EFFICIENT DYNAMIC COST SCHEDULING ALGORITHM

FOR DATA BATCH PROCESSING

by

Alia Al Sadawi

A Thesis Presented to the Faculty of the
American University of Sharjah
College of Engineering
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Engineering Systems Management

Sharjah, United Arab Emirates

May 2016

Approval Signatures

We, the undersigned, approve the Master's Thesis of Alia Al Sadawi.

Thesis Title: Efficient Dynamic Cost Scheduling Algorithm for Data Batch Process

Signature	**Date of Signature** (dd/mm/yyyy)
___________________________ Dr. Abdulrahim Shamayleh, Assistant Professor, Department of Industrial Engineering, Thesis Advisor	_______________
___________________________ Dr. Malick Ndiaye, Associate Professor, Department of Industrial Engineering, Thesis Co-Advisor	_______________
___________________________ Dr. Mahmoud Awad, Assistant Professor, Department of Industrial Engineering, Thesis Committee Member	_______________
_________________________ Dr. Norita Ahmad, Associate Professor, Department of Marketing and Information Systems, School of Business Administration, Thesis Committee Member	_______________
___________________________ Dr. Moncer Hariga, Director, Engineering Systems Management Graduate Program	_______________
___________________________ Dr. Mohamed El-Tarhuni, Associate Dean, College of Engineering	_______________
___________________________ Dr. Leland Blank, Dean, College of Engineering	_______________
__________________________ Dr. Khaled Assaleh, Interim Vice Provost for Research and Graduate Studies	_______________

Acknowledgements

First and above all I am grateful to Allah for granting me the courage and patience to finish this thesis.

I would like to thank the American University of Sharjah for providing such high standards educational programs. Special thanks goes to the faculty members of the College of Engineering, the Engineering Systems Management (ESM) Graduate Program, and the Department of Industrial Engineering for their care and support.

I would like to express my appreciation to my advisors Dr. Abdulrahim Shamayleh and Dr. Malick Ndiaye for all the time, effort and guidance they provided me with. It is because of their motivation and inspiration I managed to get the best out of this experience.

Many thanks to the committee members Dr. Mahmoud Awad and Dr. Norita Ahmad for their valuable suggestions and insightful comments. Last but not least, I would like to express my gratitude to Dr. Moncer Hariga for the special and individual care and support he provides each student with. It was an honor to learn from all of my remarkable professors.

Dedication

To the joy of my life, my family.

Abstract

Batch scheduling and processing play a critical role in many manufacturing and service industries. They are widely used in service industries such as banking to process data which makes them of great importance since data communication, monitoring and execution are essential whether they are done online or offline. Batch processing is defined as the execution of a set of required tasks within a specific time frame without violating predecessors' requirements and constraints set by the client. The goal is to achieve the agreed service level contracted with clients using the minimum amount of resources. This research investigates the scheduling problem of processing a set of tasks of non-identical sizes and priority using a set of processors. The objective is to minimize the data batch processing cost while taking into consideration the available resources and the tasks predecessors and constraints. Different types of costs will be included which are: servers and software basic leasing cost, rental cost for additional resources needed in case of overload and extra work, penalty cost of failing to execute the batch process as per the Service Level Agreement (*SLA*), and the opportunity cost representing the cost of idling a resource for any period of time due to inefficient task allocation. An iterative algorithm with an optimization model at each iteration was developed to optimize the data batching process while minimizing the aforementioned costs. A sensitivity analysis is conducted by varying the main model parameters, one at a time to study their impact on the total cost and the problem under study. Also, different network sizes and complexities were tested to study the effectiveness of the developed algorithm. It was found that it is more effective to include all types of costs in one optimization model along with priority, weight, predecessor and time factors. The algorithm proved its effectiveness by allocating files with higher priority and weight prior to other files while taking into consideration time and different types of costs which led to lower batch process total cost. It is recommended that penalty cost and extra processors different costs should be negotiated thoroughly between stakeholders prior to signing the *SLA* since it was found that those costs affect the time and number of rented extra processors which consequently affects the whole batch process.

Search Terms: Data batching, scheduling, processing cost, parallel processing, optimization, multi-processing.

Table of Contents

List of Figures

List of Tables

Abbreviations

SLA – Service Level Agreement

BW – Band Width

DCSDBP – Dynamic Cost Scheduling for Data Batch Processing

BDPS – Batch Data Processes Scheduling

MOM – Mother of Executing Jobs

JES – Job Entry Subsystem

JCL – Job Control Language

EOD – End of Day

GA – Genetic Algorithm

FIS – Fuzzy Inference Systems

FCFS – First Come First Served

RES – Reservation

SCDF – Smallest Cumulative Demand First

QoS – Quality of Service

MD – Modified Delay heuristic

GRASP – Greedy Randomized Adaptive Search Procedure

LTF – Longest Case First

Chapter 1: Introduction

Chapter 1 presents an overview of data batch processing, related definitions, difference between online and batch process, and data batch process's main components. In addition, the chapter highlights the problem statement, the research objectives and significance, and the research methodology.

