides a continuum in performance, both with respect to the mean and with respect to the standard deviation of request waiting time."* After tests with a real system data, they found that V(0.2) outperforms FIFO, SSTF and SCAN algorithms in average waiting time and system throughput criteria.

Seltzer et al. (1990) have analyzed the traditional disk scheduling policies in the presence of long queue lengths, and they proposed two algorithms taking rotational latency into account together with seek time. These algorithms were grouped as shortest time first (GSTF) and weighted shortest time first (WSTF). First one was a combination of SCAN technique with shortest time first (STF) technique. The second one, which guarantees no starvation, made use of the STF technique applying an aging function to the computed times. They used an additional disk scheduling performance measure other than average flow time, namely "disk utilization". It is defined as the percent of a job's flow time (waiting

time + seek + latency + transfer time) that was composed of the transfer time. The utilization value for FIFO has been found about 7%, while proposed algorithm GSTF was shown to have utilization close to STF (25%), it also have had comparatively little maximum flow time, close to that of C-SCAN.

An HP Laboratories technical report by Jacobson and Wilkes (1991) has also showed that the access time based algorithms (those taking latency into consideration) outperform seek time based ones. They have found that aged shortest access time first algorithm (a continuum between FIFO and shortest access time first (SATF)), having quite a same logic with VSCAN, has better performance within all the variations of SATF policy.

Thomasian and Liu (2002), apply a modification on basic disk scheduling algorithms for incorporating a lookahead of next i requests (LAi) property into them. They apply it to C-SCAN by taking the latency into account and reorder the next i requests in scanning direction to minimize the sum of their service times, instead of minimizing that of just next one (C-SCAN-LAi). They also apply the same logic to SATF considering again i requests rather than just one at a time (SATF-LAi). They make a random number driven simulation study for comparing the performances of classical policies with the two lookahead policies mentioned above. For the performance differences in between the classical policies, their results concur to that by Worthington et al. (1994). When the mean response time criterion is considered, SATF is the best while FIFO is the worst, SSTF and SCAN outperform C-SCAN policy. SATF-LA2 further improves over SATF.

Modern hard disks using rotational position optimization algorithms, utilize seek distance versus rotational distance tables (rpo tables or arrays), which are stored in flash-memory within each hard disk drive. Hence, reduction in the necessary flash-memory, directly reduce the disk production cost. The trade off is that how much it can be reduced with no or little degradation in hard disk performance (Burkhard and Palmer, 2001).

There are some studies on the synthetic workload generation and the validities of them. Ganger (1995) has examined several probability distributions, and compared them with real workload sets known in literature. He showed that the commonly assumed workload characteristics were inaccurate and especially the job arrival patterns were not independently distributed in reality.

Huang and Chiueh (2002) discuss another possible drawback of disk scheduling efforts. They claim that software based (shortest access time first) disk schedulers are becoming less and less feasible as the disk technology evolves, since the disks are getting more and more complicated.

Another approach by Popovici, Arpaci-Dusseus (2003) involves a simulation approach integrated to a real disk system. It traces the jobs and it models the service time of job by observing both request type and the logical distance from the previous request. Thus by predicting the near behavior with the past requests having same attributes.

Reuther and Pohlack (2003) suggest an algorithm based on dynamic active subset (DAS). DAS contains the most outstanding requests and it is updated after every scheduling decision. These

requests have higher priority and are scheduled according to the rotational position of the requests. So that the finite time service guarantees could be reached for every job without deterioration in performance.

Andrews et al. (2002) concern disk geometry directly and propose new algorithms for offline and online problems. They show that the problem is related to the asymmetric TSP. They define a reachability function, which gives the maximum radial distance head can travel for a rotation of angle θ. Then using this function, they develop an approximation algorithm to serve all the requests on disk within certain number of rotations depending on the reachability function. Finally they apply the idea to the online problem. They proposed an algorithm, called as CHAIN. It has similar logic with the classical STF algorithm, with key difference of better look-ahead. CHAIN considers more than the next request and forms a partial order of all the requests in the buffer. Then, it constructs a new partial order at every arrival to the buffer. However, they do not present an analytical comparison between STF and CHAIN's performances.

CHAPTER 3

PROBLEM DEFINITION AND OFFLINE DISK SCHEDULING PROBLEM FORMULATION

In this chapter the hard disk system basics are explained. After that, the general disk scheduling problem is defined with the main performance criteria. The basic mathematical models for offline problem, which is described as a TSP with time windows, are given for four different objective functions at the last section.

3.1 System Definition

A disk is a platter, made of metal or plastic with a magnetizable coating on it, and in circular shape. It is possible to store information by recording it magnetically on the platters. A conducting coil, called **head**, which is a relatively small device, facilitates the data recording on and retrieval from the disk. In a disk system, head rotates just above both surfaces of each platter. All heads, being attached to a disk arm, move collectively as a unit. To enable a read and write operation, the platter rotates beneath the stationary head.

Data are organized on the platter in tracks, which are in the form of concentric set of rings. In medias using constant linear velocity, the track densities are uniform (bits per linear inch of track). The outermost zone has about 40 percent more sectors than innermost

zone. The rotation speed increases as the head moves from the outer to the inner tracks to keep the same data transfer rate. This method is also used in CD-ROM and DVD-ROM drives. In these types of medias, the storage capacity of the disk is maximized by zoning application. A zone consists of adjacent cylinders having the same track densities (sector per track).

The other types of medias have constant disk rotation speeds, and in that kind of systems the same numbers of bits are typically stored on each track, thus the density, in bits per linear inch, increases in moving from the outermost track (track 0) to the innermost track (track N), to keep the data rate constant (constant angular velocity).

Just as the tracks are subdivisions of the platter surface, tracks have subdivisions, called sectors, which are depicted in Figure 3.1. Data are transferred to and from the disk in blocks, size of which are typically smaller than the capacity of the track. Block-size regions on the disk where data are recorded, are called sectors each having 512 bytes capacity for most disk drives. The request locations are defined with the physical block addresses over these sectors. Adjacent sectors are separated by intratrack gaps in order to avoid imposing unreasonable precision requirements on the system (Stallings, 2001).

A common disk drive has a capacity in the size of gigabytes. While the set of tracks that are at one arm position forms a cylinder, in a disk drive there may be thousands of concentric cylinders. A set of wires, called the I/O bus, attaches the disk drive to a computer. Buses vary in kind from earlier advanced technology attachment (ATA) and small computer system interface (SCSI) to Serial ATA

(SATA) and SATA II buses available for basic consumer use. While the consumer type hard disks having SATA and SATA II buses use native command queuing (NCQ), allowing the queuing of up to 32 jobs, the enterprise disks having SCSI-2 standard use tagged command queuing (TCQ), which supports up to 216 queued commands. In NCQ all the requests in queue have the same importance, whereas in TCQ it is permitted to assign high priority to some of the jobs. Data transfer is carried out through special electronic processors, which are called controllers. While the controller at the computer end is called ***host controller***, a ***disk controller*** is built on each disk drive.

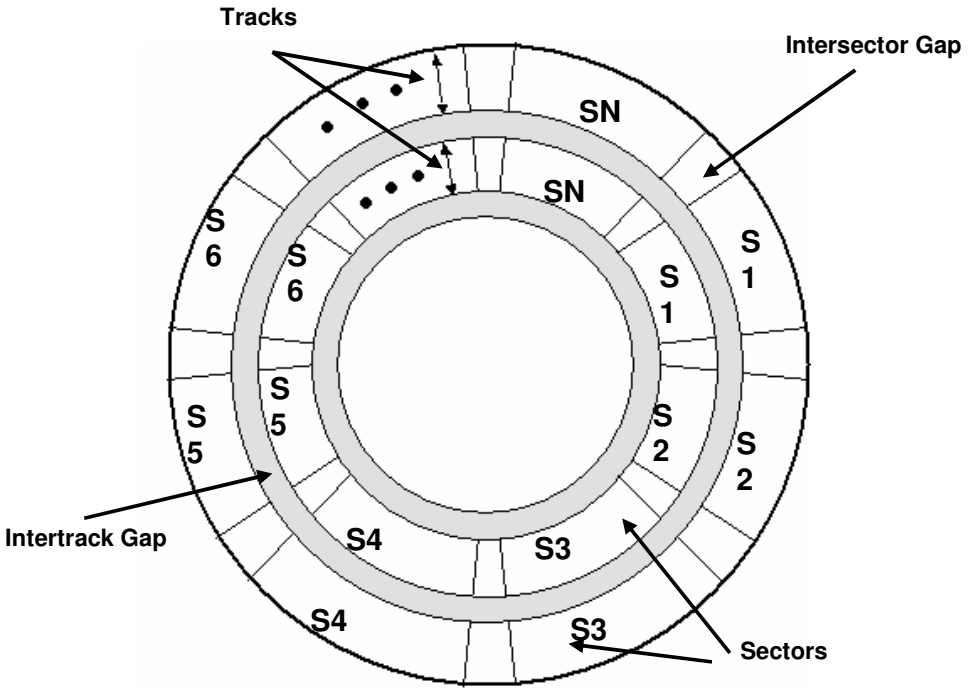

Figure 3.1 – Disk Surface

A command placed into the host controller by the computer initiates an I/O operation in the disk. For this, the computer uses

memory-mapped I/O ports and sends the command through messages to the disk controller, which in turn operates the disk-drive hardware to carry out the command. Disk drive transfers data through the interaction of built-in cache, which most disk controllers have, and the disk surface. Data transfer between the cache and the host controller is performed at high electronic speeds (Silberschatz, Galvin, Gagne 2003).

Most disk drives rotate 60 to 200 times per second, with the help of a high speed drive motor. It is possible to mention two components of the disk speed: **transfer time** and **positioning time** or random access time. Transfer time is determined by the transfer rate, the rate at which data flow between the drive and the computer. The positioning time, which refers to the time that is elapsed to move the disk arm to the desired cylinder, is also called as the **seek** time. One term to be mentioned here is the **rotational latency**: The time for the desired sector to rotate to the disk head. Typically rotational latencies and seek times of disks are in the range of several milliseconds, and they can transfer several megabytes of data per second (Silberschatz, Galvin, Gagne 2003).

3.2 General Disk Scheduling Problem

Whether the disk is a single or multiple platter type, it always works with the same principle. The only difference is that the multiple platter system scatters the data of a certain file to each platters' same tracks (called together as a cylinder), same sectors and while processing the file, it makes use of parallel reads/writes by means of the heads working as unite, as can be seen in Figure 3.2. Hence a certain file that is written on a single platter only

differs from that of written on a multiple platter one, by the area it covers on the surface (i.e. block size on single platter disk = A; block size on 6 platter disk = A / 6). Therefore, concentrating on a single surface of a disk platter, depicted in Figure 3.3, will not make any difference other than easier understanding of the problem.

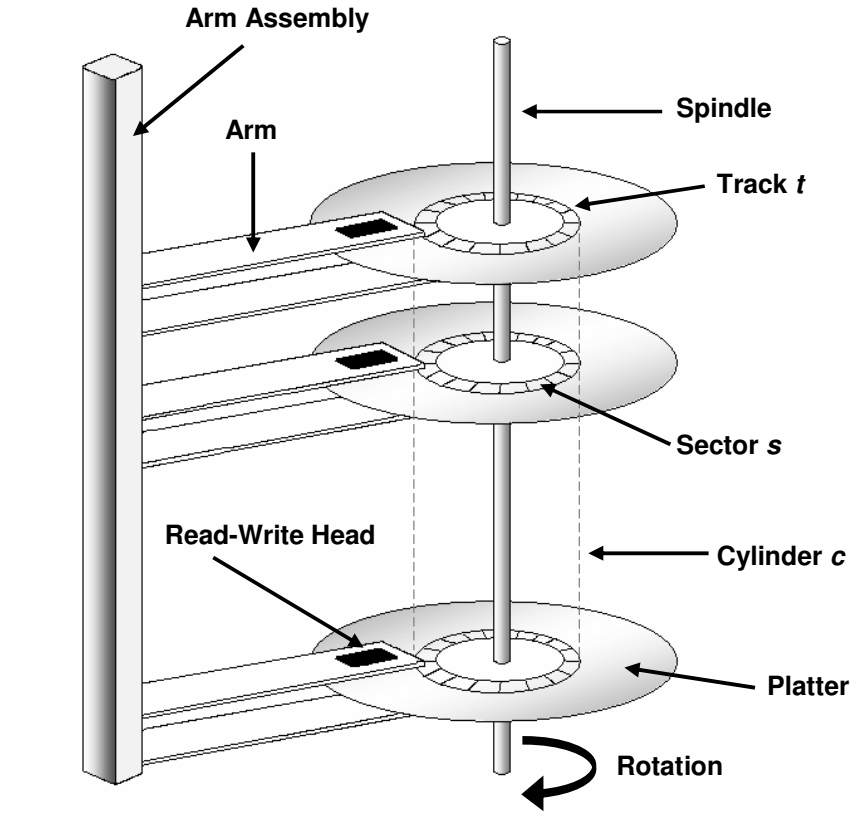

Figure 3.2 – Disk Head Mechanism

For most of the systems, in disk scheduling, non-preemptive jobs coming randomly are directed to a queue, which has a capacity of **n** (typically 128 for enterprise machines), 216 jobs as maximum. In addition to that, there is another queue, namely Run Queue (RQ),

before the disk processor. The capacity of this queue is two. Once the jobs to be placed into this queue are determined, they cannot be removed or their sequence cannot be changed. If the number of the jobs in the system is greater than two, then the RQ is full. Including RQ, total capacity of the system is n+2+1 (131 for enterprise machines typically) jobs on a distinct moment of time.

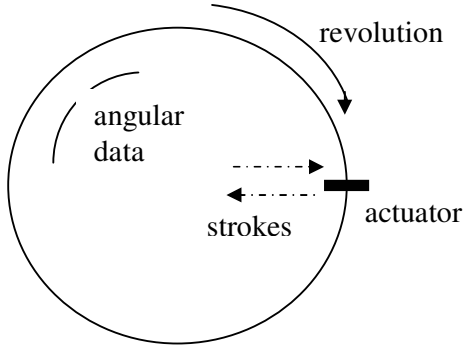

Figure 3.3 – Disk Working Principle

For a job that comes into the process, an action composed of two distinct movements is done for the actuator (disk head) to start the job processing. First movement involves inward and outward linear strokes of the actuator along the disk radius, and the second is the rotational one (disk rotation). The time passed for disk head finding the track including the job is called as **seek**. After the head finds the track and stops over that, it waits until the job address starting location comes under the head. This waiting time is called as **latency**. These two are counted as service time components, together with the **transfer time**. Their average values (read/write average) are 6 ms and 3 ms for standard 10,000 rpm (revolution per minute) disks. The writing processes necessitate greater seek times, since it requires the higher precision over disk surface. The

transfer time is related to the size of the job. Whether the disk head reads/writes data on a cylinder or stands over it without doing anything, the same time passes, which is the disk rotation time. Since electronical transfer time is an insignificant value comparing with the mechanical rotation time, transfer time is accepted as equal to the disk rotation time.

Hard disk scheduling works in such a way that at every job replacement in and addition to the queue, the scheduling algorithm defines the new job sequence in the queue. Each time the service of a job is finished, the first job in run queue (RQ) is taken into process, the second job in RQ becomes the first one, and the job scheduled as first in the queue takes the second position in RQ.

Job sequencing decisions arisen from some greedy sequencing algorithms could cause an undesired situation (called as job starvation) so that some jobs could have to wait for service unacceptably long time. To measure that, the system put a label on every job i indicating its arrival time into the system (r_i), and after a defined time stamp (τ), if it is still waiting to be serviced, timeout case happens and the job is assessed tardy ($>r_i + \tau$).

To our knowledge, the standard hard disks follow such a way that, existence of a timeout job turns the whole system processing sequence policy into FIFO, until all the jobs in the system at that instant are finished.

Although, it would be in nanoseconds, there is always a difference between the job arrival times. Hence, there could be only a single job that is timeout on a certain moment of time, and once the

system undergoes to FIFO policy, other jobs inclined to be timeout will be serviced in accordance with their strength of inclination following the timeout one. However, in reality there are plenty of cases in which more than one job can be timeout simultaneously. Especially in busy systems, arrival of bursty job bulks having even 100 jobs could be possible. In such a case, if timeout of these bursty jobs happens, defining a process reentry policy for them could improve the performance as well.

There are several objectives or disk scheduling performance criteria in the general disk scheduling problem (online):

1) Maximizing the job service rate (jobs/second):
   Maximizing the job service rate means minimizing the completion time of the last job in the system, referred as makespan in machine scheduling literature. This objective does not take the average flow time of the jobs into consideration. Thus, this method disregards the service quality concern. It does not take a measure for preventing timeouts, so does not guarantee service stability.

2) Maximizing the data service rate (bytes/second):
   A modified version of the first objective maximizes the data throughput instead of taking each job as a unit whatever the size of bytes it holds. It is useful for job sets having a greater size variation.

3) Minimizing the maximum flow time of jobs (maximum waiting time of jobs in miliseconds):

In machine scheduling, minimizing the maximum flow time in system is a worst case measure, having poor overall performance for makespan and average flow time measures. But, it guarantees service stability by preventing timeouts.

4) Minimizing the average flow time of jobs (average waiting time of jobs in miliseconds):
   The average flow time is one of the good measures of service quality as well as the overall performance. However, like 1st and 2nd objectives, it also does not guarantee service stability.

The disk scheduling problem is an online problem in reality having the features explained before in Chapter 2. The jobs that come randomly at any instant without beforehand knowledge must be served and/or queued almost instantly by the disk mechanism. Hence, in our understanding, when a new job comes, the decision maker (DM) has time to decide on the schedule of the new job for processing until the completion of the job in process. Exploiting this problem structure, the online problem can be thought as many instances of offline problems. When the processing of a job has started, the new scheduling decision for the jobs in Queue (Q), for instance, can be taken by solving an offline TSP problem within the time stamp available until the completion of the job in processing. For that reason, to solve the online problem, we suggest that a sequence of offline problem instances is solved within varying time intervals whose lengths are determined by the online problem occurrances like completions. Below we define offline disk scheduling problem for which it would take place as a decision support tool in solving the online problem.

3.3 Mathematical Modeling of Offline Disk Scheduling Problem

The basic models constructed in this section refer to the formulations of the **offline problems**. In the offline problem, we assume that all the requests are known in advance by the system, with their arrival times, locations, processing (transfer) times and due dates at the very beginning. Here, we use the same notation by Rabadi (2001). Recall that in the real problem (the online version of problem), arrival times are stochastic and are not known before jobs arrive the system. After the jobs arrive, locations, processing times and due dates are known. Note that, the arrival time of a certain job constitutes its time window's lower bound while a constant timeout value decided for the disk system is added to the job's arrival time to set the upper bound of the time window. The problem is a kind of asymmetric traveling salesman problem (TSP) with time windows. The asymmetric nature of the TSP arises from the disk's working principle. Since the hard disk platters rotate always in the same direction, the two-way time distances between two distinct jobs' physical block addresses are not equal.

Below, we first explain our parameters and then decision variables for the offline problem. Then, we present four offline disk scheduling problem formulations with different objective functions.

Parameters:

$S_{i,j}$: Sequence dependent setup time from job i to job j (seek + rotational latency).

t_i : Transfer (processing or read/write) time for job i.

r_i : Arrival time of job i into the system.

d_i : Due date of job i (r_i + allowable constant waiting time (τ) value before being timeout).

M : A large positive number.

Decision Variables:

$X_{i,j}$: Binary variable defining whether job i directly precedes job j or not in the sequence (It takes 1 if job i directly precedes job j, 0 otherwise).

C_i : Completion time of job i.

E_i : Earliness of job i (It indicates as if a job i is completed before its due date), i.e. $E_i = \max\{ 0, d_i - C_i \}$.

T_i : Tardiness of job i (It indicates as if a job i is completed after its due date), i.e. $T_i = \max\{ 0, C_i - d_i \}$.

C_{max} : Completion time of the last job in the system, which is also called makespan, i.e. $C_{max} = \max_i \{ C_i \}$.

F_i : Flow time of job i, i.e. $F_i = (C_i - r_i)$.

NT_i : Binary variable defining whether job i is tardy (being timeout) or not (It takes 1 if job i is completed after its due date, 0 otherwise), i.e. $NT_i = \begin{cases} 1 \text{ if } T_i > 0; \\ 0 \text{ o/w.} \end{cases}$

NT : Total number of tardy (timeout) jobs, i.e. $NT = \sum_i NT_i$.

The four offline disk scheduling problems (DSP):

A – DSP with Makespan Criterion

The makespan is the time by which the service of the last job in the system is finished. In the formulation below, the objective is to minimize the makespan. It indicates a good overall system performance measure, although it gives little guarantee on the service quality (i.e. the average job flow time). Besides, it does not take any direct measure for minimizing the number of tardy jobs.