1.1 Overview

Data batch processing is one of the main instruments and applications used for data processing in business networks. In today's world, economies and commodity markets are swinging rapidly. Barriers to global competition are disappearing, and customers are changing preferences and expectations faster than businesses can respond. At the same time, personal, organizational and business networks are becoming more interconnected, instrumented and intelligent [1]. Data batch process can be defined as the execution of data files as input batches using available resources and gathering the resulted files as output batches. It has many applications such as: billing, issuing reports, sorting files and others [2, 3, 7, 9, 13]. The main feature of data batch processing is its ability to handle huge amount of data files [7, 8] which made it attractive and essential to many organizations in different business fields.

Batch processing is a critical topic that has been associated with mainframe computers since the earliest days of electronic computing in the 1950s [2]. IBM mainframe z/OS operating system or platform has the most evolved set of batch processing facilities which makes IBM the owner of the data batch process platform and the many patents that cover it [2].

Data batch process handles data files by allocating them as batches to the available processors for execution and obtaining the output files as a batch while satisfying files priorities and predecessors' requirements set by the client within a specified time frame. The above explanation shows that a major component of the data batch process is missing and that is cost. It was of a surprise while searching the literature to find that cost associated with the process was never tackled in the proper way by defining all of its types and including them in the batch process. Also, cost was never been included in all the attempts to schedule input batches to resources (processors). The motivation to start this research is to consider all types of costs related to the batch process along with the rest of batch process factors in one optimization model bearing in mind the fact that not all aspects related to data batch process was taken into consideration.

Our objective is to schedule jobs in the form of input batches to available processors while satisfying all constraints and priorities within a specified time frame, considering the different types of associated costs. The main contribution of this thesis to the literature and companies which process their data in batch processing is introducing an algorithm that solves the batch processing problem while considering costs. The algorithm is called Dynamic Cost scheduling for Data Batch Processing (DCSDBP). The developed algorithm is tested for different networks, complexities, by varying different parameters.

1.1.1 Data batch processing

Data batch processing is the execution of collections of jobs (data files) on a computer that runs at a scheduled time or on a needed basis without interaction with users and with no or minimal interaction with a computer operator [2, 3]. A program takes a set of data files as input, processes the data, and produces a set of output data files. This operating environment is termed as "batch processing" because the input data are collected into batches or sets of records and each batch is processed as a unit. The output is another batch that can be reused for computation [2].

Batch processing contributes to the major part of the workload on mainframe computers. A large mainframe will often run several thousand batch jobs every day [4]. This “network” of jobs represents a business workflow with complex interrelations requiring careful scheduling and prioritizing to ensure that all batch jobs run in the correct order and meet strict deadlines [4]. In general, Jobs are gathered in a queue and will run when the user places a request. Job scheduler will schedule them as per pre-determined policy and the processor will start the batch process when Mother of Executing Jobs (MOM) gives the order [5]. Figure 1 illustrates a typical data batching system [5].

1.1.2 Difference between 'batch process and online job'

Data communication and execution can be done either online or offline as a batch process. The difference between the two types should be clarified to understand the characteristics and importance of each one, especially the one under consideration which is the offline type named “Data batch process”.

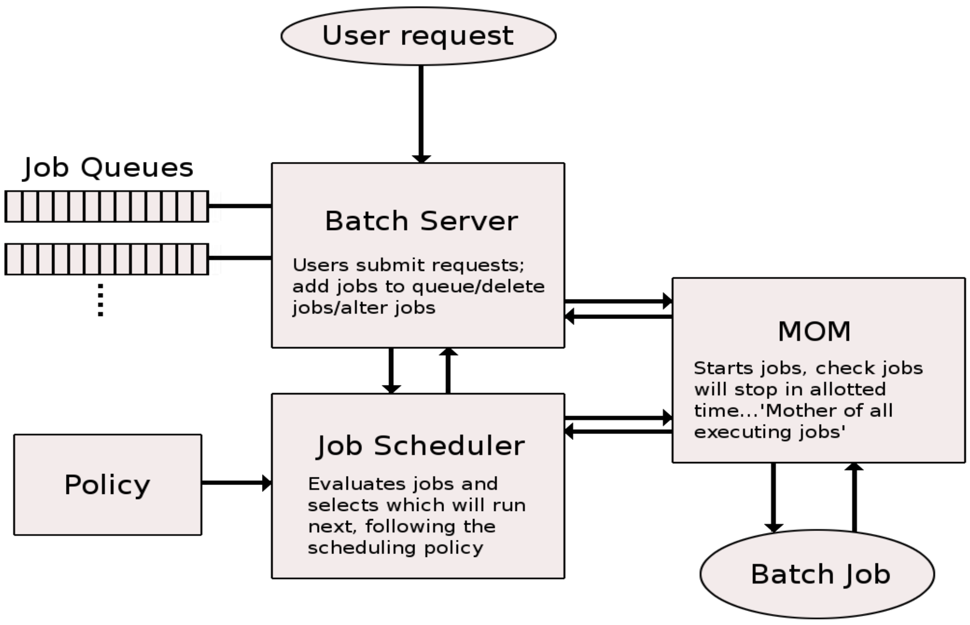
User request
Job Queues
Batch Server
Users submit requests; add jobs to queue/delete jobs/alter jobs
MOM
Starts jobs, check jobs will stop in allotted time...'Mother of all executing jobs'
Policy
Job Scheduler
Evaluates jobs and selects which will run next, following the scheduling policy
Batch Job