Minimize C_{max} (3.1)

Subject to

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad j=1,\ldots,n \quad i \neq j \qquad (3.2)$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \qquad i=1,\ldots,n \quad i \neq j \qquad (3.3)$$

$$C_i \geq r_i + S_{0,i} + t_i - M(1 - x_{0,i}) \qquad i=1,\ldots,n \qquad (3.4)$$

$$C_j \geq r_j + S_{i,j} + t_j - M(1 - x_{i,j}) \qquad i=1,\ldots,n \quad j=1,\ldots,n \quad i \neq j \qquad (3.5)$$

$$C_j - C_i + M(1 - x_{i,j}) \geq S_{i,j} + t_j \qquad i=1,\ldots,n \quad j=1,\ldots,n \quad i \neq j \qquad (3.6)$$

$$C_{max} \geq C_i \qquad i=1,\ldots,n \qquad (3.7)$$

$$x_{i,j} \in \{0,1\} \qquad i=1,\ldots,n \quad j=1,\ldots,n \quad i \neq j \qquad (3.8)$$

$$C_i \geq 0 \qquad i=1,\ldots,n \qquad (3.9)$$

As mentioned before, the model resembles an asymmetric TSP with time windows of $[r_i, \infty]$. The objective function (3.1) minimizes the makespan value that is controlled by constraint set (3.7) which ascertains that the makespan is greater than or equal to the completion time of all the jobs. Constraint set (3.2) makes sure

that there is only one job preceding job j, while (3.3) ascertains that there is only one job following job i. Constraint set (3.4) provides that the completion time for the first job to be processed is greater than or equal to the summation of its arrival time, set up time from the disk head's initial position, indicated by job j=0, and its transfer time. It is made certain by constraint set (3.5) that the completion time of any job j, which is directly preceded by job i, is greater than or equal to the summation of its arrival time, setup time from job i to j, and transfer time of job j. If job i precedes job j, then the completion time difference between job j and job i must be greater than or equal to the summation of setup time from job i to job j and transfer time of job j, and it is provided by constraint set (3.6). Big M in constraint set (3.6) together with constraint sets of (3.4) and (3.5) and constraint set (3.8) eliminates the possibility of having overlapping disjoint paths (sub-tours). $x_{i,j}$ is a binary variable having value of 1 if job i directly precedes job j, otherwise 0 as constraint set (3.8) indicates. Finally, constraint set (3.9) provides the non-negativity for the completion time value for all jobs.

B - DSP with Makespan Criterion subject to Due Date

In the following makespan minimization model, timeout is not allowed. The timeout value is a fixed predetermined constant maximum waiting time in the system showed by the symbol "τ". When it is feasible, the timeouts are avoided and so is the performance decrease caused by them. The model could do that at the expense of longer disk head movements, and could have to make disk head swing too many times over the disk surface than it

would have to in the case in which the timeout is allowed. However, depending on the problem instance data, a feasible solution is not guaranteed in the model.

As can be seen below, the objective function and all the constraints of *DSP with Makespan Criterion* model are preserved in this model. Only difference is the addition of constraint set (3.10) which is the hard constraint, imposing the completion of job i before its due date.

Minimize C_{max} (3.1)

Subject to

$$\sum_{i=1}^{n} x_{i,j} = 1 \qquad j=1,\ldots,n \quad i \neq j \qquad (3.2)$$

$$\sum_{j=1}^{n} x_{i,j} = 1 \qquad i=1,\ldots,n \quad i \neq j \qquad (3.3)$$

$$C_i \geq r_i + S_{0,i} + t_i - M(1 - x_{0,i}) \qquad i=1,\ldots,n \qquad (3.4)$$

$$C_j \geq r_j + S_{i,j} + t_j - M(1 - x_{i,j}) \qquad i=1,\ldots,n \quad j=1,\ldots,n \quad i \neq j \qquad (3.5)$$

$$C_j - C_i + M(1 - x_{i,j}) \geq S_{i,j} + t_j \qquad i=1,\ldots,n \quad j=1,\ldots,n \quad i \neq j \qquad (3.6)$$

$$C_i \leq d_i \qquad (d_i = r_i + \tau) \qquad i=1,\ldots,n \qquad (3.10)$$

$$C_{max} \geq C_i \qquad i=1,\ldots,n \qquad (3.7)$$

$$x_{i,j} \in \{0,1\} \qquad i=1,\ldots,n \quad j=1,\ldots,n \quad i \neq j \qquad (3.8)$$

$$C_i \geq 0 \qquad i=1,\ldots,n \qquad (3.9)$$

C – DSP with Tardiness Related Criterion

This model does not take a measure for minimizing the makespan, but has concern for tardiness. Minimizing number of tardy jobs

($\sum NT_i$) does not have a direct effect on the overall performance of the system in terms of makespan criterion and it may even worsely affect the overall performance (i.e. makespan) depending on characteristics of the job data. On the other hand, minimizing the total tardiness ($\sum T_i$) carries the same disadvantages with ($\sum NT_i$) and it is meaningless in the sense that in real system, whether a tardiness amount is large or very small, whenever a timeout happens the system always lose about the same amount of performance. Hence, the number of tardy (being timeout) jobs is a better measure that should be considered if tardiness is important for the decision maker.

Minimize $\sum_{i=1}^{n} NT_i$ (3.11)

Subject to

$\sum_{i=1}^{n} x_{i,j} = 1$ j=1,...,n $i \neq j$ (3.2)

$\sum_{j=1}^{n} x_{i,j} = 1$ i=1,...,n $i \neq j$ (3.3)

$C_i \geq r_i + S_{0,i} + t_i - M(1- x_{0,i})$ i=1,...,n (3.4)

$C_j \geq r_j + S_{i,j} + t_j - M(1- x_{i,j})$ i=1,...,n j=1,...,n $i \neq j$ (3.5)

$C_j - C_i + M(1- x_{i,j}) \geq S_{i,j} + t_j$ i=1,...,n j=1,...,n $i \neq j$ (3.6)

$C_i - T_i + E_i = d_i$ i=1,...,n (3.12)

$T_i \leq M(NT_i)$ i=1,...,n (3.13)

$x_{i,j} \in \{0,1\}$ i=1,...,n j=1,...,n $i \neq j$ (3.8)

$NT_i \in \{0,1\}$ i=1,...,n (3.14)

$C_i , T_i , E_i \geq 0$ i=1,...,n (3.15)

The objective function (3.11) minimizes the total number of tardy jobs within the job set consisting of n jobs. Constraint sets (3-2) to (3-6) and (3.8) apply here also. Constraint set (3.12) is the soft

constraint, allowing the completion of job i after its due date, where $d_i = r_i + \tau$. If job i is tardy, T_i must be less than or equal to a very big number, otherwise T_i equals to 0, and that is provided by constraint set (3.13). Constraint set (3.13) also counts the number of cases in which a timeout occurs. Constraint set (3.8) also applies here. NT_i is a binary variable having value of 1 if job i is tardy, otherwise 0 as constraint set (3.14) indicates. Constraint set (3.15) gives non-negativity property of the tardiness and earliness of each job, in addition to that of completion time.

D – DSP with Total Flow Time Criterion

The flow time is the time spent by a job after its arrival into the system until its departure from the system (after the job is served). Although this model is likely to give a larger makespan value than that of the makespan minimization model, it is still a useful objective. The total flow time gives an information about the quality of service, because it keeps tracks of the average waiting time in the system. Minimizing average waiting time in the system does not guarantee maximizing the job service rate in number of jobs per second. On the contrary, an increasing service quality with more balanced service is expected to increase the number of jobs in queue waiting for service in a distinct moment of time. Hence, if timeout does not likely to happen, the flow time minimization is not so preferable to apply.

Minimize $\sum_{i=1}^{n} F_i$ (3.16)

Subject to

$\sum_{i=1}^{n} x_{i,j} = 1$ j=1,...,n i ≠ j (3.2)

$\sum_{j=1}^{n} x_{i,j} = 1$ i=1,...,n i ≠ j (3.3)

$C_i \geq r_i + S_{0,i} + t_i - M(1- x_{0,i})$ i=1,...,n (3.4)

$C_j \geq r_j + S_{i,j} + t_j - M(1- x_{i,j})$ i=1,...,n j=1,...,n i≠j (3.5)

$C_j - C_i + M(1- x_{i,j}) \geq S_{i,j} + t_j$ i=1,...,n j=1,...,n i≠j (3.6)

$F_i = C_i - r_i$ i=1,...,n (3.17)

$x_{i,j} \in \{0,1\}$ i=1,...,n j=1,...,n i≠j (3.8)

$C_i , F_i \geq 0$ i=1,...,n (3.18)

In the formulation above, the objective function (3.16) minimizes the total flow time. For job i, the flow time value equals the completion time of that job minus its arrival time, as constraint set (3.17) provides. Constraint sets (3.2) to (3.6) and (3.8) apply here also. Constraint set (3.18) gives non-negativity property of the flow time of each job, in addition to that of completion time.

3.4 Computational Difficulty of the Offline Problem

From the basic NP-hard problems modeled in the previous section, DSP with Makespan Criterion (Model A) and its due date version (Model B) reduce to TSP with time windows problem in routing and scheduling literature, whereas the other two models are different variants of TSP. The first problem has time windows of $[r_i, \infty]$, while the second has $[r_i, r_i+\tau]$, where r_i is arrival time of the job i and τ is the allowable constant waiting time value for all the jobs. It is known that, as the problem size increases, the number of iterations and the time required to reach the optimal solution of the problem increase exponentialy for TSP with time windows. This remark is also valid for Models C and D. Therefore, one can expect that solving these models with even moderate size data will take longer time.

For illustrating the computational difficulty of solving the offline problem to optimality, we prepared two sets of problem data using the disk address distances. By using Lingo 8.0, we solve Model A with setting of $t_i = r_i = 0$ for all the jobs. From the uniformly distributed address values, we generated two data sets: an 8-job problem and a 10-job problem. We formed 10 different instances from each set. As given in Table 3.1, 8-job instances are solved to optimality in 290 seconds on average. However, a small increment in the number of jobs resulted in large inefficiency. Finding the optimality took more than 7 hours on average for two instances of 10-job data sets. When we run Lingo with a 1-hour time limit to solve these instances, the resulting solutions deviated from the optimal only 1% on average. It was indicating that Lingo was spending more time to justify the optimality of the solution found

in the early stages of our run than to find a reasonable solution at the first place. The number of iterations for optimality were about 90 times more on average for 10-job instances than those of 8-job instances. This also justifies the reason for using heuristics to solve the TSP models in shorter times.

Table 3.1 –TSP Solution Times in Lingo 8.0

	8-job set (runs to the optimality)		10-job set (~1 hour runs for feasible soln)	
	CPU time for optimal soln (minutes)	# of Iterations	CPU time for feasible soln (minutes)	# of Iterations
	4.10	1,595,164	62.88	18,742,638
	4.10	1,757,812	60.05	18,121,629
	6.00	2,443,566	60.02	19,871,532
	5.27	2,048,049	60.72	19,829,817
	6.10	2,584,775	61.02	18,337,626
	4.92	2,024,652	65.32	22,139,528
	5.38	2,224,410	60.07	18,321,801
	4.20	1,552,391	68.08	20,542,373
	3.63	1,452,221	60.05	20,058,472
	4.75	1,952,978	61.67	18,987,827
Average	4.83	1,963,602	61.98	19,495,324

CHAPTER 4

ONLINE DISK SCHEDULING PROBLEM

In this chapter, disk scheduling in online setting will be discussed. In the first section, the general conventional online system setting is defined. Next, our approach to tackle the online system is introduced. In the last section, the rescheduling procedures for different policies are presented.

4.1 Conventional Approach to Online Problem

In conventional approach, the online disk scheduling system works as follows. If an arrival occurs, it will change the job content of the queue and a new scheduling decision is to be made, or if a timeout happens, it turns the real system into FIFO. So, the main events in online disk scheduling are the arrival of a new job and the timeout of a job. In this type of situation, one assumes that the decision maker has enough time for making a scheduling decision until the next job arrival or occurrence of timeout. Although timeouts are rare that could be traced in system with some early alert modification, the actual system does not proactively trace timeouts and take precaution before the incident. Since timeout and arrival times of the coming jobs are not known in advance in online systems, the decision maker also does not know the available time s/he has while making a sequencing decision until a new event

happens (indefinite decision epoch). To illustrate how the system works we introduce our notation below.

RQ : The run queue, a queue with size of two, in which no resequencing is done. Once a job enters it, it cannot leave until being served.

RQ1, RQ2 : The jobs in first and second places of run queue.

Q : The set of jobs in queue following the run queue. The jobs within this queue are undergone sequencing, if needed.

Reschedule : The chosen queue sequencing policy.

Qseq(k) : The job in the k^{th} order in queue (Q).

Rescheduling is needed if the number of the jobs in Q is greater than 1 before a scheduling decision is made, otherwise no decision is needed. Possible events necessitating rescheduling decisions are as follows:

- Arrival of a new job (newcomer) **before** the completion of the job in process,
  If RQ is full and $|Q| \geq 1$
  Append newcomer job to Q ($|Q| = |Q| + 1$)
  Call Reschedule(Q)

- Arrival of a new job (newcomer) **after** the completion of the job in process,
  If RQ is full and $|Q| \geq 2$
  RQ1= RQ2
  RQ2=Qseq(1)
  Extract Qseq(1) from Q

Append newcomer job to Q (|Q|=|Q|)
Call Reschedule(Q)

- Timeout of a job in Q,
  Call FIFO (Q)
  When a timeout happens all the system's sequencing policy changes to FIFO in the conventional disk scheduling until all the jobs within the system came up to that time are served.

The reschedule procedure can be presented as follows:

Procedure Reschedule(Q)
All jobs waiting in queue reordered according to a particular scheduling policy:
For k=1 to |Q|
Update Qseq(k) according to scheduling policy
End

When a new job arrives into the system or timeout of a job happens, the decision maker has a time till the next job arrival or occurrence of timeout for making a scheduling decision. The real world hard disk systems work with this principle, which we call conventional disk scheduling.

4.2 Alternative Scheduling Approach To Online Problem

In this section, we introduce a new concept, called **Deterministic Decision Epoch (DE)**, to propose a different approach for the decision maker for solving the online DSP. The time interval that can be used for making a new ordering decision for Q is called a decision epoch. It starts with a "simultaneous incident chain"

when a service of a job has just finished and then the first job in RQ undergoes into service. Instantaneously, the second job takes the first one's position in RQ and the predefined next job in Q (say first job in Q) takes the second one's position in RQ. That time interval ends when the job under service is done and then the first job in RQ undergoes into service, and so on.

In this new approach, queuing decisions are not made at every arrival and/or timeout cases. Instead, a decision epoch is defined as the time interval bounded by the completion time of the job in service up to the completion time of the next job. Although the length of decision epoch is a variable time, the decision maker knows the exact length of it. The reason is that, the completion time of the next job which was made certain as the first RQ job is already known. The decision maker is "blind" to the events occurring within the epoch (i.e. arrival of a new job or timeout of a waiting job in queue), and makes decision with the information gained one epoch before. Hence, this approach is realistic for making scheduling decisions in meaningful time in real world cases. The main power of this approach is that, in spite of the stochastic nature of the problem, decision maker converts the stochastic nature into the deterministic one within the decision epoch that is used for making decisions on known domain. Below we demonstrate how new approach proceeds.

Assume that the identity indices of the jobs in service order are as follows: $\rightarrow p \rightarrow o \rightarrow l \rightarrow k \rightarrow j \rightarrow i$

Here, job i is the last completed job, the service of job j is in progress, jobs k and l are the jobs in RQ, and jobs p and o are in Q. Figure 4.1 illustrates the order of jobs within DE framework.

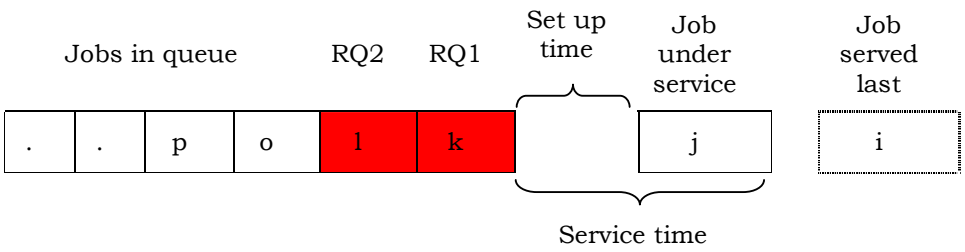

Figure 4.1 Illustration of DE_j

At the end of **deterministic** decision epoch j (DE_j), that is when job j has been just finished, the sets formed by the events happened during this decision period are classified as follows:

A_j : Set of jobs that arrived during the decision epoch j,

D_j : Set of jobs that were timeout during the decision epoch j,

The effected sets after those events are:

Q_j : Set of jobs in queue at the end of decision epoch j,

QD_j : Set of timeout jobs in queue at the end of decision epoch j,

where

$Q_j \supseteq QD_j \supset D_j$

$Q_j \supseteq A_j$.

The length of DE_j (LDE_j) can be defined as follows:

$LDE_j = C_j - C_i = S_{i,j} + t_j$

The events that happen during DE_j (while job j is under service) are not taken into consideration while making a sequencing decision within DE_j until the starting time of service of job k (i.e. at the start of DE_k). Because the events and their features are not known in advance until the end of DE_j and they become available over time during DE_j. Instead, the realized events during DE_i are considered

within LDE_j. LDE_j refers to the time available for sequencing decisions until the start of DE_k.

Let the index j in Figure 4.2 denote j^{th} job in the system, not job j. Assume that our notation is also changed according to this index definition. The complete events and activities in successive decision epochs are given in Figure 4.2 in this manner.

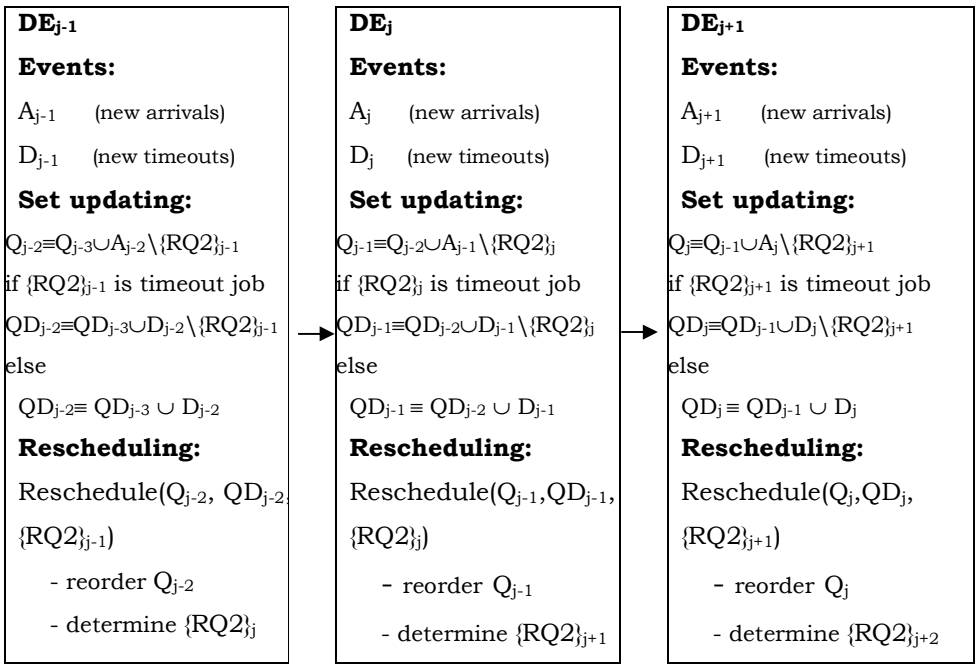

Figure 4.2 – DE Events and Activities

Let n be the number of all jobs in system plus the jobs served in service history at the start of DE_j.

Events during DE_j :

The new arrivals coming during this period of time are indexed according to arrival times, starting from n+1. Hence arrival times are known by the system over time during DE_j.

A_j : new arrivals during DE_j, i.e. n+1, n+2, …, m.

$|A_j| = m - n$

If any timeout happens during the epoch, they are separately beset in D_j set, where the set QD_j contains all the timeout jobs in system.

D_j : timeouts during DE_j.

Updating at the start of DE_j :

Q_{j-1}, QD_{j-1} : At the start of DE_j, the contents of the Q_{j-1} and QD_{j-1} are formed by simply appending the set of jobs arrived (A_{j-1}) and being timeout (D_{j-1}) during DE_{j-1} to the Q_{j-2} and QD_{j-2}, then extracting the $\{RQ2\}_j$ which is determined during DE_{j-1} by Reschedule(Q_{j-2}, QD_{j-2}, $\{RQ2\}_{j-1}$) procedure.

Rescheduling during DE_j :

Reschedule(Q_{j-1},QD_{j-1}, $\{RQ2\}_j$) : After updating is completed at the beginning of DE_j, we check the cardinality of the newly formed sets.

- If cardinality of QD_{j-1} is greater than or equals to one, then the current rescheduling policy turns to FIFO for the jobs in that set. Recall that QD_{j-1} was formed by simply appending D_{j-1} into the end of the ordered set QD_{j-2}, and the ordered set QD_{j-1} already conforms to FIFO. The jobs in Q_{j-1} is reordered again with current rescheduling policy if the cardinality of Q_{j-1} is greater than one. In processing sequence, after the last job in QD_{j-1}, the jobs in the rescheduled set Q_{j-1} is placed, according to their new sequence.

- If QD_{j-1} is empty and the cardinality of Q_{j-1} is greater than one, then the set Q_{j-1} is reordered in rescheduling according to the chosen rescheduling policy. This procedure also uses the information of the job in run queue 2 ($\{RQ2\}_j$), since the

job following it would have setup time depending the location of $\{RQ2\}_j$. After rescheduling, the first job of the reordered Q_{j-1} set is chosen as $\{RQ2\}_{j+1}$, the job that will take the 2^{nd} position in RQ during DE_{j+1}.

4.3 Reschedule Procedure in DE Setting

Whether the conventional or our approach is chosen, a rescheduling procedure is needed for determining the job sequence. Since the rescheduling procedures in DE approach reduces to those in conventional approach with number of arrivals being limited to 1, all the procedures are given for DE setting.

In many real hard disk systems, an occurrence of timeout turns the scheduling policy into FIFO until all the jobs already in the system are serviced. That kind of policy means high degradation on system performance. The theoretical system below assumes that whenever a job becomes tardy it is allowable to give it a new due date, so that the scheduling algorithm allows timeouts. Timeout cases are used as indicators of the performance of an algorithm in this study.

Although the same rescheduling policies can be applicable in both approaches, they are not the same. The conventional scheduling approach needs rescheduling at every arrival, while ours make rescheduling if a set of arrivals happen within the decision epoch. Hence, we allow happening of multiple events before rescheduling, if possible.

Below, we show how four rescheduling policies, namely, FIFO, CLOOK, and two C_{max} (Makespan Minimization) policies, are applied within DE framework. During the length of DE_j, the policies have information of every variables that are known by the system, where:

Q_{j-2} : ordered set of jobs in queue at the start of DE_{j-1} except $\{RQ2\}_j$

A_{j-1} : ordered set of jobs came during last DE_{j-1}

FIFO:

Append jobs in A_{j-1} into the end of the job set Q_{j-2} without changing their order (or simply use updated set Q_{j-1} at the beginning)

Extract $\{RQ2\}_j$ from the newly formed set

Rename the new set as Q_{j-1}

CLOOK:

Here, a convention is made in the use of sets. Instead of using Q_{j-1}, the ordered set Q_{j-2} that was rescheduled during DE_{j-1} is taken and the insertion of A_{j-1} (set of arrivals during the same epoch) into Q_{j-2} is considered. In this setting, the first job in Q_{j-2} is actually RQ_j, so it is never involved in comparisons below. Hence, a newly arrived job's best possible position can be the second place according to this convention. The addresses of jobs in rescheduled set Q_{j-2} actually make a circle of increasing numbers. Hence, finding the smallest address in this circle and proceed from there make the algorithm easier:

For i=1 to $|A_{j-1}|$

Find the insertion place of i^{th} member of A_{j-1} into the ordered set Q_{j-2}, comparing the addresses. Give a sequencing number to the inserted job, with reference to the constant first job in set Q_{j-2} and shift the order of other jobs accordingly

$Q_{j-2} \equiv$ new set (the cardinality of that increased by one)

$Q_{j-1} \equiv Q_{j-2}$

Exact Solution of Cmax (Makespan) Minimization:

We call the exact solution of C_{max} minimization as the exact solution of sequence dependent makespan minimization (SDM-E) afterwards. Solving SDM-E within DE approach seems advantageous, since we have a chance to involve more than one jobs at single run of TSP, instead of running the algorithm at every job arrival into the queue. But, the solution time cannot be expected to be shorter than that applied in conventional approach, since the computation time increases exponentially with increasing Q length. The makespan criterion disk scheduling problem introduced in Section 3.3–A is modified here for the online Reschedule procedure. The arrival times are eliminated in the formulation below, since the arrival times are taken into consideration in the way that only the jobs arrived into system are undergone reschedule. The first job's starting time constraint is also eliminated, since scheduling does not start from the first job. There is always a job preceding the jobs undergone to scheduling. Below Q represents the set Q_{j-1}, which is simply formed by appending set A_{j-1} at the end of set Q_{j-2}.

Minimize C_{max} (4.1)

Subject to

$$\sum_{i=1}^{|Q|} x_{i,j} = 1 \qquad j \in Q \qquad i \neq j \qquad (4.2)$$

$$\sum_{j=1}^{|Q|} x_{i,j} = 1 \qquad i \in Q \qquad i \neq j \qquad (4.3)$$

$$C_j \geq S_{i,j} + t_j - M(1 - x_{i,j}) \qquad i, j \in Q \qquad i \neq j \qquad (4.4)$$

$$C_j - C_i + M(1 - x_{i,j}) \geq S_{i,j} + t_j \qquad i, j \in Q \qquad i \neq j \qquad (4.5)$$

$$C_{max} \geq C_i \qquad i \in Q \qquad (4.6)$$

$$x_{i,j} \in \{0,1\} \qquad i, j \in Q \qquad i \neq j \qquad (4.7)$$

$$C_i \geq 0 \qquad i \in Q \qquad (4.8)$$

The objective function (4.1) minimizes the makespan value controlled by constraint set (4.6). (4.6) ascertains that the makespan value is greater than or equal to the completion time of all the jobs in queue. Constraint (4.2) makes sure that there is only one job preceding job j, while (4.3) ascertains that there is only one job following job i. Since the disk system is assumed to be empty at the start and the job arrivals are discrete, the first job to be processed is the job that arrives the system first, hence constraint set (3.4) of the offline formulation is eliminated here. It is made certain by (4.4) that the completion time of job j, which is directly preceded by job i, is greater than or equal to the summation of its setup time from job i to j, and transfer time of job j. If job i precedes job j, then the completion time difference between job j and job i must be greater than or equal to the summation of setup time from job i to job j and transfer time of job j, and it is provided by (4.5). $x_{i,j}$ is a binary variable having value of 1 if job i directly precedes job j, otherwise 0 as (4.7) indicates. Constraint (4.8)

provides the non-negativity for the completion time value for all the jobs.

Nearest Neighbor Heuristic Solution of Sequence Dependent Makespan Minimization (SDM-NN) :

We call the approximate solution of C_{max} minimization as nearest neighbor heuristic solution of sequence dependent makespan minimization (SDM-NN). Nearest neighbor, the well known TSP heuristic, works as its name implies. The algorithm tries to find the nearest node to the present node at each iteration. In our system, the algorithm appends the jobs in new arrival set directly at the end of the jobs in Q. And then, taking the starting job address as that of the job in RQ2, algorithm proceeds finding the next job with smallest setup time (seek+latency).

> Form Q_{j-1} by appending $|A_{j-1}|$ to Q_{j-2}
>
> Starting from the job in $\{RQ2\}_j$, proceed by finding the job having an address giving the smallest setup time (seek+latency) from the last address.

CHAPTER 5

SIMULATION IMPLEMENTATION AND ANALYSIS

In following sections, the features of simulations, the way the algorithms are coded, the parameter setting and the test bed generation are presented. Also, the experimentation of algorithms together with the brief analysis of results take place within this chapter.

5.1 The Nature of Simulation

While simulating the basic system we have two experimentation settings. In Experiment 1, we test conventional methodology based on job arrivals. Here, every job arrival necessitates a scheduling decision. In Experiment 2, we do an experimentation based on DE concept, and here, the DM has deterministic time intervals to make scheduling decision.

In Experiment 1, three sets of Monte Carlo Simulations are done for FIFO policy, CLOOK and the exact solution of sequence dependent makespan minimization (SDM-E) algorithms. SDM-E's solution time makes it impossible to use in real system, but it is simulated for the sake of unbiased benchmarking. All three are realized in the same way independent of the policy chosen. In Experiment 2, two sets of simulations are done for CLOOK and the nearest neighbor heuristic solution of sequence dependent

makespan minimization (SDM-NN) algorithms. FIFO is not applied within DE setting since, it will give the same result with that of conventional approach. SDM-E is also not applied, because of its high computational requirements arising from the longer queue lengths under this decision setting.

The exact solution of sequence dependent flow time minimization (SDF-E) was planned to be applied under DE setting, but preliminary runs proved the claim stated in Chapter 3 that, an increasing service quality is expected to increase the number of jobs in queue waiting for service in a distinct moment of time. Because of the computational difficulty of exact solution which was mentioned in the same chapter, that increase in number of jobs in Queue made SDF-E impossible to be applied in reasonable times.

The hard disk can be seen as a flow line system with single processor and two pre-processing waiting place. The entity of the system is the jobs, and the attributes of the jobs are the addresses identifying their position on hard disk surface.

In both experiments, the state variables of all five (3 from Experiment 1 and 2 from Experiment 2) simulations are the same, namely, processor's situation, Run Queue 1 and Run Queue 2 situations (empty or full), number of jobs in Queue (if any), the arrival time of the next job, and the completion time of the last job in processor. The jobs that become tardy (timeout cases) are not included into state variables. It is because of our methodology, which allows tardiness. We allow timeout cases and use the timeout statistic as a performance measure in output analsis.

The main two events that cause the change in state variables are the new arrival and the completion of the job by the processor. Since these two happens on discrete (countable) points in time, the simulation is a discrete event simulation in nature. As mentioned above, timeouts are not counted as simulation events, since they do not affect the decision in our methodology.

Although the real system is theoretically steady state, it is reduced to a "kind" of terminating simulation for practical purposes. Because actually it does not stop with the arrival of n^{th} job, but stops accepting jobs after that arrival and terminates with the completion of the last job in the Queue. So that, the length of the simulation is not a specified time, instead a condition. Hence, the length of a simulation is also a variable.

5.2 Coding

All five applications are coded in Visual Basic 6.0 programming environment. For SDM-E (to solve the developed Model A), the Lingo 8.0 optimization software package is used. For the computation of seek plus latency times of disk head from one address (PBA) to another on the disk surface, a special program "Disksim" coded on C is used. Disksim is an adaptation of the disk simulator developed by the EMC Corporation (2006, Hopkinton, Massachusetts).

The simulation code framework of the applications in Experiment 1 and in Experiment 2 differ because of the different decision structures used. Both code frameworks can be seen on Appendix A. The transfer times (t) are taken as constant. $S_{i,j}$ values (seek +

latency) are computed using the Disksim program. Since, timeout cases are allowed and can only be traced from simulation outputs, the timeout is not explicitly seen on the code.

Each application differs from the others mainly in the rescheduling procedure.

For the FIFO policy, the rescheduling procedure seen on the codes placed in Appendix A.1 and A.2, due to its nature, does not change the composition of the queue over time.

The code framework for the CLOOK algorithm used in both experiments is provided in Appendix B. The rescheduling procedure of CLOOK algorithm takes scheduling decisions over the physical block addresses instead of seek + latency times as mentioned before.

The SDM-E algorithm based application sends the queue information to an interface in Lingo 8.0. Its code designed for taking input into the TSP model is presented in Appendix D. However, the time window concept is eliminated here, since the timeouts are allowed and the already arrived jobs into the Queue are taken for rescheduling. Also the transfer times are not taken into the model since they are taken as constant for all the jobs.

The SDM-NN algorithm is Nearest Neighbor heuristic solution approach to the Model A in Chapter 3, as explained before in Chapter 4. The code framework for the SDM-NN algorithm used under DE setting is provided in Appendix C.

5.3 Parameter Setting, Test Bed Generation and Experimentation

Hard disks show different performances on read and write operations. Disksim uses read operation average times for all the computations of hard disk processing time values.

For transfer time parameter, different from seek and latency total, a single constant value is chosen, instead of computing that for each single job. This is because the transfer time is insignificant in overall time spent for processing a job. Since the jobs usually come in size of plus/minus 8 KB and a standard hard disk transfers the data with 50-60 MB/second (conservatively taken), the time that would be spent for the transfer of 8 KB data is about 0.15 miliseconds. For the job addresses which are sent to Disksim, the jobs' starting points of the addresses are used. Hence, the job addresses are assumed as points, instead of lines. This assumption does little harm because of the constant data size assumption.

Job arrival rates and physical block addresses (PBA) are randomly generated using Minitab 13.1 statistical software package's random data calculation property. The poisson distribution that has been widely used for interarrival times of jobs in computer hard disk systems is used for generation of interarrival times of jobs. The random interarrival times (r_i) are generated using poisson distribution with 5 distinct arrival rates (λ), namely 125, 135, 150, 175, 180 job arrivals per second with corresponding mean interarrival times of 8, 7.41, 6.67, 5.71, 5.56 miliseconds, respectively. The arrival rates have been decided after some

preliminary simulation runs with several rates. For each rate, 30 sets are generated, each of which includes 1000 jobs. Then 5x30 job arrival time sets are produced by summing 1000 interarrival times line by line for each.

r_i ~ Poisson with λ

$\lambda = \{125, 135, 150, 175, 180\}$

For generating jobs' physical block addresses (PBA), uniform distribution, which has been used generally for that purpose in the literature, is used. Uniform distribution's range is chosen as the interval between 0 and 281,916,703. It is the physical block address range of the hard disk (146 GB) used within the simulation. Only 30 physical block address sets are generated.

PBA ~ Uniform[0 and 281,916,703]

The same random data set pairs are used for the sake of unbiased comparison in a single run of five policy simulations. For 30 randomly generated sets composed of arrival times and job addresses pairs, 30 replications are done for each policy. The label convention for that randomly generated sets are made in the following manner:

λ – P (for Poisson) – Set number **x** U (for Uniform) – Set number.

For instance, 150P12xU12 stands for 12th arrival time set formed with 150 Poisson arrival rate combined with 12th job address set formed with Uniform distribution.

The verification of the SDM-E application can be done investigating the simulation output provided in Appendix E for 50 PBA data

from 180P1-U1 random data pair and queue depth of 8 (where 8 is the length of the queue allowed for coming requests). “Seek + latency” column values are computed by Disksim within the simulation. The output of Disksim is not provided here since it is quite long.

As mentioned before the timeouts are allowed in our system. Anyway, for tracing them from outputs, we need a maximum allowable constant waiting time value (τ). We have chosen it as 1000 miliseconds that is a value used in common practice for modern hard disks.

5.4 Experimental Analysis

All the simulations (except SDM-E applications with λ={175, 180} because of the reason that will be explained under SDM-E subheading of Results for Conventional Approach) were run for 128 queue depth and 1000 jobs. The simulation outputs for three applications on conventional decision setting (Experiment 1) are provided in Appendices F, G and H, and those for two applications on decision epoch setting (Experiment 2) are provided in Appendices I and J. Results are analyzed separately on each decision setting for making inferences on the performances of them.

The performance measures used in the experiment analysis are as follows:

- *Service time:* The summation of seek, latency and transfer times for a job is the service time of that job (miliseconds - ms).

- *Throughput rate:* Number of serviced jobs per second (jobs/s).
  1000 (ms/s) / Average service time (ms/job).
- *Makespan (C_{max}):* The time at which the last job in system is served (ms).
- *Average Flow time ($\overline{F}$):* Average time that a job spend in system from its arrival time to its service completion time (ms).
- *Number of Timeouts (T):* Timeout is a flow time related measure. Actually the timeout case is very unacceptable one, since it reduces quality of service much. Hence, barely the number of timeouts cannot be a performance measure. But, the occurrence of timeouts can be taken as warning for poor service quality and even for system instability in some cases.

Results for Conventional Approach

Overall average service times (μ_s) and their corresponding standard deviations (σ_s) are given in Table 5.1 for three approaches in conventional decision setting. Since times are given in miliseconds, the average throughput rates by the algorithms (as serviced job per second) can be found by dividing 1000 miliseconds with corresponding average service time value. The average throughput rates are presented in Table 5.2.

As can be seen in Table 5.1 the average service time for FIFO is the same for all arrival rates. Since FIFO does not take any decision, service time components seek, latency and transfer time values are

independent of the arrival times so the arrival rates. They undergone service always in the same sequence with FIFO policy, and the same job addresses are used at every arrival rate. Hence, the average service time and the throughput rate result in a constant.

Table 5.1 – Conventional Approach-Average Service Times with Corresponding Standard Deviations (ms/job)*

Arrival Rate	FIFO		CLOOK		SDM-E	
	μ_s	σ_s	μ_s	σ_s	μ_s	σ_s
125	7.90	2.45	7.76	2.48	7.70	2.46
135			7.38	2.56	7.34	2.46
150			6.69	2.55	6.67	2.37
175			5.80	2.33	5.75	2.08
180			5.62	2.27	5.56	2.01

* μ_s & σ_s denote average service time and standard deviations.

Table 5.2 – Conventional Approach-Average Throughput Rates (jobs/second)

		FIFO	CLOOK	SDM-E
Arrival Rate	125	126.51	128.95	129.92
	135		135.46	136.33
	150		149.38	149.99
	175		172.55	174.02
	180		177.80	179.77

Table 5.3 shows the maximum flow times (F_{max}), the average flow times (μ_F), and corresponding standard deviations (σ_F) found by the algorithms at each arrival rate (for 30x1000 sets of data) in conventional decision setting.

The average flow time and flow time standard deviation changes as percentage from one arrival rate to the others are presented in Table 5.4 (taking 125 Hz as the basis).

Table 5.3 – Conventional Approach- Flow Time Results (ms)*

λ	FIFO			CLOOK			SDM-E		
	μ_F	σ_F	F_{max}	μ_F	σ_F	F_{max}	μ_F	σ_F	F_{max}
125	52.06	32.82	220.03	22.44	12.32	103.07	18.82	8.91	95.03
135	272.82	156.66	700.48	33.00	15.45	117.37	24.01	9.58	107.88
150	636.40	363.29	1449.6	45.33	20.77	153.59	28.54	10.66	164.27
175	1091.46	634.42	2358.7	74.80	40.18	278.18	33.03	15.35	230.82
180	1191.31	687.43	2567.5	86.05	47.69	317.18	34.22	16.81	202.05

* F_{max} denotes maximum flow time, μ_F & σ_F denote average flow time & standard deviations.

Table 5.4 – Conventional Approach- Percent Changes in Average Flow Times and Standard Deviations for Different Rates

Arrival Rate	FIFO		CLOOK		SDM-E	
	μ_F	σ_F	μ_F	σ_F	μ_F	σ_F
125	-	-	-	-	-	-
135	80.92%	79.05%	32.00%	20.26%	21.62%	6.99%
150	91.82%	90.97%	50.50%	40.68%	34.06%	16.42%
175	95.23%	94.83%	70.00%	69.34%	43.02%	41.95%
180	95.63%	95.23%	73.92%	74.17%	45.00%	47.00%

The average makespan values of 30 replications are presented for different rates in Table 5.5.

Table 5.5 – Conventional Approach- Average Makespan Values (ms)

Arrival Rate	C_{max} Average		
	FIFO	CLOOK	SDM-E
125	8060.73	8031.24	8027.40
135	7931.27	7442.99	7432.06
150	7922.59	6719.32	6694.00
175	7917.15	5830.95	5759.53
180	7916.67	5664.00	5574.85

The performances of three algorithms in conventional decision setting at different rates for different measures can be interpreted as follows:

FIFO

Table 5.1 shows that FIFO has biggest average service time with 7.90 miliseconds. As mentioned above, FIFO has a constant average service time, and so constant average throughput rate for every arrivals because of the reasons mentioned before. Actually, this is the main weakness of the algorithm. It reacts to increasing arrival rates with doing nothing. Hence, starting with the arrival rate of 135, as arrival rate increases, the requests are observed to be accumulated in system (saturation). Although, at the 135 Hz (arrival/second) queue depth violations are not observed, there is a tendency for long run (the maximum queue length is seen to be 92 at Table F.2 in Appendices). The average flow times, except those in λ=125, show increasing trend as Table 5.6 indicates. Table 5.6 shows the average flow time values for 250, 500 and 1000 jobs, whereas Table 5.7 gives flow time standard deviations for each case. The considerable increments in both average and standard deviation indicate that the system does not reach the steady state for arrival rates other than 125 (see tables F.2 to F.5 in Appendices for detailed statistics).

At 150 Hz, queue depth violations are seen for the first time. After the 150 Hz arrival rate, FIFO completely fails with queue depth violations in high numbers, high starvation, and of course timeouts. At 175 and 180 Hz arrival rates, average flow time value is even above the timeout value of 1000 ms (see Tables F.3 to F.5 in Appendices).

Table 5.6 – Conventional Approach- Average Flow Times for FIFO

	μ_F values for FIFO		
Arrival Rate	# of jobs		
	250	500	1000
125	35.01	44.01	52.06
135	84.41	143.60	272.82
150	169.38	323.46	636.40
175	275.11	543.26	1091.46
180	305.00	597.33	1191.31

Table 5.7 – Conventional Approach- Flow Time Standard Deviations for FIFO

	σ_F values for FIFO		
Arrival Rate	# of jobs		
	250	500	1000
125	19.20	26.52	32.82
135	46.17	78.96	156.66
150	96.56	182.98	363.29
175	157.22	312.67	634.42
180	173.57	341.38	687.43

The Table 5.8 shows the average and maximum numbers of tardy jobs having flow time of over 1000 miliseconds (T_{avg}, T_{max}) and also

the average and maximum numbers of jobs being tardy twice ($T2_{avg}$, $T2_{max}$) for each arrival rate. Since no timeout cases occurred for other policies, the table was designed only for FIFO. The timeout occurrences seen in Table 5.8 certainly proves that FIFO is not an acceptable policy for modern hard disks which are necessitating service of jobs coming into system with high arrival rates of more than 150 Hz.

Table 5.8– Conventional Approach-Average and Maximum Timeout Occurrences for FIFO (# of jobs being tardy once and twice)

		FIFO			
		T_{avg}	T_{max}	$T2_{avg}$	$T2_{max}$
Arrival Rate	125	-	-	-	-
	135	-	-	-	-
	150	209	337	-	-
	175	538	600	77	146
	180	577	635	160	238

CLOOK

At the 125 Hz arrival rate, CLOOK has 128.95 average throughput rate. It means that CLOOK can handle more requests, so that the system has several idle times waiting for arrival. The maximum queue length is 9 (see Table G.1 in Appendices).

CLOOK has almost reached steady state at the arrival rate of 135 Hz, with an average throughput rate of 135.46 jobs per second. The flow time variance, which was expected to be little comparing with other arrival rates, indicates the same situation.

At λ={150, 175, 180}, the average throughput rates are not far from these values. It is expected that, with increasing number of arrivals

the average throughput rate increases and gets near to the arrival rates, since the algorithm get more room for improvement. The algorithm is expected to show better performance as the number of jobs in the queue increases. Hence, most probably, the algorithm never permits the queue length exceeding some number. However, the service quality gets worse with increasing arrival rate since the flow times and their corresponding standard deviations increase also as Table 5.3 and Table 5.4 show.