Figure 1: Data batching process [5]

Data batch process includes processing transactions in a group or batch, where no user interaction is required once batch processing is underway. This is the main difference between batch processing and "online" or interactive programs which prompts the user for inputs and involves processing transactions one at a time [3, 6, 7, 8].

After loading the data into the system, batch processing does not usually require any further interaction from the user. Therefore, the user interface is often code-based. The user enters the parameters of the batch and then leaves them to carry on allocating data files to available resources [6, 7, 8]. Figure 2 illustrates the difference between batch job and online interactive transaction [6].

1.1.3 When and where to use data batching?

Data batching is used when non-continuous (non-real time) processing of data, instructions, or materials are accepted. In data transmission, batch processing is used for very large files or where a fast response time is not critical. The files to be transmitted are gathered over a pre-specified period of time and are then sent together as a batch [7, 8].

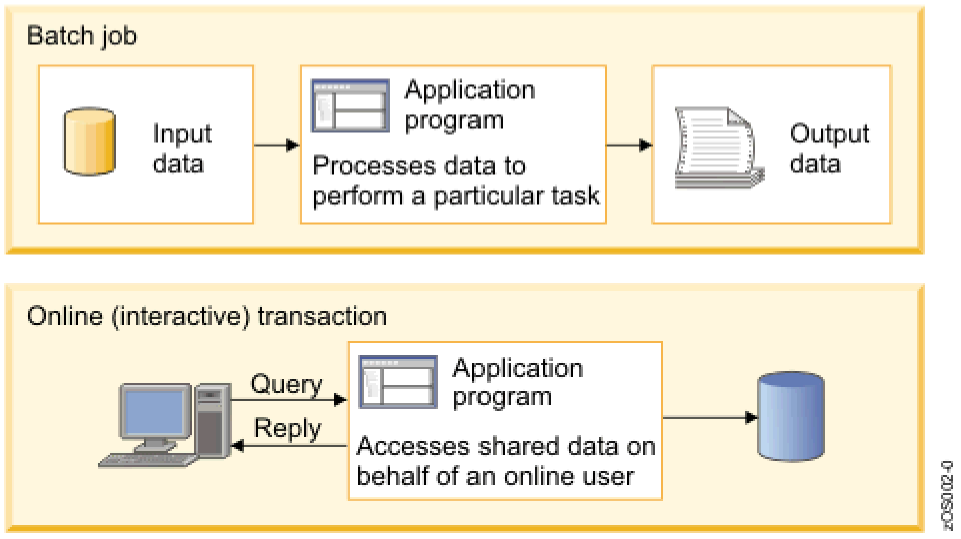
Batch job
Input data
Application program
Processes data to perform a particular task
Output data
Online (interactive) transaction
Query
Reply
Application program
Accesses shared data on behalf of an online user
zOS002-0

Figure 2: Difference between batch job and online interactive transaction [6]

Most mature mainframe systems rely on batch jobs to perform significant portions of the total application logic. The types of tasks undertaken include [2, 3, 7, 9, 13]:

- Merging the day’s transactions into master files.
- Sorting data files for optimal processing the following day.
- Merging data from multiple locations.
- Providing daily, weekly, monthly and annual reports.
- Producing daily, weekly, monthly and annual bank statements to send out to each account holder.
- Issuing daily, weekly and monthly bills or invoices to customers such as (electricity, gas, telephone and credit card bills).
- Performing daily, weekly, biweekly and monthly payroll calculations which results in a set of pay slips to be issued. The salaries of employees can be printed at the end of period by the batch system.
- Consolidating multiple orders into single shipments and invoices.
- Printing checks.
- Performing special mailings.
- Applying interest to financial accounts.
- Batching orders for transmission to another company.
- Performing backups.
- Archiving data.

- Auditing transactions and systems.
- Fraud detection system.
- Other than business fields, scientists may have a huge amount of data on an experiment that could be submitted to a super-computer as a batch run. Once completed, an output data set is available for further analysis and visualization.

Some companies may be able to survive for a day or two without running batch systems, but most will find business operations degrading rapidly if the batch applications are not run [4].