In addition, after the 135 Hz arrival rate, while the average service time decreases, its corresponding average standard deviation shows also decreasing trend for CLOOK but with lower rate.

Table 5.9 and Table 5.10 show that the system has reached steady state for λ={125, 135, 150}, since the values do not show considerable changes with more arrivals. It is near to steady state for λ={175, 180}. These indicate the stability of system under CLOOK policy.

Table 5.9 – Conventional Approach- Average Flow Times for CLOOK

Arrival Rate	μ_F values for CLOOK		
	# of jobs		
	250	500	1000
125	22.74	22.90	22.44
135	30.95	32.11	33.00
150	43.10	44.42	45.33
175	65.39	70.94	74.80
180	71.74	80.81	86.05

Table 5.10 – Conventional Approach- Flow Time Standard Deviations for CLOOK

Arrival Rate	σ_F values for CLOOK		
	# of jobs		
	250	500	1000
125	12.09	12.33	12.32
135	14.83	15.51	15.45
150	21.05	20.78	20.77
175	36.57	38.76	40.18
180	41.39	45.64	47.69

SDM-E

As can be predicted before, SDM-E shows the best performance at all arrival rates. Especially Table 5.3 and 5.4 show superior performance of SDM-E in service quality. It outperforms CLOOK in the average flow time and its corresponding standard deviation values. SDM-E is very successful especially in holding the queue lengths at very small levels (see Tables 1 to 5 in Appendix H). For example, at the 175 Hz arrival rate, SDM-E seems not likely to allow the queue length to exceed 11-12 while at the same rate, CLOOK has a maximum queue length of 26. At the same arrival rate, SDM-E has an average flow time of 33.03, whereas that value is 74.80 for CLOOK, more than double of the value of SDM-E. A convention is used in SDM-E simulation as holding the queue depth as 8 at λ={175, 180} to avoid very long computational times, and not to make scheduling for the jobs coming when the queue depth is full (permitting 8 queue depth violation). While the actual queue depth of 128 is not violated (Q+RQ+pending jobs <= 128), the lowering of queue depth only means that the system does not make rescheduling for the jobs which are in queue depth violation.

Hence, the scheduling performance will be inferior comparing to 128 queue depth scheduling decisions. In spite of that, the performance of SDM-E is still superior comparing with other two policies.

Table 5.11 and Table 5.12 indicate that the steady state has been reached at about 1000th job's arrival at all arrival rates.

Table 5.11 – Conventional Approach- Average Flow Times for SDM-E

Arrival Rate	μ_F values for SDM-E		
	# of jobs		
	250	500	1000
125	18.81	19.03	18.82
135	23.52	23.76	24.01
150	27.93	28.14	28.54
175	32.42	32.73	33.03
180	33.41	34.06	34.22

Table 5.12 – Conventional Approach- Flow Time Standard Deviations for SDM-E

Arrival Rate	σ_F values for SDM-E		
	# of jobs		
	250	500	1000
125	8.79	8.96	8.91
135	9.60	9.62	9.58
150	11.36	10.86	10.66
175	15.24	14.92	15.35
180	16.85	16.83	16.81

The arrival rate versus the average makespan graph for the makespan values provided in Table 5.5 is presented in Figure 5.1.

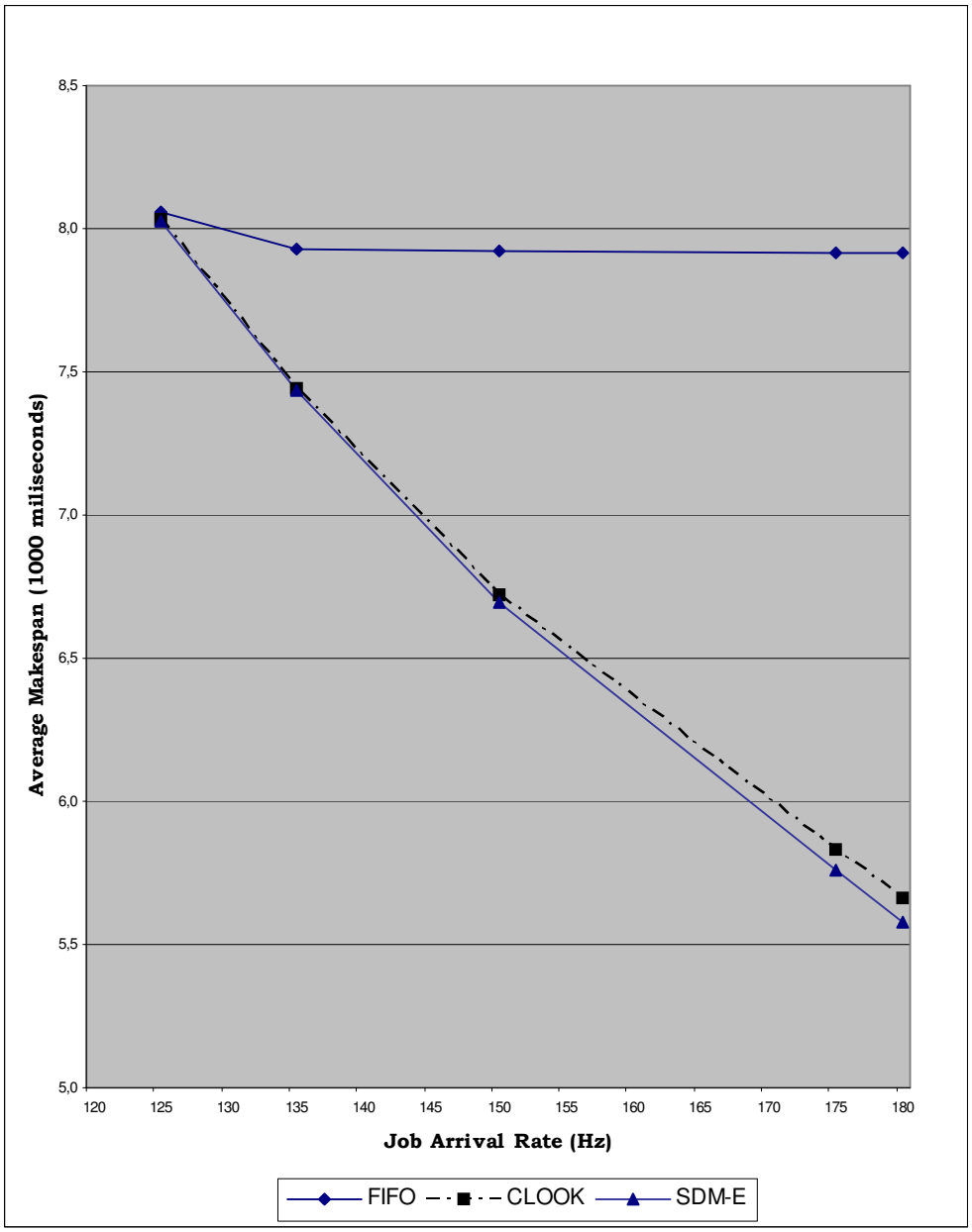

Figure 5.1 – Conventional Approach-Arrival Rate versus Average Makespan Values

The arrival rate versus average makespan graph illustrated in Figure 5.1 shows that CLOOK and SDM-E approaches to FIFO in makespan value at a low arrival rate of 125. The reason is that, the queues are not filled enough for these algorithms making use of the scheduling advantage over FIFO, at this low rate.
Since the interarrival times are greater for lower arrival rates, the makespan values show a decreasing trend for increasing arrival rates. But, this trend is followed very insignificantly by the FIFO approach, since it does nothing to make use of increasing arrival rates. In contrast to FIFO, the two algorithms enlarge the schedule domain as the arrival rate increases and queue starts to be filled. As the graph shows, the makespan values of CLOOK and SDM-E proceed very close in favor of SDM-E. However, it tends to widen as the arrival rate increases.

While the makespan objective guarantees the fastest operation, the average flow time is more important as it is the main indicator of the service stability. The number of jobs waiting in the queue is also important for monitoring the queue depth violations, which may cause system instability. The arrival rate versus the average flow time graph for the average flow time values provided in Table 5.3 is presented in Figure 5.2. For an unbiased comparison, the graph is provided in Figure 5.3 for percent changes given in Table 5.4. The detailed statistics for each experiment can be seen in Appendices F, G and H.

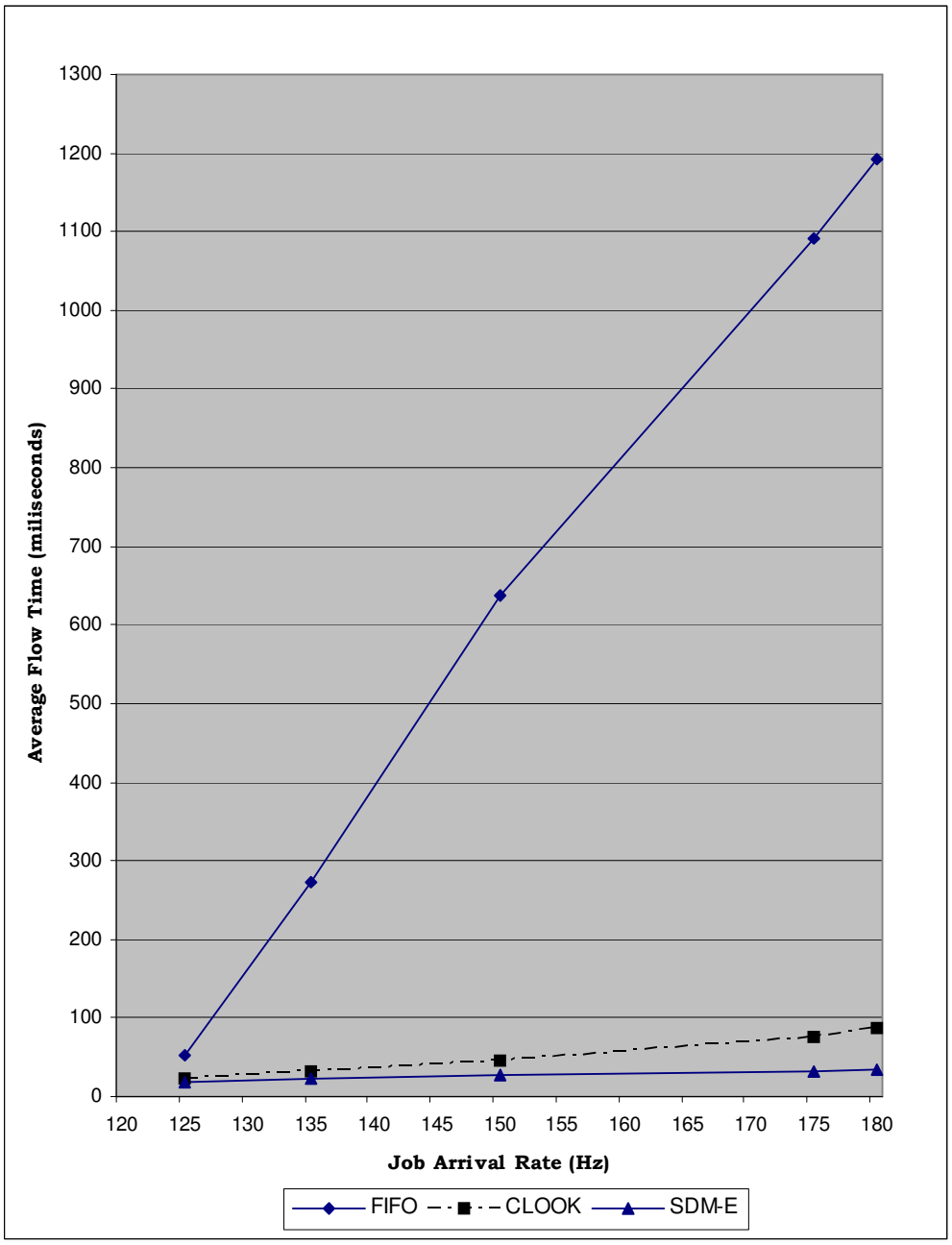

Figure 5.2 – Conventional Approach-Arrival Rate versus Average Flow Time

As can be seen from Figure 5.2, the SDM-E and CLOOK algorithms' flow times reach to that of FIFO at lower arrival rates.

But, we observe a significant difference in flow times between the two algorithms and FIFO, at the level of 135. After that level, the gap widenens, the FIFO is saturated, and the queue depth violations become inevitable.

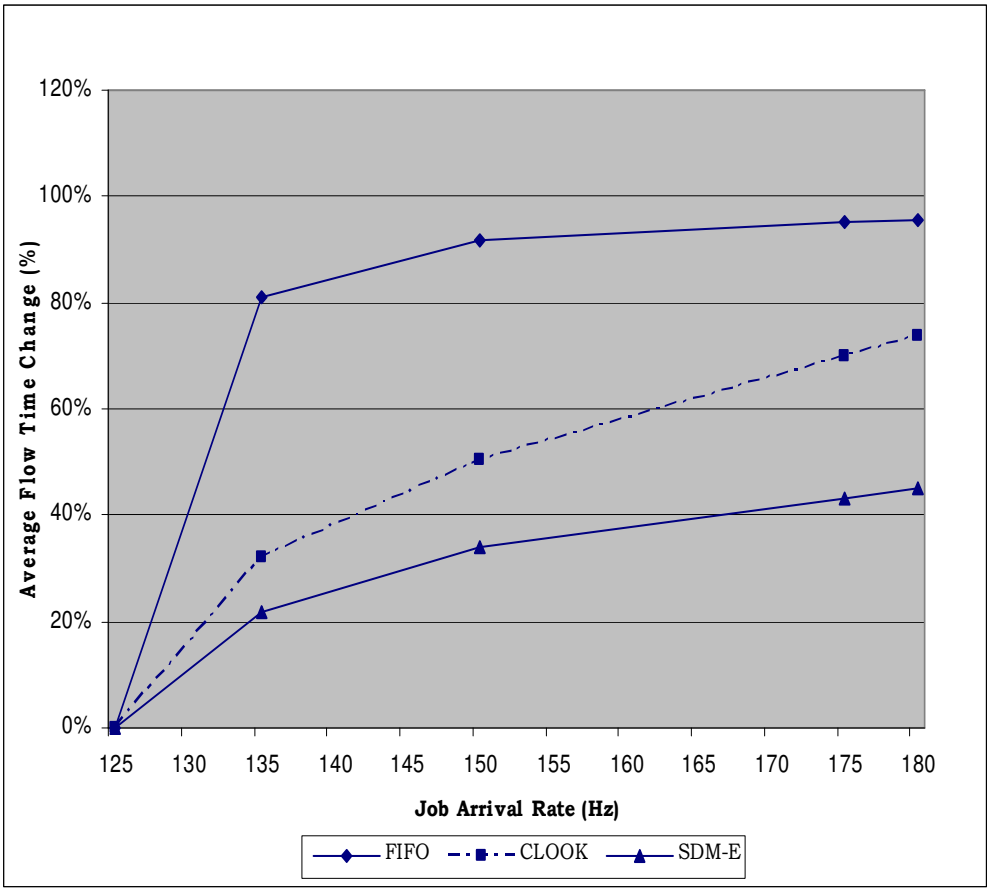

Figure 5.3 – Conventional Approach-Arrival Rate versus Average Flow Time Percent Change

Relative to the flow time values of FIFO, the flow times of SDM-E and CLOOK seem to behave similarly till 180 arrivals per second. But, when the values are considered as relative to each other, it is understood that SDM-E performs much better than CLOOK as

shown in Figure 5.3 indicates. Moreover, Figure 5.3 shows that the present gap between them indicates a widening trend in favor of SDM-E as the arrival rate increases.

Results for DE Approach

Overall average service times (μ_s) and their corresponding standard deviations (σ_s) are given in Table 5.13 and the corresponding average throughput rates are presented in Table 5.14 for two approaches in deterministic decision epoch (DE) setting.

Table 5.13 – DE Approach-Average Service Time Results with Corresponding Standard Deviations (ms/job)

Arrival Rate	CLOOK		SDM-NN	
	μ_s	σ_s	μ_s	σ_s
125	7.77	2.49	7.71	2.47
135	7.39	2.55	7.35	2.46
150	6.70	2.55	6.67	2.37
175	5.79	2.34	5.75	2.04
180	5.63	2.28	5.57	1.96

Table 5.14 – DE Approach-Average Throughput Rates (jobs/second)

		CLOOK	SDM-NN
Arrival Rate	125	128.75	129.72
	135	135.40	136.09
	150	149.31	149.91
	175	172.62	173.85
	180	177.78	179.56

Table 5.15 shows the maximum flow times (F_{max}), the average flow times (μ_F) and corresponding flow time standard deviations (σ_F) found by the algorithms at each arrival rate in DE decision setting (for 30x1000 sets of data). The makespan averages are presented in Table 5.16.

Table 5.15 – DE Approach- Flow Time Results (ms)

Arrival Rate	CLOOK			SDM-NN		
	μ_F	σ_F	F_{max}	μ_F	σ_F	F_{max}
125	22.82	12.49	112.72	19.06	9.32	124.27
135	34.38	15.48	121.00	24.67	10.57	148.52
150	46.80	20.91	150.72	29.62	12.63	181.63
175	75.36	39.81	270.50	34.25	18.68	259.68
180	86.22	47.38	325.66	35.76	20.32	357.95

Table 5.16 – DE Approach- Percent Changes in Average Flow Times and Standard Deviations for Different Rates

Arrival Rate	CLOOK		SDM-NN	
	μ_F	σ_F	μ_F	σ_F
125	-	-	-	-
135	33.62%	19.32%	22.74%	11.83%
150	51.24%	40.27%	35.65%	26.21%
175	69.72%	68.63%	44.35%	50.11%
180	73.53%	73.64%	46.70%	54.13%

Table 5.17 – DE Approach- Average Makespan Values (ms)

Arrival Rate	C_{max} Average	
	CLOOK	SDM-NN
125	8031.73	8028.71
135	7442.28	7435.09
150	6723.44	6697.95
175	5835.43	5773.84
180	5667.02	5590.43

The performances of two algorithms in DE decision setting at different rates for different measures can be interpreted as follows:

CLOOK

At λ={125, 135}, CLOOK has 128.75 and 135.40 average throughput rates. It means that CLOOK can handle more requests as in the case of conventional approach, so that the system has several idle times waiting for arrival. The maximum queue lengths are 10 and 12 jobs (see Tables I.1 and I.2 in Appendices). At λ={150, 175, 180}, the average throughput rates are not far from those values of conventional approach. Again, the service quality gets worse with increasing arrival rate since the flow times and their corresponding standard deviations increase also as Table 5.15 and 5.16 show. These values also show similarity to that of conventional approach.

Table 5.18 shows the average flow time values for 250, 500 and 1000 jobs, whereas Table 5.19 gives standard deviations for each case. They indicate that the system has almost reached the steady state for λ={125, 135, 150}, whereas it is near to steady state for λ={175, 180} (see tables I.1 to I.5 in Appendices for detailed statistics).

Table 5.18 – DE Approach- Average Flow Times for CLOOK

Arrival Rate	μ_F values for CLOOK		
	# of jobs		
	250	500	1000
125	22.65	23.16	22.82
135	32.47	33.48	34.38
150	44.00	45.96	46.80
175	64.90	70.83	75.36
180	71.98	80.66	86.22

Table 5.19 – DE Approach- Flow Time Standard Deviations for CLOOK

Arrival Rate	σ_F values for CLOOK		
	# of jobs		
	250	500	1000
125	12.06	12.58	12.49
135	14.77	15.56	15.48
150	20.75	21.04	20.91
175	35.19	38.04	39.81
180	40.92	44.72	47.38

SDM-NN

SDM-NN shows better performance than CLOOK at all arrival rates. As can be seen in Table 5.15 and 5.16, it cannot be distinctly seen at lower rates (125 and 135), but as the rate increases, the average flow time increase in SDM-NN is very little comparing with CLOOK. Hence, SDM-NN shows superior performance in service quality. It outperforms CLOOK in the average flow time and its corresponding standard deviation values and that advantage becomes clearer as the arrival rate increases. SDM-NN is very successful especially in holding the queue lengths at very small levels (see Tables 1 to 5 in Appendix J). For example, at λ={175}, SDM-NN seems not likely to allow the queue length to exceed 11-12 jobs while at the same rate, CLOOK has a maximum queue length of 25 jobs.

As Table 5.20 and Table 5.21 indicate, the system has been reached or very near to steady state for all arrival rates (see tables J.1 to J.5 in Appendices for detailed statistics).

Table 5.20 – DE Approach- Average Flow Times for SDM-NN

Arrival Rate	μ_F values for SDM-NN		
	# of jobs		
	250	500	1000
125	19.29	19.38	19.06
135	24.06	24.44	24.67
150	28.82	29.20	29.62
175	33.08	33.60	34.25
180	34.57	35.36	35.76

Table 5.21 – DE Approach- Flow Time Standard Deviations for SDM-NN

Arrival Rate	σ_F values for SDM-NN		
	# of jobs		
	250	500	1000
125	9.37	9.44	9.32
135	10.27	10.61	10.57
150	13.05	12.76	12.63
175	18.27	18.30	18.68
180	19.72	20.14	20.32

The arrival rate versus the average makespan graph is presented in Figure 5.4. The arrival rate versus the average flow time graph is presented in Figure 5.5. For an unbiased comparison, the graph displaying percent changes is given in Figure 5.6. The detailed statistics for each experiment can be seen in Appendices I and J.