1.1.4 Manual and automatic batch close

The point-of-sale terminal or credit card processing software can be set on manual batch close or automatic batch close. If it is set on manual batch close, the merchant will need to batch out at the end of each day. This sends a command to the processor to settle all transactions that have been entered. Once a batch is settled, a report is usually printed showing the transaction totals in the batch. Before a batch is settled, changes can be made to existing transactions in the batch. For example, one may want to void a transaction, or change an amount of one of the transactions. Changing the amount is typically done for a merchant that enters tips, such as a restaurant. In the case of tips, the amount of the transaction is adjusted to include the tip before the batch is closed [10].

In automatic batch close, no manual intervention is needed by the merchant. Instead the terminal or software will automatically close the batch (settle the transactions) at a certain time each day, or in some case the processor will settle the batch at the processor level (this type of settlement is called host batch close). It is recommended for most businesses to setup on automatic batch close unless manual intervention is required, in which case manual batch close would be the better option. Usually for automatic and manual batch setup, processors will charge a small fee equal to a single transaction fee at the time when the batch is closed [10].

1.1.5 Batch operating system

Jobs are set up so they can be run to completion without human interaction meaning that users of batch operating system do not interact with the computer directly [2]. In other words, each user prepares his job on an off-line device and submits it to the computer operator. To speed up processing, jobs with similar needs are batched together and are run as a group. Thus, the programmers left their programs with the operator. The operator then sorts programs into batches with similar requirements [11, 12].

Computer operator sets parameters at the start of a batch job and changes data storage devices when prompted. These input parameters are predefined through scripts, command-line arguments, control files, or job control language[2]. Many of today's batch systems automate these tasks so there is no interaction with humans unless something goes wrong. Figure 3 illustrates batch operating system [12, 13].

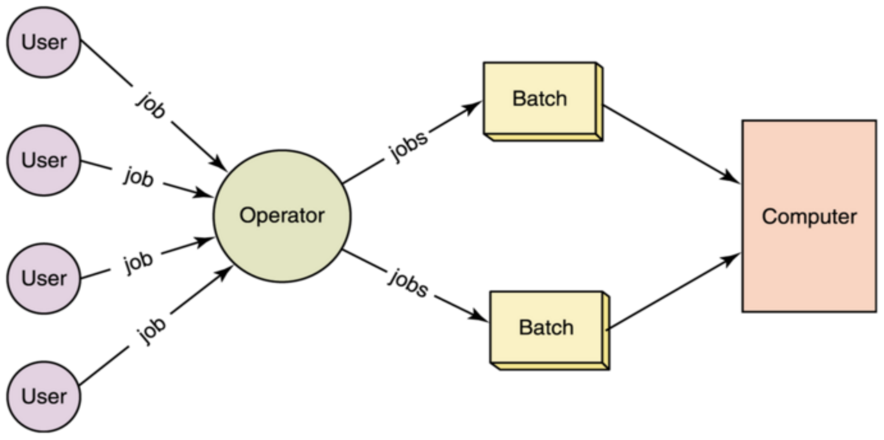

Figure 3: Batch operating system [13]

1.1.6 Batch process benefits and difficulties

Data batch system has many advantages that makes it highly utilized by enterprises of all sizes where it can be used in many ways such as: [2, 3, 7, 11, 13]

- It can shift the time of job processing to when the computing resources are less busy.
- Avoids idling the computing resources with minute-by-minute manual intervention and supervision.
- Allows the system to use different priorities for interactive and non-interactive work.
- Rather than running one program multiple times to process one transaction each time, batch processes will run the program only once for many transactions, reducing system overhead.
- Repeated jobs are done fast in batch systems without user interaction.

- No special hardware and system support is needed to input data in batch systems.
- Manage large volume of transactions
- Best for large organizations but small organizations can also benefit from it.
- Batch systems can work offline so it makes less stress on processor.
- Processor consumes good time while processing, which means it knows which job to process next. In real time systems we don’t have expectation time of how long the job is and what is estimated time to complete it. But in batch systems, the processor knows how long the job is as it is queued.
- Sharing of batch system for multiple users.

Although data batch system has many pros which makes it useful, there are few difficulties associated with it. Some of them are: [11, 13].

- Batch systems are tremendously costly.
- Computer operators must be trained for using batch systems.
- It is difficult to debug batch systems.
- Time delay between collecting the input data and getting an output. It can also be frustrating to find out only later that a batch run has failed due to a data input problem.
- If some job takes too much time i.e. if error occurs in job then other jobs will wait for unknown time.
- Difficult to provide the desired priority.

1.1.7 Batch window

A batch window is "a period of less-intensive online activity", when the computer system is able to run batch jobs without interference from online systems [2].

Many early computer systems offered only batch processing, so jobs could be run any time 24-hour/day. With the advent of transaction processing, the online applications might only be required from 9:00 a.m. to 5:00 p.m., leaving two shifts available for batch work [2], in this case the batch window would be sixteen hours. The problem is not that the computer system is incapable of supporting concurrent online and batch work, but the batch systems usually require access to data in a consistent state, free from online updates until the batch processing is complete [2].