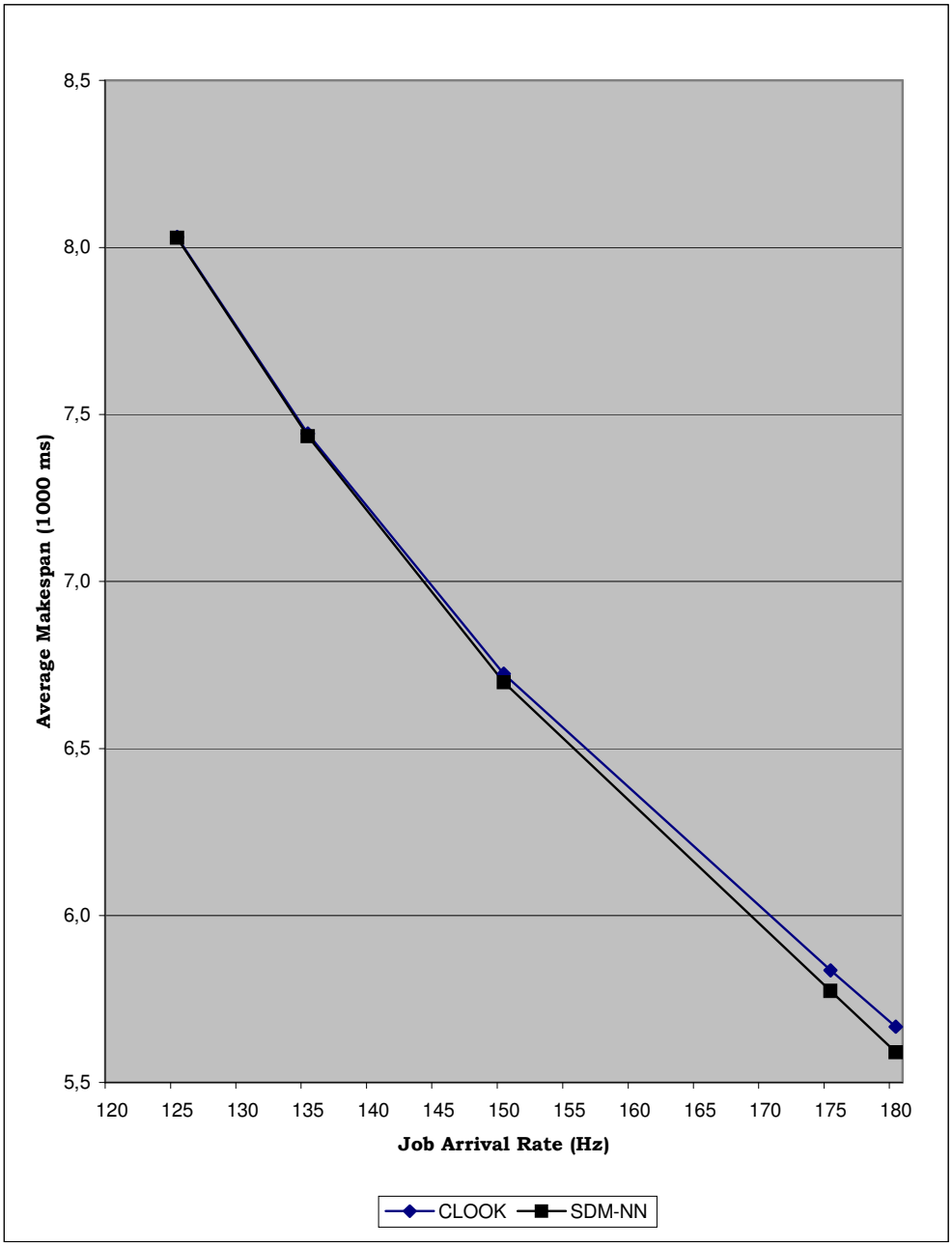

Figure 5.4 – DE Approach-Arrival Rate versus Average Makespan Values

The arrival rate versus average makespan graph illustrated in Figure 5.4 shows that CLOOK and SDM-NN perform almost the same, with better performance of SDM-NN as the arrival rate is

getting bigger, so SDM-NN widens the gap as the arrival rate increases. Both of the two algorithms enlarge the schedule domain as the arrival rate increases and queue starts to be filled.

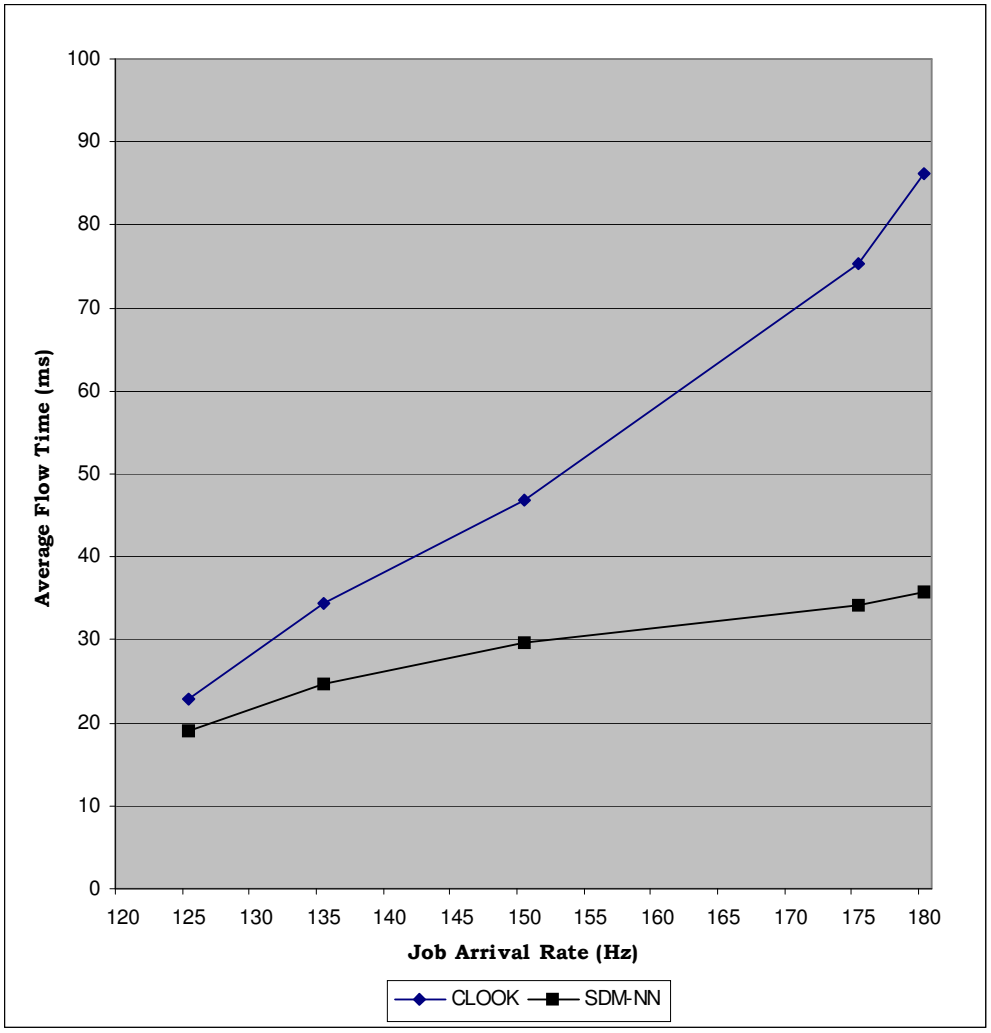

Figure 5.5 – DE Approach-Arrival Rate versus Average Flow Time

As can be seen from Figure 5.5, the main advantage of SDM-NN over CLOOK algorithm is its superior performance in average flow time near to that of SDM-E of conventional approach. As the graph in Figure 5.6 shows, at every increase in arrival rate, the average

flow time increase in SDM-NN is also relatively low comparing with that of CLOOK.

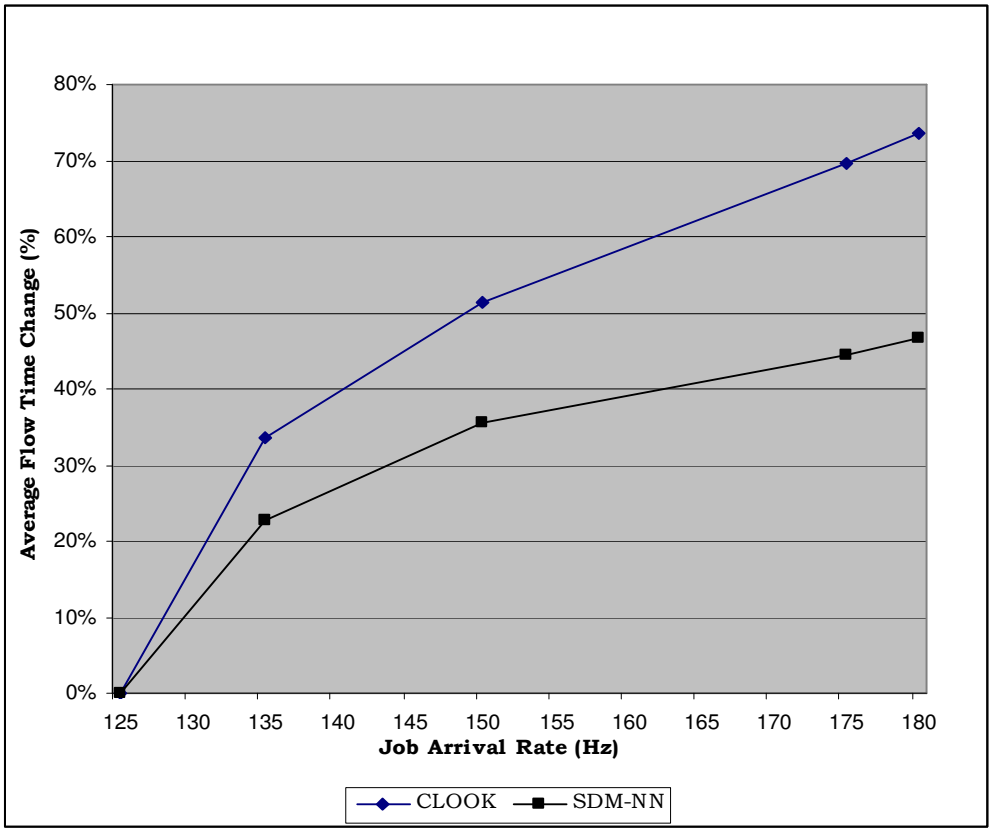

Figure 5.6 – DE Approach-Arrival Rate versus Average Flow Time Percent Change

The graph shows that the present gap between them indicates a widening trend in favor of SDM-NN especially after the 150 Hz arrival rate. Hence, SDM-NN has the short average flow time advantage of SDM-E without high computational requirements.

CHAPTER 6

CONCLUSIONS AND FURTHER STUDY

In the study, the classical disk scheduling problem has been investigated under two categories, offline and online settings. A generic mathematical model is developed for the offline problem as an adaptation of traveling salesman problem with time windows for the offline problem. It is given four different performance criteria: makespan, makespan with due date, number of tardy jobs, and total (or average) flow time. The offline problem is a deterministic problem, whereas the online problem is a stochastic one in which the information comes over time. The arrival times and addresses are not known until requests are realized. However, online problem could be handled by solving consecutive offline problems, when it is possible.

For online problem a deterministic decision epoch (DE) concept is introduced for the decision making problem in disk scheduling for the purpose of reducing stochastic decision structure to a deterministic one. In addition to conventional decision approach, which is based on taking decision at every new arrival and/or timeout, DE approach was implemented within the study.

In conventional decision approach, the simulations developed for the online problem used the offline problem for the frozen state of the system. By this way, the exact solution of sequence dependent makespan minimization (SDM-E) is implemented. Two more

simulations also are developed for the FIFO and CLOOK algorithms for benchmarking purposes.

After an experimentation, it has been seen that the SDM-E showed a better performance over CLOOK with a slight margin in makespan criterion in conventional decision setting. Apparently, the margin has been widening as the arrival rate is getting larger. In average flow time criterion, especially, after the rate of 150 jobs, SDM-E performs much better than CLOOK. FIFO, especially for the arrival rates greater than 125, has showed an inferior performance. Experimentation implies that FIFO is certainly not an acceptable policy for modern hard disk systems necessitating high throughput rates.

Although the SDM-E performs so well, its high computational time requirements do not support practical applications. Hence, we applied the Nearest Neighbor TSP heuristic for the approximate solution of sequence dependent makespan minimization problem (SDM-NN) and assessed its performance over CLOOK under DE setting.

After an experimentation under DE setting, it is found that the improvement by SDM-NN over CLOOK is not high for makespan criterion. But the average flow time by SDM-NN is superior compared to CLOOK and getting more and more advantageous than CLOOK as the arrival rate increases. Queue lengths also show the same behaviour in favor of SDM-NN. SDM-NN shows almost same performance to SDM-E applied under conventional setting but does not bear the computational disadvantage of it. Hence, it is found possible to improve the quality of service by

implementing SDM-NN approach having similar computational requirements with CLOOK.

In DE setting, every time interval between consecutive completion of processing of jobs is taken as an available time for making decisions. However, that epoch may not be enough in high arrival rates or it may be too short for enough number of jobs coming into the system for making the effective decisions in rich domains. Hence a DE-k system could be developed in which, the DM waits for k consecutive completions instead of just one as s/he does in our system. This is left to a further study.

In addition to the experimentation with artificially generated data, an experimentation with the real data would be helpful for assessing the validity of our conclusions. It would also help assessing the validity of the assumptions that job addresses are uniformly distributed over disk surface and job interarrival times conform to Poisson distribution. This work is left to a further study.

An interesting future research subject in the disk scheduling area is about the possible performance improvements for a disk system having two queues instead of one. This is left to a further study too. A brief information on the subject can be found in Appendix H.

REFERENCES

Albers, Suzanne, S. Leonardi. 1999. On-line algorithms. *ACM Computing Surveys, Vol.* ***31*** *, Article No. 4.*

Andrews, M., M. A. Bender, and L. Zhang. 2002. New algorithms for disk scheduling. *Algorithmica, Vol.* ***32*** 277-301.

Ascheuer, N., M. Fischetti, and M. Grötschel. 2001. Solving the asymmetric traveling salesman problem with time windows by branch-and-cut. *Mathematical Programming, Ser. A* ***90*** 475-506.

Balakrishnan, N., J.J. Kanet, and S.V. Sridharan. 1999. Early/tardy scheduling with sequence dependent setups on uniform parallel machines. *Computers & Operations Research, Vol.* **26** 127-141.

Ben-Daya, M., and M.Al-Fawzan. 1998. A tabu search approach for the flow shop scheduling problem. *European Journal of Operational Research, Vol.* **109** 88-95.

Bonifaci, Vincenzo. 2005. Online vehicle routing problems *http://www.dis.uniroma1.it/~dottorato/db/relazioni/relaz_bonifaci_2. pdf.*

Burkhard, W.A., and J.D. Palmer. 2001. Rotational position optimization (RPO) disk scheduling. *Proceedings of the First Conference on File and Storage Technologies (FAST'02), January 28-29, 2002, Monterey, California.*

Chang, MS., SR. Chen, CF. Hsueh. 2003. Real-time vehicle routing problem with time windows and simultaneous delivery/pickup demands. *Journal of the Eastern Asia Society for Transportation.*

Correa, Jose R., and Michael R. Wagner. 2005. LP-based online scheduling: From single to parallel machines. *Lecture Notes in Computer Science, Vol.* ***3509****, Springer Berlin / Heidelberg,* 196-209.

Denning, P.J. Effects of scheduling on file memory operations. *Proc. AFIPS 1967 SJCC, Vol.* ***30****, AFIPS Press, Montvale, N.J., 9-21.*

Desrosiers, J., Y. Dumas, M. M. Solomon, and F. Soumis. 1995. Network routing. *Handbook on Operations Research and Management Science,* Vol. ***8****, North Holland, Amsterdam,* 35-139.

Ganger, G.R. 1995. Generating representative synthetic workloads: An unsolved problem. *Proceedings of the Computer Measurement Group (CMG) Conference,* 1263-1269.

Geist, R, and S. Daniel, 1987. A continuum of disk scheduling algorithms. *ACM Transactions on Computer Scheduling, Vol.* ***5(1)*** 77-92.

Huang, L., and T.C. Chiueh. 2002. Experiences in building a software-based SATF scheduler. *State University of New York at Stony Brook Technical Report, www.ecsl.cs.sunysb.edu/tr/TR81.ps.*

Jacobson, M., and J. Wilkes. 1991. Disk scheduling algorithms based on rotational position. *HP Laboratories Technical Report.*

Kordon, A.M., and J.B. Note. 2005. A buffer minimization problem for the design of embedded systems. *European Journal of Operational Research Vol.* ***164*** 669-679.

Minitab Inc. 2000. Minitab 13.1 - *Statistical Software Package*.

Nutt, G. 1999. Operating systems – A modern perspective, Addison-Wesley, 116-122.

Pandit, V., and R. Khandekar. 2005. Online sorting buffers on line. *IBM Research Division Technical Report.*

Popovici, F.I., A.C. Arpacı-Dusseau, and R.H. Arpacı-Dusseau. 2003. Robust, portable I/O scheduling with the disk mimic. *Proceedings of the USENIX 2003 Annual Technical Conference, June 9-14, 2003, San Antonio, Texas.*

Rabadi, G. 2001. Application of constraint programming to a scheduling problem with a common due date and setup times. *Proceedings of the First International Industrial Engineering Conference, Sept. 23-27, 2001, Amman, Jordan.*

Reinelt, Gerhard. 1994. The traveling salesman: Computational solutions for TSP applications. *Lecture Notes in Computer Science Vol.* ***840***. *Springer-Verlag.*

Reuther, L., and M. Pohlack. 2003. Rotational-position-aware real-time disk scheduling using a dynamic active subset (DAS). *Proceedings of the 24th IEEE International Real-Time Systems Symposium, December, 2003, Cancun, Mexico.*

Seltzer M., P. Chen, and J. Ousterhout. 1990. Disk scheduling revisited. *Proceedings of the Winter Usenix, January 1990, Washington, D.C.*

Silberschatz, A., P.B. Galvin, and G. Gagne. 2002. Operating system concepts 6th ed., John Wiley & Sons, 491-536.

Sleator, D.D., and R.E. Tarjan. 1985. Amortized efficiency of list update and paging rules. *Communications of the ACM, Vol.* ***28*** 202-208.

Stallings, W. 2001. Operating systems: Internals and Design Principles 4th ed., Prentice Hall, 486-493.

Teorey, T., and T. Pinkerton. 1972. A comparative analysis of disk scheduling policies. *Communications of the ACM, Vol.* ***15(3)*** 177-184.

Thomasian, A., C. Liu. 2002. Some new disk scheduling policies and their performance. *ACM SIGMETRICS Performance Evaluation Review Vol.* ***30*** *, Issue 1. Measurement and modeling of computer systems*, 266 – 267.

Winter, T., and U. T. Zimmermann. 1998. Discrete online and real-time optimization. *Proceedings of the 15th IFIP World Computer Congress, Budapest / Vienna, August 31 - September 4, 1998.*

Worthington, B.L., G.R. Ganger, and Y.N. Pratt. 1994. Scheduling algorithms for modern disk drives. *Proceedings of the ACM Sigmetrics Conference, May, 1994*, 241-251.

APPENDIX A

A.1 Simulation Code Framework for Conventional Decision Approach

Where:

i: job identity index

j: job processing index

Seq (1..n) : Job processing array

Qseq (1..queue_depth) : Array of Jobs in Queue, subject to change in Reschedule procedures

```
for i=1 to n
    if      r_i < c          "the next arrival before completion of job in process
        if      RQ1 = 0
                RQ1 = 1
                RQ(1) = i
        if      RQ1 = 1 and RQ2 = 0
                RQ2 = 1
                RQ(2) = i
        if      Q >= 0
                Q = Q+1
                Qseq(Q) = i
                if      Q > 1
                        Reschedule (Q)
        i = i + 1
    else if     r_i > c      "completion of job in process before the new arrival
```

```
if      RQ1 = 0
        Seq(j) = i
        c = c + S_{j-1,j} +t
        j = j + 1
if      RQ1 = 1 and RQ2 = 0
        Seq(j) = RQ(1)
        RQ(1) = i
        c = c + S_{j-1,j} +t
        j = j + 1
        if      r_i > c         "more than one completion before the arrival
                Seq(j) = RQ1
                RQ1 = 0                          } JobSlipping
                c = c + S_{j-1,j} +t
                j = j + 1
if      RQ1 = 1 and RQ2 = 1 and Q = 0
        Seq(j) = RQ1
        RQ1 = RQ2
        RQ2 = i
        c = c + S_{j-1,j} +t
        j = j + 1
        while           r_i > c
                JobSlipping
if      RQ1 = 1 and RQ2 = 1 and Q > 0
        Seq(j) = RQ1
        RQ1 = RQ2
        RQ2 = Qseq(1)
        c = c + S_{j-1,j} +t
        j = j + 1
        for k = 1 to Q-1
                Qseq(k) = Qseq(k+1)
        Qseq(Q) = i
```

Reschedule(Q)

while $r_i > c$

JobSlipping

while $j < n$

JobSlipping

terminate.

A.2 Simulation Code Framework for DE Approach

Where:

i : job identity index

ind: job processing index

Comp(ind): Completion time of the job which is processed in "ind" sequence.

Seq (1..n) : Job processing array

Qseq (1..queue_depth) : Array of Jobs in Queue, subject to change in Reschedule procedures

```
for i=1 to n
    if      c > r_i          "the next arrival before completion of job in process
        if      RQ1 = 0
                ind = ind+1
                Seq(ind) = i
                RQ1 = 1
                RQ(1) = i
        if      RQ1 = 1 and RQ2 = 0
                ind = ind+1
                Seq(ind) = i
                RQ2 = 1
                RQ(2) = i
        if      Q >= 0
                Q = Q+1
                Qseq(Q) = i
    else if     c <= r_i     "completion of job in process before the new arrival
        if      RQ1 = 0
                ind = ind+1
                Seq(ind) = i
```

```
                Comp(ind) = r_i + S_Seq(ind-1),Seq(ind) +t
                c = Comp(ind)
        if      RQ1 = 1 and RQ2 = 0
                RQ(1) = I
                ind = ind+1
                Comp(ind-1) = c + S_Seq(ind-2),Seq(ind-1) +t
                c = Comp(ind-1)
                if      c <= r_i        "more than one completion before the arrival
                        RQ1 = 0                                          }
                        Comp(ind) = r_i + S_Seq(ind-1),Seq(ind) +t       } Job Slipping
                        c = Comp(ind)                                    }
        if      RQ1 = 1 and RQ2 = 1 and Q = 0
                RQ1 = RQ2
                RQ2 = i
                ind = ind+1
                Comp(ind-2) = c + S_Seq(ind-3),Seq(ind-2) +t
                c = Comp(ind-2)
                while           c <= r_i
                        JobSlipping
        if      RQ1 = 1 and RQ2 = 1 and Q > 0
                Reschedule (Q)
                RQ1 = RQ2
                RQ2 = Qseq(1)
                for k = 1 to Q-1
                        Qseq(k) = Qseq(k+1)
                Qseq(Q) = i
                while           c <= r_i
                        JobSlipping
while   ind < n
        JobSlipping
terminate.
```

APPENDIX B

CLOOK Code Framework

```
if Qseq(1) < RQ(2)
        if newjob > RQ(2)
                Qseq(1) = newjob
                Reorder Qseq
        else if newjob < RQ(2)
                i = 1
                while newjob >Qseq(i)  and  i <= Q
                        i = i + 1
                Qseq(i) = newjob
                Reorder Qseq
else if Qseq(1) >= RQ(2)
        if      newjob >= RQ(2)
                i = 1
                while   Qseq(i)>RQ(2)  and  newjob>Qseq(i)  and  i<=Q
                        i = i +1
                Qseq(i) = newjob
                Reorder Qseq
        else if newjob < RQ(2)
                i = 1
                while   Qseq(i)>RQ(2)  and  i<=Q
                        i = i +1
                if  i > Q
                        Qseq(i) = newjob
                else if i <= Q
```

```
while Qseq(i) < newjob  and  i <= Q
        i = i +1
Qseq(i) = newjob
Reorder Qseq
```

terminate.

APPENDIX C

SDM-NN Code Framework

```
for i=1 to |Q_{j-1}|
        if setup time of i^th member of Q_{j-1} depending on {RQ2}_j is
        smaller than that of New(1)
                New(1) = i^th member
next i
Newseq(1) = Qseq(New(1)

for i=2 to |Q_{j-1}|
        for k=1 to |Q_{j-1}|
                z=1,   control=0
                while control=0 and z<i
                        if k=New(z)
                                control=1
                        z=z+1
                if control=0
                        Newdist=s_{i,j} (Newseq(i-1), Qseq(k))
                if Newdist<Dist
                        Dist=Newdist
                        New(i)=k
        next k
next i
```

APPENDIX D

Lingo 8.0 TSP Modeling Interface Code

```
MODEL:
DATA:
NCITY= @pointer(1);
ENDDATA

SETS:
CITY/1..NCITY/:;
LINK(CITY,CITY):DIST,X;
COMP(CITY):C;
ENDSETS

[OBJECTIVE] MIN = CMAKS ;
N=@SIZE(CITY);
@FOR(CITY(J):@FOR(CITY(K)|J#EQ#K:X(J,K)=0;));
@FOR(CITY(K)|K#NE#1:@SUM(CITY(I)|I#NE#N:X(I,K))=1;);
@FOR(CITY(K)|K#NE#N:@SUM(CITY(J)|J#NE#1:X(K,J))=1;);
@SUM(CITY(J):X(J,1))=0;
@FOR(CITY(J):@FOR(CITY(K)|J#NE#K#AND#K#GT#1:
C(K)>=DIST(J,K)-100000*(1-X(J,K));));
@FOR(CITY(J):@FOR(CITY(K)|J#NE#K#AND#K#GT#1:
C(K)-C(J)+100000*(1-X(J,K))>=DIST(J,K);));
@FOR(LINK:@BIN(X););
@FOR(CITY(J):CMAKS>=C(J););
DATA:
DIST=@pointer(2);
@pointer(3)=OBJECTIVE;
@pointer(4)=@status();
@TEXT( C:\MYFILE.OUT) = X;
ENDDATA
END
```

APPENDIX E

Sample SDM-E Simulation Formatted Output for 50 jobs

Job Identity(i)	Arrival Time(r_i)	Comp. Time(c_i)	Seek +Latency	FlowTime (c_i - r_i)	Jobs in System at r_i								
					Serv.	RQ1	RQ2	Q1	Q2	Q3	Q4	Q5	Q6
1	11	11.15	0	0.15	1								
2	16	27.03	10.876	11.03	2								
3	20	36.03	8.852	16.03	2	3							
4	29	41.6	5.419	12.6	3	4							
5	35	52.99	11.243	17.99	3	4	5						
6	39	61.8	8.66	22.8	4	5	6						
7	45	71.31	9.358	26.31	5	6	7						
9	51	76.25	4.791	25.25	5	6	7	8	9				
8	49	82.6	6.206	33.61	5	6	7	8					
11	65	88.26	5.506	23.26	7	9	8	10	11				
10	63	95	6.584	32	7	9	8	10					
12	74	99.78	4.633	25.78	9	8	11	10	12				
13	81	104.54	4.612	23.54	8	11	10	12	13				
14	91	109.54	4.854	18.54	10	12	13	14					
15	98	115.65	5.956	17.65	12	13	14	15					
16	102	118.89	3.088	16.89	13	14	15	16					
17	105	128.88	9.84	23.88	14	15	16	17					
18	112	137.86	8.833	25.86	15	16	17	18					
19	121	141.98	3.973	20.98	17	18	19						
20	128	150.67	8.533	22.67	17	18	19	20					
21	137	160.16	9.348	23.16	18	19	20	21					
22	145	168.01	7.692	23.01	20	21	22						
23	148	176.82	8.668	28.83	20	21	22	23					
24	152	185.58	8.61	33.59	21	22	23	24					
25	158	192.04	6.3	34.03	21	22	23	24	25				
26	164	196.89	4.705	32.89	22	23	24	25	26				
29	173	201.91	4.872	28.91	23	24	25	26	28	27	29		
31	186	208.18	6.123	22.18	25	26	29	28	27	30	31		
27	169	213.52	5.182	44.52	23	24	25	26	27				
30	178	217.9	4.229	39.9	24	25	26	29	28	27	30		
35	207	222.69	4.644	15.69	31	27	30	33	28	32	34	35	
33	191	229.94	7.099	38.94	25	26	29	28	32	30	31	27	33
28	171	236.11	6.021	65.11	23	24	25	26	27	28			
32	187	241.28	5.022	54.28	25	26	29	28	27	30	31	32	

Sample SDM-E Simulation Formatted Output for 50 jobs (continued)

38	228	243.76	2.33	15.76	33	28	32	34	37	36	38		
34	199	245.77	1.857	46.77	29	31	27	30	33	28	32	34	
36	218	249.81	3.894	31.81	35	33	28	32	34	36			
39	233	257.1	7.141	24.1	28	32	38	34	37	36	39		
41	242	259.77	2.517	17.77	38	34	36	39	37	40	41		
37	225	271.8	11.883	46.8	33	28	32	34	36	37			
43	254	277.74	5.786	23.74	39	41	37	42	40	43			
42	249	286.33	8.44	37.33	36	39	41	37	40	42			
40	235	288.87	2.394	53.87	28	32	38	34	39	37	36	40	
45	264	291.59	2.563	27.59	37	43	42	40	44	45			
47	279	296.24	4.5	17.24	42	40	45	46	44	47			
48	285	301.25	4.863	16.25	42	40	45	47	44	46	48		
44	261	306.35	4.948	45.35	37	43	42	40	44				
50	294	311.21	4.712	17.21	47	48	44	46	49	50			
46	272	316.17	4.815	44.17	43	42	40	45	44	46			
49	288	323.24	6.916	35.24	40	45	47	48	44	46	49		

APPENDIX F

FIFO Simulation Average Results in Conventional Approach

Table F.1 – Results for FIFO at 125 Hz Arrival Rate

125	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
FIFO 1	7.92	7966.55	145.60	68.26	18
FIFO 2	8.00	8103.43	170.33	72.15	21
FIFO 3	7.88	8103.79	124.81	34.63	16
FIFO 4	7.97	8008.61	220.03	113.27	26
FIFO 5	7.82	8057.14	64.90	26.43	9
FIFO 6	8.04	8064.79	157.22	103.25	19
FIFO 7	7.84	8076.99	88.01	29.39	11
FIFO 8	8.01	8119.89	134.64	53.29	17
FIFO 9	7.92	8179.38	76.51	27.12	9
FIFO 10	7.95	7995.59	197.03	87.69	25
FIFO 11	7.95	8083.55	115.13	58.99	15
FIFO 12	7.76	7978.05	85.37	31.81	10
FIFO 13	7.95	8020.52	173.16	82.83	21
FIFO 14	7.77	7982.41	125.66	45.06	15
FIFO 15	7.97	8082.42	77.39	35.89	9
FIFO 16	7.85	8017.28	101.98	36.22	12
FIFO 17	7.79	8109.55	76.77	24.97	9
FIFO 18	7.85	8135.58	54.78	22.36	6
FIFO 19	7.69	7936.82	76.53	25.68	9
FIFO 20	7.87	8053.65	123.71	47.92	16
FIFO 21	7.95	8130.67	109.39	39.79	13
FIFO 22	7.92	8096.53	102.53	31.23	12
FIFO 23	7.90	8163.90	103.37	35.74	13
FIFO 24	8.00	8022.76	138.04	69.45	17
FIFO 25	8.01	8079.22	169.67	82.83	20
FIFO 26	7.89	8016.17	122.53	46.88	14
FIFO 27	7.93	8058.22	82.23	34.59	11
FIFO 28	7.89	7998.02	172.18	76.94	23
FIFO 29	8.05	8089.61	182.19	90.46	21
FIFO 30	7.81	8090.72	76.97	26.53	9
FIFO	7.90	8060.73	220.03	52.06	26

Table F.2 - Results for FIFO at 135 Hz Arrival Rate

135	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
FIFO 1	7.92	7938.10	653.57	306.88	79
FIFO 2	8.00	8006.24	700.48	392.34	85
FIFO 3	7.88	7897.07	411.42	244.33	53
FIFO 4	7.97	8000.87	603.02	332.75	76
FIFO 5	7.82	7838.72	291.68	147.86	34
FIFO 6	8.04	8065.79	649.02	422.24	79
FIFO 7	7.84	7854.98	422.98	184.62	57
FIFO 8	8.01	8047.17	698.17	381.87	85
FIFO 9	7.92	7927.89	644.47	335.12	82
FIFO 10	7.95	7973.34	687.34	334.13	89
FIFO 11	7.95	7988.10	598.84	249.36	74
FIFO 12	7.76	7786.23	417.35	216.81	52
FIFO 13	7.95	7987.43	487.04	241.98	61
FIFO 14	7.77	7820.37	573.85	260.86	72
FIFO 15	7.97	7984.75	646.95	341.65	81
FIFO 16	7.85	7880.54	503.98	270.29	66
FIFO 17	7.79	7798.54	334.36	180.58	42
FIFO 18	7.85	7870.49	448.86	188.73	57
FIFO 19	7.69	7721.41	305.92	127.93	40
FIFO 20	7.87	7925.62	605.62	258.09	73
FIFO 21	7.95	8008.14	476.07	265.12	60
FIFO 22	7.92	7945.99	679.99	329.48	81
FIFO 23	7.90	7913.59	689.57	322.88	92
FIFO 24	8.00	8011.85	572.85	267.59	71
FIFO 25	8.01	8024.45	568.64	282.68	70
FIFO 26	7.89	7927.17	519.76	262.79	66
FIFO 27	7.93	7959.04	517.87	254.76	65
FIFO 28	7.89	7907.22	419.71	236.30	56
FIFO 29	8.05	8087.26	614.62	357.34	78
FIFO 30	7.81	7839.69	428.60	187.19	53
FIFO	7.90	7931.27	700.48	272.82	92

Table F.3 - Results for FIFO at 150 Hz Arrival Rate

150	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
FIFO 1	7.92	7937.40	1275.51	615.83	154
FIFO 2	8.00	8015.24	1259.01	660.78	159
FIFO 3	7.88	7891.63	1200.28	656.52	157
FIFO 4	7.97	7981.06	1164.16	587.99	146
FIFO 5	7.82	7836.35	1338.77	732.36	168
FIFO 6	8.04	8067.79	1316.85	670.12	166
FIFO 7	7.84	7859.49	1193.49	601.43	153
FIFO 8	8.01	8028.91	1268.91	646.83	153
FIFO 9	7.92	7942.90	1228.90	642.72	156
FIFO 10	7.95	7972.89	1356.89	661.83	176
FIFO 11	7.95	7959.89	1269.89	617.68	159
FIFO 12	7.76	7772.20	1145.73	508.43	146
FIFO 13	7.95	7956.14	1225.12	619.40	155
FIFO 14	7.77	7803.14	1095.13	558.06	137
FIFO 15	7.97	7983.75	1266.20	624.05	161
FIFO 16	7.85	7860.28	1326.86	674.31	175
FIFO 17	7.79	7808.52	1289.61	640.88	166
FIFO 18	7.85	7867.24	1253.35	623.24	160
FIFO 19	7.69	7708.43	1216.43	596.87	154
FIFO 20	7.87	7888.07	1131.81	469.82	136
FIFO 21	7.95	7973.74	1242.74	598.93	158
FIFO 22	7.92	7954.55	1243.55	572.56	149
FIFO 23	7.90	7914.11	1334.30	708.02	172
FIFO 24	8.00	8014.85	1377.85	730.84	175
FIFO 25	8.01	8016.55	1449.55	747.62	181
FIFO 26	7.89	7906.17	1216.55	643.00	154
FIFO 27	7.93	7947.92	1274.92	640.32	164
FIFO 28	7.89	7903.22	1332.57	714.24	173
FIFO 29	8.05	8076.45	1406.17	749.88	182
FIFO 30	7.81	7828.82	1156.82	577.29	143

FIFO	7.90	7922.59	1449.55	636.39	182

Table F.4 - Results for FIFO at 175 Hz Arrival Rate

175	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
FIFO 1	7.92	7937.10	2250.57	1120.66	275
FIFO 2	8.00	8007.24	2294.24	1172.93	290
FIFO 3	7.88	7890.07	2061.73	1029.90	264
FIFO 4	7.97	7983.00	2239.00	1189.01	283
FIFO 5	7.82	7839.76	2127.86	1044.65	273
FIFO 6	8.04	8053.79	2358.70	1208.43	298
FIFO 7	7.84	7854.12	2068.12	1006.80	262
FIFO 8	8.01	8028.59	2303.59	1138.07	282
FIFO 9	7.92	7938.90	2183.90	1086.50	278
FIFO 10	7.95	7957.31	2209.31	1098.64	280
FIFO 11	7.95	7961.89	2331.57	1164.32	286
FIFO 12	7.76	7779.49	2119.61	1033.51	271
FIFO 13	7.95	7954.12	2259.12	1142.45	284
FIFO 14	7.77	7786.16	1974.16	939.03	251
FIFO 15	7.97	7980.75	2308.32	1134.10	293
FIFO 16	7.85	7854.63	2020.43	1035.82	264
FIFO 17	7.79	7799.54	2079.80	1030.96	264
FIFO 18	7.85	7856.24	2098.24	1046.84	268
FIFO 19	7.69	7704.43	2051.43	1009.53	265
FIFO 20	7.87	7882.07	2094.07	976.80	257
FIFO 21	7.95	7967.60	2266.60	1108.76	283
FIFO 22	7.92	7933.82	2330.82	1095.42	285
FIFO 23	7.90	7918.59	2248.63	1118.68	284
FIFO 24	8.00	8004.80	2198.80	1092.66	277
FIFO 25	8.00	8008.45	2322.45	1137.45	290
FIFO 26	7.89	7906.78	2211.78	1111.66	282
FIFO 27	7.93	7938.46	2236.12	1124.60	283
FIFO 28	7.89	7906.22	2262.22	1162.78	295
FIFO 29	8.05	8063.48	2172.29	1129.77	279
FIFO 30	7.81	7817.16	2110.16	1053.15	266
FIFO	7.90	7917.15	2358.70	1091.46	298

Table F.5 - Results for FIFO at 180 Hz Arrival Rate

180	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
FIFO 1	7.92	7937.10	2464.10	1178.32	303
FIFO 2	8.00	8011.24	2531.24	1250.67	320
FIFO 3	7.88	7884.28	2304.93	1180.75	297
FIFO 4	7.97	7982.06	2413.06	1191.66	307
FIFO 5	7.82	7831.04	2276.46	1183.08	292
FIFO 6	8.04	8061.79	2421.79	1297.04	306
FIFO 7	7.84	7853.12	2237.12	1102.73	284
FIFO 8	8.01	8027.54	2567.54	1276.61	315
FIFO 9	7.92	7935.90	2443.90	1231.82	308
FIFO 10	7.95	7953.31	2430.31	1231.52	306
FIFO 11	7.95	7958.89	2348.64	1165.03	288
FIFO 12	7.76	7772.20	2112.73	1030.90	271
FIFO 13	7.95	7957.12	2405.12	1231.57	303
FIFO 14	7.77	7786.38	2142.38	1041.88	274
FIFO 15	7.97	7976.75	2437.88	1209.82	306
FIFO 16	7.85	7866.28	2309.08	1219.09	302
FIFO 17	7.79	7795.54	2152.24	1082.36	274
FIFO 18	7.85	7858.24	2383.24	1192.02	303
FIFO 19	7.69	7706.63	2086.63	1036.42	269
FIFO 20	7.87	7881.07	2338.07	1118.01	290
FIFO 21	7.95	7962.56	2527.56	1234.17	315
FIFO 22	7.92	7932.58	2423.58	1173.54	297
FIFO 23	7.90	7909.62	2478.99	1261.42	311
FIFO 24	8.00	8017.85	2565.85	1287.59	323
FIFO 25	8.01	8016.95	2361.95	1148.92	294
FIFO 26	7.89	7904.78	2449.78	1260.28	309
FIFO 27	7.93	7938.46	2560.46	1273.53	326
FIFO 28	7.89	7897.22	2335.22	1185.23	305
FIFO 29	8.05	8061.32	2496.73	1306.48	321
FIFO 30	7.81	7822.16	2309.16	1156.87	293

FIFO	7.90	7916.67	2567.54	1191.31	326

APPENDIX G

CLOOK Simulation Average Results in Conventional Approach

Table G.1 - Results for CLOOK at 125 Hz Arrival Rate

125	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	7.69	7949.71	71.87	22.60	6
CLOOK 2	7.80	8065.21	80.08	22.99	8
CLOOK 3	7.72	8102.72	77.03	19.63	6
CLOOK 4	7.76	7970.60	78.96	24.24	9
CLOOK 5	7.77	8052.05	71.48	20.44	6
CLOOK 6	7.88	8025.58	85.75	26.50	8
CLOOK 7	7.67	8062.36	62.16	17.99	6
CLOOK 8	7.82	8094.67	74.65	21.75	7
CLOOK 9	7.83	8182.65	89.25	22.39	6
CLOOK 10	7.68	7878.80	89.54	25.13	8
CLOOK 11	7.79	8083.55	76.99	24.00	7
CLOOK 12	7.68	7972.79	74.72	19.81	6
CLOOK 13	7.76	7907.66	74.09	25.40	8
CLOOK 14	7.61	7987.01	73.97	22.36	7
CLOOK 15	7.85	8061.87	86.77	23.50	6
CLOOK 16	7.70	8017.28	80.80	22.04	7
CLOOK 17	7.69	8109.55	71.99	19.03	7
CLOOK 18	7.80	8135.58	70.08	20.61	6
CLOOK 19	7.62	7936.82	69.47	20.70	7
CLOOK 20	7.69	7974.07	83.05	22.07	7
CLOOK 21	7.84	8106.14	71.03	22.60	6
CLOOK 22	7.82	8086.01	81.25	21.81	8
CLOOK 23	7.74	8163.90	70.95	20.68	6
CLOOK 24	7.84	7960.92	70.29	23.16	7
CLOOK 25	7.75	7994.85	95.55	25.18	8
CLOOK 26	7.73	7988.58	79.43	22.09	7
CLOOK 27	7.82	8058.22	103.07	23.29	7
CLOOK 28	7.71	7880.39	87.22	24.99	8
CLOOK 29	7.84	8079.59	81.57	22.70	7
CLOOK 30	7.74	8048.00	79.72	23.35	6
CLOOK	7.75	8031.24	103.07	22.44	9