Typically, before and after each batch window critical files backup is made and database is updated.

1.1.8 Partitioning concurrent batch & parallel processing

Parallel processing is the use of more than one processor to execute or run many different jobs at the same time. Ideally, parallel processing makes a batch run faster because there are more resources (processors) running it concurrently [14].

Partitioning is a processing option where execution of many instances of the same job is performed at the same time. This is another way to speed up the batch process [14]. Figure 4 illustrates parallel processing [15].

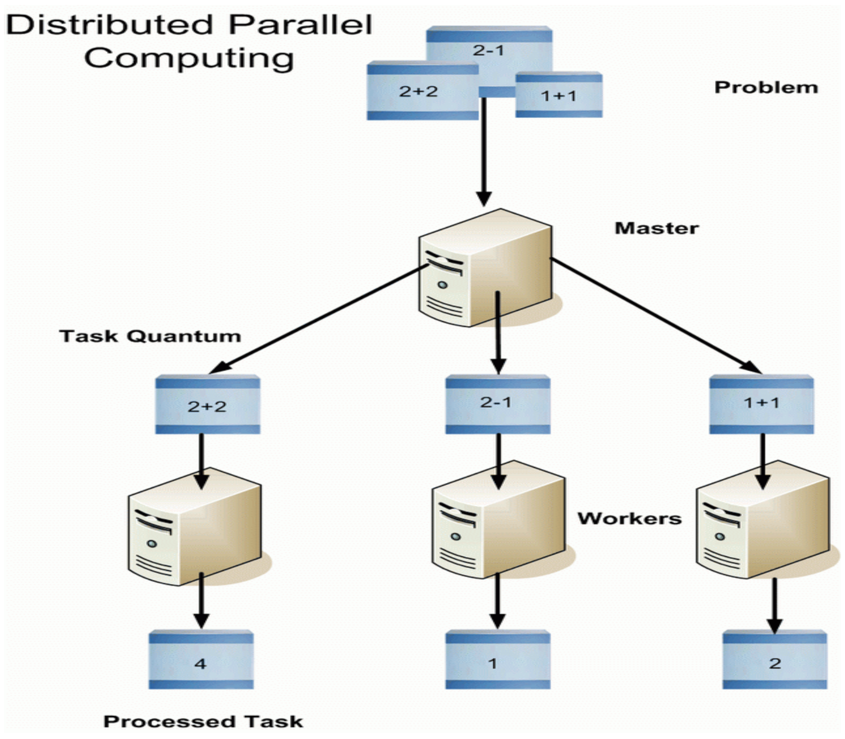

Figure 4: Parallel batch processing [15]

1.1.9 Batch failure and recovery

The Batch Processor uses parallel processors to progress through cases quickly and efficiently. However, when using multiple parallel processors, it is possible for an error to occur. Generally, errors might prevent some, but not all, of the processors from continuing their work. It is important in this scenario that the data batch process doesn't stop and the batch processor handles the error gracefully where it can be restarted to complete the processing for any cases that were missed [16]. There are two types of errors that can occur when the batch processor is running: fatal and non-fatal. The Batch Processor manages occurrences of these errors separately [16].

1.1.9.1 Fatal errors. Fatal errors are unexpected errors that apply to a processor as a whole; for example, when the database of the processor, which is reading from or writing to, becomes unexpectedly unavailable. Fatal errors may affect one or more of the parallel processors, but do not necessarily affect all parallel processors [16].

When a fatal error is encountered, the error details are logged according to the log configuration, and the affected processor is stopped. When using multiple parallel processors, any unaffected processors will continue working to ensure that as many cases as possible are processed and the batch process is carried on. The final summary message provided by the Batch Processor will indicate that an error occurred during processing, and will identify the total number of cases that were successfully processed [16].

1.1.9.2 Non-Fatal errors. Non-Fatal errors are predictable errors that apply to a single case only. An example of these non-fatal errors is data validation error (such as trying to read the value 'abc' into a numeric attribute).

When a non-fatal error is encountered, the error details are logged according to the log configuration. Thus, the affected case is ignored, the batch process is carried on and the processor continues on to the next case. The final summary message provided by the batch processor will identify the total number of cases that were successfully processed and the total number of cases that were ignored due to non-fatal errors [16].

1.1.9.3 Recovery after failure. If the batch processor has encountered a fatal error, it is likely that there will be cases that were not processed. Once the cause of the fatal error has been identified and resolved, the batch processor can be run again to reprocess all cases, including those missed previously due to the fatal error [16].

1.1.10 Batch system components

The main components of batch system are [4]:

1. **Job entry subsystem (JES)**: It queues and assigns jobs to initiators based on the job priority set by the client and the availability of initiators. It can be given a priority order for job classes so that jobs of a higher priority are always executed before jobs of a lower priority [1].

2. **Initiators:** are sets of machine resources that can support one job at a time. They are responsible for running a job. Initiators are used to:
   - ensure that jobs do not conflict in data set usage.
   - ensure that single-user devices (tape drives) are allocated correctly.
   - find executable programs requested by jobs.
   - clean up after the job ends and request the next job.

   System programmers spend considerable time deciding the number of initiators that can be created for batch processing without affecting the performance of online workloads.
3. **Job:** One or more executable programs. It is considered the surface layer of batch systems. A job consists of multiple tasks. In other words, a collection of tasks that is used to perform a computation is known as a job. A task is often a part of a job, sometimes the only part.
4. **Steps:** Every job consists of a number of steps, and each step essentially does two things:
   - Indicates the program to be executed in the step and defines the files to be used as input to and output from that program.
   - Sets up the environment within which the program should run (such as job priorities, and print file classes).

   Generally the steps in a job are logically connected and make sense to run as a batch, such as end-of-day processing or calculating monthly interest payments.
5. **Sort:** Data sorting is a common function of job steps. This is because it is usually quicker to arrange data into an appropriate order for the application process than to have the process fetch each record from an indexed file or database that does not have the data in the desired order.
6. **Job class:** It is used to group jobs by priority or function and to determine which initiator(s) can be used to execute the job**.** Every job has a job class, and system programmers specify the classes that can be executed in each initiator. Configuring the properties, number of initiators and the classes assigned to the initiators provides the ability to give different job classes different amounts of processing power or priority.
7. **JCL:** (Job Control Language), a set of instructions executed together and

written in IBM JCL. It is the language used to describe the steps of a batch job (including file name cross-references, executable commands, control flow and programs to be run) and to provide control parameters to the JES as well as instructions on which programs will be run in the job's steps and which files or databases will be accessed or created by those jobs. JCL can contain conditional instructions or branches so that different actions can be taken based on the input to the job and the results and failures of the job steps.

Through JCL, you specify: who you are (important for security reasons) and which resources (programs, files, memory) and services are needed from the system to process your program.

8. **PROCs:** are prepackaged frequently used sets or features of JCL stored in the system catalog. JCL statements "PROCs" might be used by several batch jobs which means that blocks of JCL that are similar or identical across many jobs can be maintained in one source file and be involved when needed by each job.
9. **Scheduler:** handles the following tasks:
   - Scheduling job start times, and specifying days and times that jobs run may be determined by complex rules that account for public holidays or leap years or for working around the schedules of other batch jobs.
   - Codifying dependencies between jobs and their runtime requirements. The execution of some jobs may be dependent on the execution of other jobs or the arrival of data from third parties.
   - Providing reporting and alerting capabilities. Other scheduling contingencies involve postponing or continuing with a job run if a particular job fails.
   - Job schedulers have thousands of stored parameters for documenting the event sequence requirements for thousands of jobs.
10. **Starting Jobs:** Originally, jobs were started by operators following instructions for when each job should be started and the order in which jobs should be run. Today, jobs are generally started by job scheduling software.
11. **Business Rules:** Although the programs run are responsible for the main business processing, the jobs and scheduling systems generally embody many business rules that are vital to preserve.
12. **Programs:** A large part of migrating batch applications involves the programs

executed by the steps.

13. **Data files and databases:** The purpose of batch jobs is to manipulate and/or report on the data held in data files and databases. IBM mainframes support a large variety of data files and databases such as Oracle.

Figure 5 provides a simplified view of these components [4].

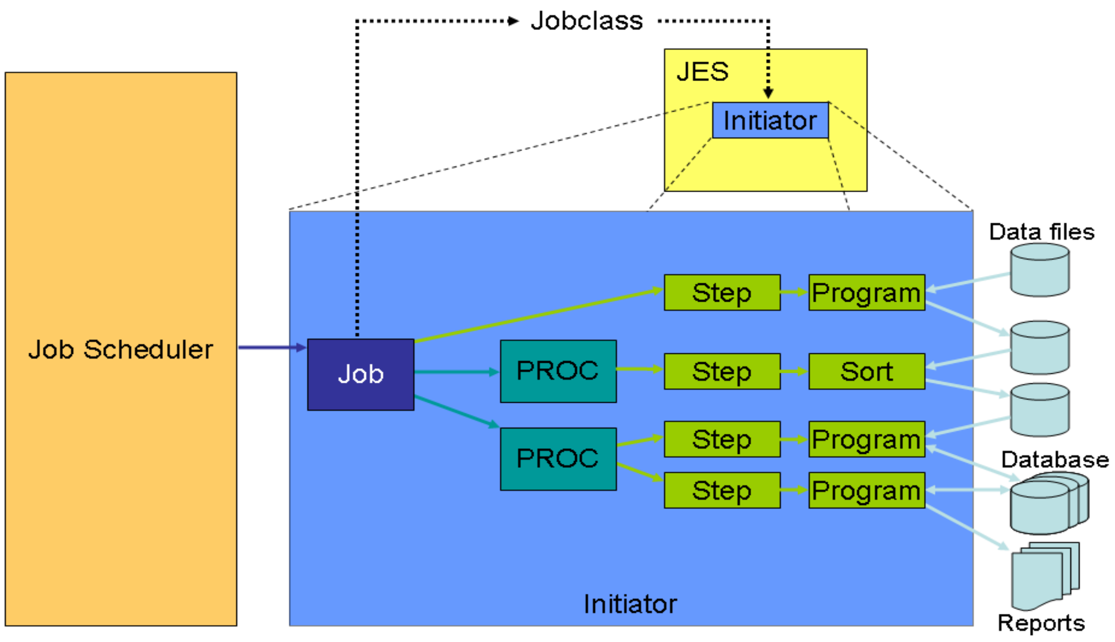

Figure 5: Data batch system components [4]