Table G.2 - Results for CLOOK at 135 Hz Arrival Rate

135	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	7.30	7314.62	105.33	36.53	9
CLOOK 2	7.34	7350.22	102.25	34.60	9
CLOOK 3	7.42	7525.65	103.00	28.48	10
CLOOK 4	7.37	7430.75	101.72	34.26	8
CLOOK 5	7.51	7596.22	92.82	28.43	9
CLOOK 6	7.35	7460.32	102.32	33.52	9
CLOOK 7	7.42	7497.88	93.07	30.16	8
CLOOK 8	7.36	7412.51	98.02	35.42	9
CLOOK 9	7.33	7343.45	108.81	36.28	10
CLOOK 10	7.30	7322.82	102.38	35.90	11
CLOOK 11	7.41	7452.28	91.07	33.18	9
CLOOK 12	7.36	7432.51	106.62	31.97	12
CLOOK 13	7.49	7541.41	112.97	34.48	9
CLOOK 14	7.22	7285.67	96.26	33.74	9
CLOOK 15	7.34	7375.17	87.58	34.16	8
CLOOK 16	7.38	7424.25	117.37	33.86	12
CLOOK 17	7.46	7532.09	82.58	29.52	9
CLOOK 18	7.38	7495.35	101.30	32.78	9
CLOOK 19	7.37	7472.44	90.77	32.10	8
CLOOK 20	7.28	7365.99	94.24	33.76	10
CLOOK 21	7.48	7575.32	94.18	29.21	8
CLOOK 22	7.28	7316.75	89.12	35.14	10
CLOOK 23	7.25	7275.74	100.12	35.39	8
CLOOK 24	7.45	7480.49	90.87	34.32	9
CLOOK 25	7.46	7530.62	86.17	30.77	8
CLOOK 26	7.39	7452.01	99.20	33.94	10
CLOOK 27	7.44	7512.22	105.12	33.32	9
CLOOK 28	7.46	7531.57	90.35	30.64	9
CLOOK 29	7.45	7532.77	105.90	32.71	9
CLOOK 30	7.42	7450.49	91.67	31.38	9
CLOOK	7.38	7442.99	117.37	33.00	12

Table G.3 - Results for CLOOK at 150 Hz Arrival Rate

150	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	6.70	6737.17	116.72	45.84	12
CLOOK 2	6.78	6796.62	134.23	44.34	13
CLOOK 3	6.75	6771.42	106.99	42.75	11
CLOOK 4	6.85	6857.20	124.77	44.72	11
CLOOK 5	6.52	6547.35	131.22	47.36	13
CLOOK 6	6.78	6809.78	113.81	44.60	11
CLOOK 7	6.70	6757.57	113.16	42.54	10
CLOOK 8	6.82	6850.13	130.15	43.12	12
CLOOK 9	6.74	6782.09	127.42	45.90	12
CLOOK 10	6.65	6686.69	120.48	45.10	11
CLOOK 11	6.76	6756.99	144.06	44.63	12
CLOOK 12	6.66	6675.31	136.14	44.01	12
CLOOK 13	6.75	6764.33	122.78	45.61	11
CLOOK 14	6.72	6766.70	113.89	45.26	13
CLOOK 15	6.75	6759.82	127.08	46.81	12
CLOOK 16	6.56	6578.18	125.97	45.27	13
CLOOK 17	6.55	6563.37	151.74	47.30	14
CLOOK 18	6.66	6686.95	150.66	48.57	14
CLOOK 19	6.55	6566.50	125.46	48.12	14
CLOOK 20	6.80	6815.15	107.75	42.83	11
CLOOK 21	6.77	6799.39	109.88	44.60	12
CLOOK 22	6.73	6775.93	119.17	45.64	10
CLOOK 23	6.60	6608.88	114.55	44.93	11
CLOOK 24	6.65	6682.49	141.56	46.49	13
CLOOK 25	6.61	6620.37	138.34	47.43	14
CLOOK 26	6.72	6741.58	114.13	44.82	13
CLOOK 27	6.70	6742.57	114.82	45.12	11
CLOOK 28	6.60	6618.28	153.59	46.16	14
CLOOK 29	6.70	6727.27	124.48	44.90	12
CLOOK 30	6.70	6733.42	143.22	45.13	13

CLOOK	6.69	6719.32	153.59	45.33	14

Table G.4 - Results for CLOOK at 175 Hz Arrival Rate

175	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	5.75	5778.96	203.39	75.97	20
CLOOK 2	5.81	5850.86	223.20	78.98	22
CLOOK 3	5.90	5915.23	161.18	66.68	16
CLOOK 4	5.79	5823.41	238.40	82.88	26
CLOOK 5	5.77	5831.32	229.01	79.02	20
CLOOK 6	5.77	5816.77	207.08	86.41	22
CLOOK 7	5.84	5892.92	177.08	68.84	19
CLOOK 8	5.78	5830.56	197.50	71.33	18
CLOOK 9	5.85	5925.70	224.94	78.18	22
CLOOK 10	5.84	5879.85	179.47	66.10	19
CLOOK 11	5.73	5737.98	241.65	85.79	22
CLOOK 12	5.73	5782.98	225.40	76.45	25
CLOOK 13	5.74	5754.76	199.92	76.20	21
CLOOK 14	5.88	5919.46	167.98	66.14	17
CLOOK 15	5.72	5745.24	221.27	81.88	21
CLOOK 16	5.88	5904.07	191.23	70.49	20
CLOOK 17	5.79	5794.26	191.33	70.68	19
CLOOK 18	5.86	5896.07	214.78	73.52	24
CLOOK 19	5.71	5726.39	225.05	82.13	21
CLOOK 20	5.84	5852.51	175.85	67.51	18
CLOOK 21	5.79	5834.54	228.10	75.73	23
CLOOK 22	5.69	5737.18	207.01	76.11	21
CLOOK 23	5.75	5777.94	201.44	76.37	21
CLOOK 24	5.88	5918.33	208.63	73.54	21
CLOOK 25	5.82	5869.31	278.18	74.25	25
CLOOK 26	5.76	5809.16	204.80	80.04	19
CLOOK 27	5.76	5791.24	176.64	68.14	18
CLOOK 28	5.70	5734.81	212.77	73.95	21
CLOOK 29	5.97	6000.67	167.90	63.79	18
CLOOK 30	5.77	5796.10	230.53	76.93	25

CLOOK	5.80	5830.95	278.18	74.80	26

Table G.5 - Results for CLOOK at 180 Hz Arrival Rate

180	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	5.59	5644.01	247.71	82.77	27
CLOOK 2	5.59	5636.67	262.47	95.42	28
CLOOK 3	5.66	5709.48	227.23	85.71	23
CLOOK 4	5.65	5679.46	230.87	84.35	23
CLOOK 5	5.62	5669.22	224.20	91.01	26
CLOOK 6	5.69	5721.72	219.31	84.34	21
CLOOK 7	5.68	5731.56	234.19	85.29	24
CLOOK 8	5.57	5618.95	218.99	84.65	23
CLOOK 9	5.59	5647.32	233.14	95.92	22
CLOOK 10	5.62	5672.82	184.40	75.86	21
CLOOK 11	5.69	5682.91	257.52	89.97	28
CLOOK 12	5.74	5781.56	195.24	75.09	20
CLOOK 13	5.62	5658.56	205.03	86.26	21
CLOOK 14	5.71	5753.87	225.25	78.36	23
CLOOK 15	5.64	5674.28	246.85	90.55	27
CLOOK 16	5.59	5624.53	286.94	90.71	28
CLOOK 17	5.74	5750.60	200.16	81.11	22
CLOOK 18	5.59	5627.28	266.70	85.87	25
CLOOK 19	5.69	5719.90	232.30	87.22	24
CLOOK 20	5.59	5594.75	243.00	75.57	22
CLOOK 21	5.53	5570.60	239.06	90.40	26
CLOOK 22	5.59	5641.79	317.18	96.11	33
CLOOK 23	5.54	5579.88	244.88	84.00	25
CLOOK 24	5.55	5618.32	222.86	95.53	25
CLOOK 25	5.76	5792.68	231.19	80.39	23
CLOOK 26	5.53	5570.44	250.27	99.89	26
CLOOK 27	5.49	5535.94	269.47	94.34	27
CLOOK 28	5.65	5685.51	211.33	77.75	21
CLOOK 29	5.64	5672.61	202.82	73.94	20
CLOOK 30	5.61	5652.87	242.15	83.11	24

CLOOK	5.62	5664.00	317.18	86.05	33

APPENDIX H

SDM-E Simulation Average Results in Conventional Approach

Table H.1 - Results for SDM-E at 125 Hz Arrival Rate

125	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-E 1	7.67	7940.75	56.73	19.58	5
SDM-E 2	7.78	8062.99	78.82	19.44	7
SDM-E 3	7.70	8092.36	55.11	17.78	5
SDM-E 4	7.68	7962.31	66.67	19.47	8
SDM-E 5	7.71	8045.65	59.37	18.38	5
SDM-E 6	7.79	8009.14	63.86	19.90	6
SDM-E 7	7.66	8043.75	71.10	17.61	6
SDM-E 8	7.80	8096.83	65.40	18.89	7
SDM-E 9	7.74	8168.70	95.03	18.49	6
SDM-E 10	7.58	7878.80	63.75	20.00	6
SDM-E 11	7.71	8083.55	60.64	19.25	5
SDM-E 12	7.62	7972.79	49.43	17.74	5
SDM-E 13	7.70	7907.66	74.44	19.77	6
SDM-E 14	7.53	7982.41	75.35	18.45	6
SDM-E 15	7.78	8061.87	51.38	18.68	5
SDM-E 16	7.64	8017.28	53.31	18.12	6
SDM-E 17	7.65	8109.55	63.27	17.15	7
SDM-E 18	7.76	8135.58	55.95	17.52	5
SDM-E 19	7.58	7936.82	53.49	17.38	5
SDM-E 20	7.63	7959.43	52.79	18.51	6
SDM-E 21	7.77	8106.14	67.85	19.20	6
SDM-E 22	7.74	8086.01	67.52	19.05	6
SDM-E 23	7.71	8163.90	60.51	18.89	5
SDM-E 24	7.83	7960.92	49.12	19.43	6
SDM-E 25	7.71	7994.85	74.82	21.16	6
SDM-E 26	7.70	7986.47	75.22	19.59	6
SDM-E 27	7.71	8058.22	75.98	19.39	6
SDM-E 28	7.62	7872.66	74.37	19.17	7
SDM-E 29	7.79	8079.59	62.20	18.72	6
SDM-E 30	7.65	8044.88	59.95	17.88	6
SDM-E	7.70	8027.40	95.03	18.82	8

Table H.2 - Results for SDM-E at 135 Hz Arrival Rate

135	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-E 1	7.26	7318.08	71.48	25.00	7
SDM-E 2	7.30	7337.44	103.73	25.21	7
SDM-E 3	7.36	7520.38	74.95	22.44	7
SDM-E 4	7.34	7428.61	77.43	24.63	6
SDM-E 5	7.44	7580.12	88.08	22.70	7
SDM-E 6	7.32	7448.36	80.15	24.17	7
SDM-E 7	7.38	7469.13	62.23	23.16	7
SDM-E 8	7.34	7380.60	85.04	25.83	8
SDM-E 9	7.23	7319.78	93.83	24.10	8
SDM-E 10	7.25	7314.75	98.72	25.31	7
SDM-E 11	7.32	7427.07	77.29	23.93	7
SDM-E 12	7.30	7405.84	69.60	23.36	7
SDM-E 13	7.43	7529.70	54.94	22.95	6
SDM-E 14	7.19	7279.05	79.39	23.60	7
SDM-E 15	7.32	7374.26	74.86	25.07	7
SDM-E 16	7.35	7412.22	79.58	23.91	7
SDM-E 17	7.39	7529.58	60.85	23.07	6
SDM-E 18	7.33	7495.35	72.51	22.68	6
SDM-E 19	7.29	7443.04	80.79	22.67	6
SDM-E 20	7.27	7352.95	66.18	24.75	6
SDM-E 21	7.47	7575.32	105.06	23.18	7
SDM-E 22	7.24	7304.56	71.60	24.95	6
SDM-E 23	7.20	7249.63	107.88	26.06	7
SDM-E 24	7.42	7478.01	79.95	25.26	6
SDM-E 25	7.41	7530.62	74.06	23.71	7
SDM-E 26	7.34	7443.20	83.11	24.95	6
SDM-E 27	7.41	7512.22	90.69	24.55	6
SDM-E 28	7.41	7527.29	74.09	22.63	7
SDM-E 29	7.42	7532.77	84.47	24.32	7
SDM-E 30	7.32	7441.77	62.24	22.28	6

SDM-E	7.34	7432.06	107.88	24.01	8

Table H.3 - Results for SDM-E at 150 Hz Arrival Rate

150	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-E 1	6.66	6692.94	74.39	28.69	8
SDM-E 2	6.74	6783.68	113.72	28.97	10
SDM-E 3	6.70	6724.26	78.58	27.53	8
SDM-E 4	6.81	6849.56	102.87	28.01	8
SDM-E 5	6.51	6539.49	84.64	28.08	8
SDM-E 6	6.75	6788.46	111.95	28.58	8
SDM-E 7	6.68	6704.74	79.11	27.56	7
SDM-E 8	6.78	6793.71	147.96	29.41	8
SDM-E 9	6.73	6756.91	83.60	28.69	7
SDM-E 10	6.62	6651.88	79.87	28.98	8
SDM-E 11	6.71	6718.75	84.83	28.57	7
SDM-E 12	6.64	6663.94	94.54	28.29	8
SDM-E 13	6.74	6762.52	108.57	28.83	8
SDM-E 14	6.69	6740.74	88.68	27.67	7
SDM-E 15	6.73	6750.57	92.10	28.40	7
SDM-E 16	6.56	6566.81	85.02	29.03	7
SDM-E 17	6.53	6551.74	93.00	29.03	9
SDM-E 18	6.62	6638.68	131.70	29.44	9
SDM-E 19	6.50	6521.21	114.53	28.40	8
SDM-E 20	6.74	6784.59	79.92	27.46	8
SDM-E 21	6.74	6765.00	95.04	28.64	8
SDM-E 22	6.71	6746.88	75.14	28.08	7
SDM-E 23	6.59	6609.12	95.45	28.82	7
SDM-E 24	6.66	6677.08	78.74	28.99	7
SDM-E 25	6.59	6596.73	101.58	30.51	9
SDM-E 26	6.69	6715.50	85.85	27.96	8
SDM-E 27	6.68	6710.61	84.49	27.97	7
SDM-E 28	6.58	6602.80	136.01	28.66	8
SDM-E 29	6.68	6704.83	164.27	29.83	8
SDM-E 30	6.66	6706.22	109.34	27.12	7
SDM-E	6.67	6694.00	164.27	28.54	10

Table H.4 - Results for SDM-E at 175 Hz Arrival Rate

175	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-E 1	5.71	5729.63	130.80	33.70	10
SDM-E 2	5.74	5751.32	151.24	33.34	9
SDM-E 3	5.86	5868.47	126.55	32.00	10
SDM-E 4	5.77	5783.18	230.82	32.01	10
SDM-E 5	5.72	5740.73	111.72	32.66	11
SDM-E 6	5.73	5737.19	104.23	32.76	10
SDM-E 7	5.81	5824.29	112.86	32.50	10
SDM-E 8	5.74	5760.28	106.70	32.54	10
SDM-E 9	5.78	5803.37	126.42	33.31	10
SDM-E 10	5.78	5792.34	119.32	32.82	10
SDM-E 11	5.65	5658.48	113.25	32.97	12
SDM-E 12	5.68	5697.96	191.12	33.14	10
SDM-E 13	5.72	5724.23	151.70	34.19	10
SDM-E 14	5.83	5852.42	134.16	32.73	10
SDM-E 15	5.69	5704.91	131.72	33.57	11
SDM-E 16	5.86	5868.22	94.20	32.41	9
SDM-E 17	5.75	5753.90	128.39	33.26	11
SDM-E 18	5.78	5790.74	141.29	31.68	10
SDM-E 19	5.67	5683.61	103.53	31.76	9
SDM-E 20	5.81	5825.12	99.13	32.37	10
SDM-E 21	5.72	5735.32	134.30	33.88	10
SDM-E 22	5.63	5640.17	131.47	34.08	11
SDM-E 23	5.68	5697.31	104.90	34.06	11
SDM-E 24	5.83	5838.11	145.11	32.93	10
SDM-E 25	5.71	5717.52	130.86	33.83	11
SDM-E 26	5.72	5736.44	113.66	33.05	9
SDM-E 27	5.72	5723.88	150.19	33.77	11
SDM-E 28	5.66	5680.95	138.85	33.62	11
SDM-E 29	5.92	5931.00	112.43	33.64	9
SDM-E 30	5.73	5734.75	140.58	32.34	12
SDM-E	5.75	5759.53	230.82	33.03	12

Table H.5 - Results for SDM-E at 180 Hz Arrival Rate

180	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-E 1	5.50	5513.22	130.51	34.16	11
SDM-E 2	5.50	5517.68	158.02	35.09	11
SDM-E 3	5.61	5616.69	134.55	34.02	11
SDM-E 4	5.60	5609.01	137.12	33.74	9
SDM-E 5	5.58	5592.71	145.65	33.15	10
SDM-E 6	5.65	5672.91	140.46	33.89	12
SDM-E 7	5.63	5645.76	130.02	33.19	10
SDM-E 8	5.48	5498.32	143.02	34.85	11
SDM-E 9	5.51	5530.52	145.91	35.43	14
SDM-E 10	5.56	5567.13	133.52	34.45	12
SDM-E 11	5.63	5640.94	162.07	34.44	11
SDM-E 12	5.67	5685.88	109.40	32.39	10
SDM-E 13	5.58	5587.55	113.72	33.95	10
SDM-E 14	5.67	5687.54	122.56	32.98	10
SDM-E 15	5.57	5578.80	164.53	34.48	13
SDM-E 16	5.58	5592.53	139.79	34.11	12
SDM-E 17	5.66	5667.32	117.31	32.63	10
SDM-E 18	5.50	5508.68	202.05	35.32	12
SDM-E 19	5.63	5651.65	161.99	33.01	10
SDM-E 20	5.56	5572.79	141.36	35.43	14
SDM-E 21	5.46	5473.75	148.83	35.28	11
SDM-E 22	5.54	5552.88	169.44	34.55	11
SDM-E 23	5.45	5456.50	125.74	35.17	11
SDM-E 24	5.47	5487.05	137.55	34.79	13
SDM-E 25	5.68	5691.39	161.69	33.50	11
SDM-E 26	5.47	5481.69	159.13	34.89	11
SDM-E 27	5.41	5414.15	106.42	35.70	10
SDM-E 28	5.59	5598.76	132.10	34.09	10
SDM-E 29	5.59	5602.66	149.32	33.55	11
SDM-E 30	5.54	5549.11	137.81	34.37	11
SDM-E	5.56	5574.85	202.05	34.22	14

APPENDIX I

CLOOK Simulation Average Results in DE Approach

Table I.1 - Results for CLOOK at 125 Hz Arrival Rate

125	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	7.72	7960.98	72.29	22.49	6
CLOOK 2	7.82	8065.21	76.44	24.07	8
CLOOK 3	7.74	8102.72	77.03	20.07	6
CLOOK 4	7.72	7962.31	76.25	22.23	9
CLOOK 5	7.78	8051.81	80.26	21.30	6
CLOOK 6	7.88	8025.95	91.23	26.38	8
CLOOK 7	7.71	8062.36	63.16	19.50	6
CLOOK 8	7.85	8094.67	70.77	22.03	7
CLOOK 9	7.85	8182.65	72.28	22.68	6
CLOOK 10	7.68	7878.80	89.54	24.68	8
CLOOK 11	7.77	8083.55	74.92	23.09	6
CLOOK 12	7.68	7972.79	73.00	20.50	6
CLOOK 13	7.79	7907.66	112.72	26.27	10
CLOOK 14	7.67	7987.01	82.56	25.67	8
CLOOK 15	7.87	8061.87	77.27	24.32	6
CLOOK 16	7.70	8017.28	66.99	21.81	7
CLOOK 17	7.72	8109.55	64.76	19.14	7
CLOOK 18	7.87	8135.58	67.05	23.04	6
CLOOK 19	7.61	7936.82	69.47	20.06	7
CLOOK 20	7.68	7969.82	83.71	22.85	7
CLOOK 21	7.83	8106.14	87.92	22.14	7
CLOOK 22	7.78	8086.01	97.08	21.08	7
CLOOK 23	7.75	8163.90	70.95	20.68	7
CLOOK 24	7.87	7975.15	77.93	22.89	7
CLOOK 25	7.76	7994.85	90.32	25.90	8
CLOOK 26	7.76	7988.58	82.86	24.52	9
CLOOK 27	7.87	8058.22	103.07	25.20	7
CLOOK 28	7.69	7880.39	85.20	24.07	7
CLOOK 29	7.85	8079.59	94.25	23.43	7
CLOOK 30	7.73	8049.71	79.72	22.46	7
CLOOK	7.77	8031.73	112.72	22.82	10

Table I.2 - Results for CLOOK at 135 Hz Arrival Rate

135	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	7.31	7325.76	90.78	37.90	9
CLOOK 2	7.32	7343.69	101.12	34.17	10
CLOOK 3	7.42	7527.21	121.00	31.38	11
CLOOK 4	7.40	7444.41	120.34	37.74	10
CLOOK 5	7.53	7598.72	92.82	29.58	9
CLOOK 6	7.36	7452.74	98.96	34.69	10
CLOOK 7	7.43	7487.57	107.12	32.49	9
CLOOK 8	7.35	7412.73	97.53	35.83	9
CLOOK 9	7.31	7337.12	95.70	37.00	9
CLOOK 10	7.29	7320.75	93.11	37.25	10
CLOOK 11	7.40	7451.51	104.23	34.47	10
CLOOK 12	7.36	7424.56	110.67	33.40	12
CLOOK 13	7.48	7535.93	99.36	35.30	10
CLOOK 14	7.22	7288.20	97.29	34.53	10
CLOOK 15	7.36	7375.33	101.42	35.83	10
CLOOK 16	7.37	7424.43	119.70	35.37	12
CLOOK 17	7.46	7532.09	95.07	32.53	9
CLOOK 18	7.41	7496.31	101.30	33.69	9
CLOOK 19	7.40	7473.98	90.38	32.26	8
CLOOK 20	7.30	7369.21	93.46	34.57	10
CLOOK 21	7.49	7575.32	94.15	30.01	8
CLOOK 22	7.29	7321.45	101.29	38.98	10
CLOOK 23	7.23	7253.45	100.26	35.74	8
CLOOK 24	7.47	7493.40	89.23	35.07	9
CLOOK 25	7.46	7530.62	90.46	33.72	9
CLOOK 26	7.40	7452.41	106.49	35.11	11
CLOOK 27	7.45	7512.22	102.70	34.39	9
CLOOK 28	7.46	7527.29	87.82	32.63	9
CLOOK 29	7.47	7532.77	99.33	34.22	9
CLOOK 30	7.37	7447.28	83.50	31.59	8

CLOOK	7.39	7442.28	121.00	34.38	12

Table I.3 - Results for CLOOK at 150 Hz Arrival Rate

150	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	6.71	6759.13	138.68	45.38	13
CLOOK 2	6.79	6805.19	126.82	46.01	12
CLOOK 3	6.74	6772.62	101.35	41.59	11
CLOOK 4	6.86	6872.36	124.08	45.26	12
CLOOK 5	6.53	6547.50	131.23	50.45	13
CLOOK 6	6.78	6810.38	116.12	46.43	13
CLOOK 7	6.72	6767.53	112.56	43.86	12
CLOOK 8	6.83	6857.51	131.62	44.56	12
CLOOK 9	6.76	6793.29	113.26	47.17	12
CLOOK 10	6.65	6690.04	130.15	45.97	12
CLOOK 11	6.75	6751.90	111.99	46.05	12
CLOOK 12	6.66	6685.04	150.72	45.51	13
CLOOK 13	6.76	6771.73	132.83	48.56	11
CLOOK 14	6.72	6767.67	122.22	47.18	13
CLOOK 15	6.75	6760.25	129.45	47.79	13
CLOOK 16	6.56	6578.15	136.26	50.12	13
CLOOK 17	6.55	6566.56	129.35	49.55	13
CLOOK 18	6.66	6696.01	130.10	47.45	14
CLOOK 19	6.53	6554.58	148.17	49.03	14
CLOOK 20	6.79	6808.49	118.19	44.69	12
CLOOK 21	6.76	6797.66	126.51	43.78	12
CLOOK 22	6.73	6776.96	114.66	45.79	11
CLOOK 23	6.61	6622.47	122.16	47.13	11
CLOOK 24	6.66	6696.73	145.44	48.13	14
CLOOK 25	6.62	6632.70	129.78	49.45	14
CLOOK 26	6.72	6730.10	131.40	47.26	13
CLOOK 27	6.70	6741.25	124.74	47.80	13
CLOOK 28	6.60	6616.75	145.32	48.85	12
CLOOK 29	6.71	6736.85	125.78	46.72	13
CLOOK 30	6.71	6735.74	125.36	46.33	12

CLOOK	6.70	6723.44	150.72	46.80	14

Table I.4 - Results for CLOOK at 175 Hz Arrival Rate

175	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	5.75	5778.71	179.73	72.18	18
CLOOK 2	5.80	5836.31	216.32	81.02	22
CLOOK 3	5.88	5927.62	192.84	69.46	18
CLOOK 4	5.79	5821.14	231.04	78.42	22
CLOOK 5	5.78	5833.02	224.18	80.25	21
CLOOK 6	5.75	5820.36	194.27	81.93	20
CLOOK 7	5.85	5915.19	220.61	72.53	20
CLOOK 8	5.78	5834.47	186.32	73.89	20
CLOOK 9	5.85	5935.50	224.21	73.82	23
CLOOK 10	5.85	5901.41	193.37	69.54	20
CLOOK 11	5.71	5733.98	237.48	86.06	24
CLOOK 12	5.73	5780.31	240.95	77.49	23
CLOOK 13	5.73	5748.61	188.99	72.52	20
CLOOK 14	5.88	5920.36	169.37	66.84	18
CLOOK 15	5.71	5752.35	215.09	80.49	22
CLOOK 16	5.87	5922.11	186.80	73.05	20
CLOOK 17	5.79	5804.53	222.87	70.55	18
CLOOK 18	5.89	5909.62	253.58	76.39	25
CLOOK 19	5.71	5732.91	191.92	80.33	21
CLOOK 20	5.84	5868.53	191.11	68.33	19
CLOOK 21	5.79	5847.01	207.90	75.11	21
CLOOK 22	5.70	5743.13	220.80	78.58	22
CLOOK 23	5.72	5755.27	202.66	75.86	21
CLOOK 24	5.87	5919.71	186.70	72.00	19
CLOOK 25	5.83	5869.35	270.50	77.86	25
CLOOK 26	5.76	5820.97	227.36	81.83	22
CLOOK 27	5.76	5801.55	176.64	71.00	20
CLOOK 28	5.69	5733.21	220.12	77.54	19
CLOOK 29	5.97	5996.89	179.79	65.39	19
CLOOK 30	5.77	5798.83	230.53	80.57	25

CLOOK	5.79	5835.43	270.50	75.36	25

Table I.5 - Results for CLOOK at 180 Hz Arrival Rate

180	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
CLOOK 1	5.58	5651.15	253.71	85.16	28
CLOOK 2	5.59	5620.18	260.31	92.35	28
CLOOK 3	5.68	5733.74	213.78	82.17	23
CLOOK 4	5.64	5689.63	227.22	82.82	23
CLOOK 5	5.62	5662.10	227.69	80.79	21
CLOOK 6	5.70	5721.13	222.93	84.39	23
CLOOK 7	5.70	5779.76	267.65	88.19	26
CLOOK 8	5.57	5628.50	236.09	85.57	23
CLOOK 9	5.59	5651.45	234.41	94.65	23
CLOOK 10	5.62	5676.87	201.24	80.63	20
CLOOK 11	5.68	5698.34	275.30	90.94	26
CLOOK 12	5.73	5790.62	217.34	78.95	22
CLOOK 13	5.61	5658.92	210.98	85.58	22
CLOOK 14	5.72	5751.97	205.26	77.79	21
CLOOK 15	5.63	5675.88	246.12	86.26	26
CLOOK 16	5.60	5631.05	255.15	95.19	28
CLOOK 17	5.72	5729.85	214.87	77.99	22
CLOOK 18	5.59	5626.11	254.99	86.06	24
CLOOK 19	5.69	5696.49	220.14	78.46	22
CLOOK 20	5.59	5594.74	260.14	80.93	26
CLOOK 21	5.56	5614.68	265.53	89.00	26
CLOOK 22	5.59	5657.80	325.66	98.57	33
CLOOK 23	5.54	5579.14	275.84	90.14	26
CLOOK 24	5.55	5597.54	268.57	95.63	28
CLOOK 25	5.74	5781.24	214.81	80.80	23
CLOOK 26	5.52	5569.96	250.27	98.07	26
CLOOK 27	5.49	5538.60	264.71	96.31	26
CLOOK 28	5.64	5684.51	226.80	83.20	22
CLOOK 29	5.64	5667.21	200.51	76.62	21
CLOOK 30	5.62	5651.39	202.44	83.51	23
CLOOK	5.63	5667.02	325.66	86.22	33

APPENDIX J

SDM-NN Simulation Average Results in DE Approach

Table J.1 - Results for SDM-NN at 125 Hz Arrival Rate

125	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-NN 1	7.66	7940.58	67.44	19.30	6
SDM-NN 2	7.80	8068.44	69.55	19.18	6
SDM-NN 3	7.73	8096.05	61.87	18.48	5
SDM-NN 4	7.68	7962.31	73.98	19.49	7
SDM-NN 5	7.72	8043.12	65.02	19.11	5
SDM-NN 6	7.79	8008.26	124.27	20.80	7
SDM-NN 7	7.70	8060.32	105.11	18.02	6
SDM-NN 8	7.83	8097.37	58.49	19.14	6
SDM-NN 9	7.76	8171.91	61.93	18.61	6
SDM-NN 10	7.60	7878.80	79.55	20.16	7
SDM-NN 11	7.73	8083.55	85.13	20.00	5
SDM-NN 12	7.62	7972.79	65.69	18.03	6
SDM-NN 13	7.71	7907.66	73.05	19.69	6
SDM-NN 14	7.55	7988.07	93.09	18.88	6
SDM-NN 15	7.78	8061.87	67.20	18.89	5
SDM-NN 16	7.65	8017.28	77.82	18.28	5
SDM-NN 17	7.64	8109.55	58.79	17.18	7
SDM-NN 18	7.74	8135.58	71.87	17.94	5
SDM-NN 19	7.57	7936.82	56.36	17.31	6
SDM-NN 20	7.64	7959.60	79.05	19.17	6
SDM-NN 21	7.79	8106.14	66.59	20.41	6
SDM-NN 22	7.74	8086.01	61.76	18.64	6
SDM-NN 23	7.72	8163.90	75.50	18.51	6
SDM-NN 24	7.83	7958.24	73.04	19.64	5
SDM-NN 25	7.71	7994.85	112.32	20.73	6
SDM-NN 26	7.70	7986.47	72.76	19.32	5
SDM-NN 27	7.74	8058.22	85.14	19.83	6
SDM-NN 28	7.63	7883.14	87.14	19.43	7
SDM-NN 29	7.80	8079.59	77.83	19.03	6
SDM-NN 30	7.69	8044.88	63.92	18.67	6
SDM-NN	7.71	8028.71	124.27	19.06	7

Table J.2 - Results for SDM-NN at 135 Hz Arrival Rate

135	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-NN 1	7.27	7318.21	99.34	25.10	7
SDM-NN 2	7.30	7333.18	93.49	24.87	7
SDM-NN 3	7.38	7523.28	114.09	22.66	7
SDM-NN 4	7.35	7425.76	102.26	25.54	6
SDM-NN 5	7.46	7580.80	72.36	22.29	6
SDM-NN 6	7.34	7457.46	112.37	26.04	7
SDM-NN 7	7.40	7483.81	87.36	23.58	7
SDM-NN 8	7.34	7410.05	99.33	27.51	8
SDM-NN 9	7.25	7317.93	111.33	24.49	8
SDM-NN 10	7.27	7325.31	100.58	26.10	7
SDM-NN 11	7.32	7427.07	100.93	24.93	6
SDM-NN 12	7.31	7408.82	117.76	23.95	8
SDM-NN 13	7.45	7537.41	80.61	23.99	7
SDM-NN 14	7.18	7280.70	101.00	24.26	7
SDM-NN 15	7.34	7368.35	148.52	25.18	7
SDM-NN 16	7.37	7416.59	77.77	24.93	6
SDM-NN 17	7.41	7532.09	74.22	23.00	7
SDM-NN 18	7.35	7495.35	85.21	23.79	7
SDM-NN 19	7.31	7443.60	108.02	23.34	7
SDM-NN 20	7.29	7354.67	81.03	25.74	6
SDM-NN 21	7.47	7575.32	108.62	23.98	7
SDM-NN 22	7.25	7304.40	76.59	25.28	6
SDM-NN 23	7.21	7256.94	87.54	27.19	7
SDM-NN 24	7.44	7482.29	74.91	26.18	6
SDM-NN 25	7.45	7530.62	95.58	24.62	7
SDM-NN 26	7.35	7441.95	81.29	25.74	6
SDM-NN 27	7.41	7512.22	74.15	24.45	6
SDM-NN 28	7.41	7527.29	87.14	23.11	6
SDM-NN 29	7.42	7532.77	94.36	24.66	7
SDM-NN 30	7.33	7448.46	87.37	23.46	7

SDM-NN	7.35	7435.09	148.52	24.67	8

Table J.3 - Results for SDM-NN at 150 Hz Arrival Rate

150	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-NN 1	6.66	6693.07	127.44	30.00	8
SDM-NN 2	6.76	6788.57	114.95	29.81	8
SDM-NN 3	6.71	6733.34	106.86	28.08	8
SDM-NN 4	6.80	6842.86	101.67	29.84	8
SDM-NN 5	6.52	6534.36	181.63	29.06	8
SDM-NN 6	6.74	6785.62	107.54	28.95	7
SDM-NN 7	6.68	6721.28	139.74	30.06	8
SDM-NN 8	6.77	6804.87	117.25	29.71	8
SDM-NN 9	6.73	6768.97	104.19	29.86	7
SDM-NN 10	6.63	6668.65	127.25	29.37	8
SDM-NN 11	6.71	6719.78	119.27	29.97	8
SDM-NN 12	6.65	6676.07	91.91	28.43	8
SDM-NN 13	6.75	6765.59	144.63	30.49	8
SDM-NN 14	6.70	6748.70	94.80	29.08	8
SDM-NN 15	6.72	6742.55	110.61	29.56	8
SDM-NN 16	6.56	6565.41	102.54	30.22	7
SDM-NN 17	6.52	6549.55	132.94	29.44	8
SDM-NN 18	6.62	6641.58	101.50	29.90	8
SDM-NN 19	6.50	6520.71	122.32	29.24	9
SDM-NN 20	6.75	6797.91	109.08	28.49	8
SDM-NN 21	6.75	6774.36	106.84	29.82	8
SDM-NN 22	6.71	6756.23	115.83	29.68	7
SDM-NN 23	6.59	6604.68	113.89	29.17	7
SDM-NN 24	6.65	6669.68	157.21	31.04	8
SDM-NN 25	6.60	6611.80	138.24	30.52	8
SDM-NN 26	6.71	6727.12	106.82	29.72	7
SDM-NN 27	6.69	6710.28	118.96	30.07	7
SDM-NN 28	6.59	6610.19	106.38	30.14	8
SDM-NN 29	6.68	6703.11	110.39	30.07	8
SDM-NN 30	6.65	6701.55	88.73	28.70	8
SDM-NN	6.67	6697.95	181.63	29.62	9

Table J.4 - Results for SDM-NN at 175 Hz Arrival Rate

175	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-NN 1	5.71	5739.24	215.51	36.03	12
SDM-NN 2	5.74	5759.56	255.35	34.49	10
SDM-NN 3	5.86	5883.13	140.32	33.35	10
SDM-NN 4	5.78	5792.14	166.36	34.20	10
SDM-NN 5	5.72	5753.17	174.58	33.78	10
SDM-NN 6	5.73	5742.49	159.20	34.17	10
SDM-NN 7	5.81	5846.78	248.12	33.85	11
SDM-NN 8	5.74	5778.46	199.00	34.63	10
SDM-NN 9	5.80	5836.30	166.38	35.51	10
SDM-NN 10	5.78	5817.61	199.55	33.93	9
SDM-NN 11	5.65	5671.03	259.68	35.72	11
SDM-NN 12	5.68	5712.47	158.04	34.14	10
SDM-NN 13	5.73	5736.11	175.10	35.49	10
SDM-NN 14	5.84	5871.62	208.85	33.60	9
SDM-NN 15	5.71	5711.22	160.51	33.85	10
SDM-NN 16	5.87	5877.55	158.58	33.08	10
SDM-NN 17	5.74	5754.51	145.45	33.41	9
SDM-NN 18	5.77	5792.77	113.26	32.58	10
SDM-NN 19	5.68	5688.65	186.57	34.13	10
SDM-NN 20	5.82	5834.44	156.71	33.94	9
SDM-NN 21	5.73	5743.02	119.64	34.32	9
SDM-NN 22	5.64	5665.09	196.54	34.33	10
SDM-NN 23	5.69	5716.29	202.13	35.43	11
SDM-NN 24	5.83	5855.80	178.43	34.37	10
SDM-NN 25	5.72	5734.50	181.42	34.73	10
SDM-NN 26	5.73	5754.91	177.83	34.10	10
SDM-NN 27	5.73	5752.29	199.82	34.40	10
SDM-NN 28	5.67	5696.34	202.86	35.01	10
SDM-NN 29	5.92	5943.09	169.37	33.09	9
SDM-NN 30	5.73	5754.62	222.21	33.68	10

SDM-NN	5.75	5773.84	259.68	34.24	12

Table J.5 - Results for SDM-NN at 180 Hz Arrival Rate

180	Average Service Rate	C_{max}	F_{max}	F_{avg}	Max. # of Jobs Waiting (Q+RQ)
SDM-NN 1	5.52	5534.46	199.47	36.35	10
SDM-NN 2	5.50	5526.24	227.30	36.86	12
SDM-NN 3	5.61	5630.57	164.66	36.42	12
SDM-NN 4	5.61	5638.64	159.31	36.29	10
SDM-NN 5	5.58	5597.18	175.89	35.54	11
SDM-NN 6	5.65	5672.44	186.46	35.74	12
SDM-NN 7	5.64	5660.86	163.79	34.25	10
SDM-NN 8	5.50	5521.53	211.28	35.10	11
SDM-NN 9	5.52	5557.95	211.28	36.90	13
SDM-NN 10	5.56	5598.97	151.56	36.40	12
SDM-NN 11	5.64	5644.16	227.58	35.63	11
SDM-NN 12	5.68	5687.95	170.21	35.13	10
SDM-NN 13	5.59	5598.74	153.18	34.98	9
SDM-NN 14	5.68	5699.00	156.08	35.16	10
SDM-NN 15	5.57	5597.24	357.95	36.04	11
SDM-NN 16	5.59	5608.58	176.56	34.80	11
SDM-NN 17	5.67	5674.15	174.12	33.71	10
SDM-NN 18	5.50	5518.68	215.14	36.55	11
SDM-NN 19	5.65	5664.01	136.02	35.50	10
SDM-NN 20	5.57	5575.98	212.49	35.65	11
SDM-NN 21	5.47	5483.80	163.88	36.04	11
SDM-NN 22	5.55	5590.23	166.27	35.39	11
SDM-NN 23	5.46	5479.03	171.24	37.04	11
SDM-NN 24	5.47	5506.11	182.52	36.28	13
SDM-NN 25	5.68	5708.49	162.66	35.48	10
SDM-NN 26	5.48	5495.77	223.25	35.24	10
SDM-NN 27	5.40	5427.78	207.32	38.48	12
SDM-NN 28	5.60	5625.08	206.48	34.37	10
SDM-NN 29	5.60	5617.24	287.87	35.95	11
SDM-NN 30	5.55	5571.96	143.15	35.38	12

SDM-NN	5.57	5590.43	357.95	35.76	13

APPENDIX K

Disk Scheduling with Double Queue

In double queue scheduling, there are two importance classes, of queues, namely, High Importance Queue (HIQ) and Low Importance Queue (LIQ). Coming jobs are classified according to their weights (attached to job identity). While the jobs having higher weights than a threshold value are classified as high importance class jobs (HIC), others are considered as low importance class (LIC). The sequencing policy determines the processing order of the HIC and LIC jobs. The general policy is such that after all the jobs in HIQ are finished, LIC jobs can be served. The maximum number of jobs in each queue is 128 in a certain moment of time (i.e. number of jobs in HIQ + number of jobs in LIQ + 2 jobs in run queue + 1 job in service = N_{system} <=259).

Apart from this queue difference, the system works similarly as that of a typical hard disk with single queue. If HIQ is not empty and there is no timeout, the algorithm sorts all the jobs in HIQ and determines the next second job in run queue (RQ). Each time the processing of a job is finished, the first job in RQ is taken into process and the second job in RQ becomes the 1st one.

As in the case of typical hard disks, the job starvation is an important problem. Another difficulty arises from the predominance of the HIC jobs over the LIC jobs that bring the higher susceptibility of the jobs in LIQ to starvation. If HIQ does

not become empty for a long time, the LIC jobs will stay undone for that long time and may be some of them will never be done. This could be the case for some HIC jobs as well. To overcome this, a fixed timeout (probably different for the HIC and LIC jobs) must be decided for every job. The system puts a label on every job, whether it is HIC or LIC, indicating its arrival time (rh_i) or (rl_i). If it is still waiting to be serviced after a defined time stamp (τ), it is assessed timeout ($rh_i + \tau_{HIC}$) or ($rl_i + \tau_{LIC}$). That brings the time windows concept into our problem.

A clever strategy necessitates defining addresses on the disk space with the aid of partitions over the disk. When this is done, the service time for a certain job is said to be dependent on (a) the size of the coming job, (b) the last served job's address, and (c) the coming job's address (Since a job is not a single point over the disk, but a continuous line with starting and end points, the mean value between those points can be taken as that job's address as a convention).

Decision: Implement the algorithm or not

If the number of newcomers is below a defined "break even" number, then it will be better not to implement the algorithm (re-sequencing newcomers together with standing jobs). In that case the newcomers would be taken into proper queue according to some basic policy, i.e. FIFO. Or as a better way, some heuristics could be utilized to insert the newcomers into the standing sequence.

The two main objectives of this problem are, (1) Minimizing the average response time of the system to the jobs in HIQ, (2) Maximizing the throughput of the jobs in LIQ (i.e. number/s). The

timeout cases could negatively affect the system's performance and it could not be avoided by utilizing a constraint because of the disk structure. A contributive objective would be "minimizing the number of timeouts (i.e. number/second)". The main constraint of the problem is the limits of HIQ and LIQ (128 each).

These two objectives are contradicting actually. Especially in an environment of heavy workload, the LIQ jobs' throughput objective will seriously be jeopardized.