The processes illustrated in figure 5 can be described as follows [4]:

- The job scheduler determines when a job runs and starts the job.
- The job runs within an initiator—a set of machine resources such as memory and processor time. The mainframe operating system is configured for a limited number of initiators.
- Initiator assignment is determined by the JES.
- Jobs consist of one or more steps.
- The steps may be coded directly in the JCL that defines the job or in procedures (PROCs) invoked from the job, which is also written in JCL and stored in the libraries.
- Typically, program steps read and update files or databases and create reports.

1.1.11 Batch system industrial applications

Batch process system has wide applications in many industries ranging from chemical, pharmaceutical and food industries all the way to plants and heavy industries. Other than Data Batch systems , different plants use batch systems in their production lines where their raw materials and assembling parts are gathered in batches and undergo a certain manufacturing process based on predetermined priorities to end up with an output batch that represents the next stage or end products. It is clear that our model to utilize data batch system can be generalized to be applied in manufacturing batch systems and other batch systems where input batches can be allocated efficiently and effectively to available machinery while all aspects are taken into consideration including different associated costs.

1.2 Problem Statement

Batch processing plays a critical role in daily operations carried out in most organizations in different business fields. This process is widely used in the banking sector due to the need for it in processing end-of-day (EOD) jobs [2] which includes most of the highly important and frequently used bank activities such as: preparing employees' payroll, interest rate calculations, customer's bank statements, credit card companies processing bills, fraud detection data and many others. Although data batch process is applicable in different fields, the scope of our work is mainly focused on the banking sector since it relies heavily on data batch processing in their daily operations.

In previous work, researchers have focused their effort on addressing most of the aspects related to data batch process, especially scheduling. Even though the needed resources are costly and should be part of the decision process, only few papers dealt with the cost issue which initiated the motivation of the work in this thesis.

The high cost of software and hardware resources used in batch process and the patents rights of IBM which own this platform made it beneficial and essential for data batch process users to minimize the cost associated with it. In today's rapidly changing business world, every firm is looking to cut down expenses especially medium to high scale organizations (such as banks). It is those companies which need to process their data using batch process on a regular and intense basis and those are the ones which will appreciate a significant minimization in the data batch process cost without compromising other aspects of the process; in other words, getting the data batch process done efficiently with minimum possible cost. There are five types of costs in the data batching process: First, the service provider has to lease the required number of resources, and hardware processors from sole provider who owns the platform "IBM" as per the job requirements. This is usually the basic cost, however, there are other costs where in certain occasions, more tasks are assigned for processing. In that case, more resources are needed and those will be rented in a higher rate since they were not negotiated in the beginning stages. Third, there is the software cost, which shall be rented from IBM as well. Fourth is the penalty cost, which represents the fees the service provider has to pay to the client in case the required results could not be met according to the contract between the two parties referred to as service level agreement (*SLA*). Lastly, the opportunity cost represents the cost of idling a resource for any period of time

The main problem studied in this thesis is data batch process optimization. The problem is addressed in terms of the following aspects or sub-problems: scheduling, time and cost while satisfying all client's priorities, predecessors and constraints. The objective of the work is to address the problem of minimizing the amount of resources (processors and software), needed to process a group of tasks according to a certain set of priorities while satisfying all the predecessors and constraints, and maintaining the lowest possible cost.

1.3 Research Objective and Significance

This research is tackling the significantly effective and widely used data batching process. The work focuses on scheduling a set of required jobs to be processed in a batch using the available resources (i.e. processors) as per the predecessors, job priorities and constraints stated in the *SLA* while batch job's predecessors and priorities are specified by the client depending on the type of tasks handled. The main constraints taken into consideration are: time "batch window limitation" and process associated costs. In the literature related to batch process, the focus was on scheduling methods to fulfill customer needs using available resources while satisfying predecessors and constraints without trying to minimize customer cost. This research targets data batch process in all its aspects including minimizing the associated five types of cost mentioned earlier. Consequently, this work will achieve a better schedule, and results, and will increase the utility of data batch process. Bearing in mind how costly resources used in batch process are, our model is considered to have a huge impact equally on industry and literature.

1.4 Research Methodology

The problem illustrated in this research was studied and analyzed as per the following steps:

Step 1: Reviewing the literature related to data batching process, especially papers dealing with batch scheduling algorithms and effective batching process.

Step 2: Developing an algorithm which includes an optimization model at each iteration. The model's assumptions, objective function, constraints, decision variables, and problem parameters were defined. The developed algorithm is called Dynamic Cost scheduling for Data Batch Process (DCSDBP).

Step 3: Coding the formulated model using LINGO 15.0 x64.

The computer that the LINGO 15.0 x64 software is installed on has the following specifications:

- Processor: Intel (R) Core (TM) i7-4510U CPU @ 2.00 GHz
- Installed Memory (RAM): 8.00 GB
- Operating System: Windows 8_OS
- System Type: 64-bit Operating System
- Hard Disk: 30 GB.

Step 4: Testing the DCSDBP scheduling algorithm and comparing it to a benchmark.

Step 5: Performing sensitivity analysis by varying the model parameters to study their effect on the developed model. Different network sizes and complexities were tested as well.

1.5 Thesis Organization

Chapter 1 introduces the main definitions related to the concept of data batch process and its basic components using the necessary illustrative figures. In addition, chapter 1 also states the research problem, objective and significance and methodology. Chapter 2 is dedicated to review all the relevant literature done before proceeding with the model. Chapter 3 presents the developed mathematical model along with a numerical example. The result of the sensitivity analysis conducted on the developed model is provided in Chapter 4. Finally, conclusion and suggestions for future work will be in discussed in Chapter 5.

Chapter 2: Literature Review

Modern computer network and distribution systems are known for their massive complexity which makes traditional approaches to resource allocation impractical and hard to optimize, nevertheless, extremely costly. Overall, relevant literature dealt with different aspects of resource allocation such as: methods, effectiveness, optimization and process time. However, only limited amount of work considered cost as a way of optimizing the resource allocation issue whether it is utilized in online or batch systems.

This literature review section is divided according to the research sub-problems structure as explained in the problem statement. It is divided into scheduling, time, and cost. Also a fourth section reviewing batch process optimization in the chemical engineering field was added due to its relevance to the data batch process. Table 1 summarizes the most relevant papers in the literature.

Table 1: Summary of relevant papers in the literature

Author	**Scheduling**	**Reducing time**	**Optimization**	**Market based allocation**	**Methodology**	**Application**
Page *et al.*	●	●			Genetic algorithm, combined with eight common heuristics	Heterogeneous distributed system
Mendez *et al.*	●				Modeling optimization approach	General networks and sequential batch plants
Osman *et al.*	●				Dynamic optimization algorithm modeling	Multiprocessor computer network
Lim and Cho	●				Fuzzy inference with user preference model	Computer operating systems
Xhafa and Abraham	●				Computational models using heuristic and meta-heuristic approaches	Grid computing systems

Table 1: Summary of relevant papers in the literature (continued)

Author	Scheduling	Reducing time	Optimization	Market based allocation	Methodology	Application
Aida	●				Scheduler Performance evaluation using simulation	Parallel computer systems
Stoica *et al.*	●				Developing micro-economic paradigm for online scheduling	Multiprocessor systems.
Stoica and Potheny	●				Simulation and comparing	Multiprocessor systems.
Agarwal and Kumar	●				Developing algorithm	Grid environment
Andresen and McCune	●				Modeling	Processors clusters
Islam *et al.*	●				Developing scheduling heuristic	Computer networks
Srinivasan *et al.*			●		Interpretation of optimal solutions using analytical and numerical methods	Chemicals industry
Srinivasan *et al.*			●		Analyze measurement-based optimization strategies	Chemicals industry (penicillin production)
Zhou *et al.*			●		Systematic model design and hybrid optimization strategy for mixed –integer nonlinear programming (MINLP)	Chemical industry (water allocation system)

Table 1: Summary of relevant papers in the literature (continued)

Author	**Scheduling**	**Reducing time**	**Optimization**	**Market based allocation**	**Methodology**	**Application**
Damodaran and Vélez-Gallego		●			Simulation, lower bond computation and approach comparison	Parallel batching machines
Mehta *et al.*	●	●			Analytical and simulation model to scheduling algorithms	Parallel Database Systems
Grigoriev *et al.*	●	●			Two-phased LP rounding technique to build scheduling model	Production planning
Bouganim *et al.*		●			Experimentation	Data integration systems
Ngubiri and Vliet		●			Modeling	Parallel queuing systems
Arpaci-Dusseau and Culler		●			Extending existing scheduler	Parallel distributed environment
Ferguson *et al.*				●	Human economic model	Distributed systems and computer networks
Kuwabara *et al.*				●	Market model simulation	Communication network control
Chun and Culler				●	Modeling, simulation, synthetic workloads and user-centric performance metrics	Cluster computing workstations
Sairamesh *et al.*	●			●	Modeling	Packet networks
Yeo and Buyya	●			●	Taxonomy	Cluster computing systems

Table 1: Summary of relevant papers in the literature (continued)

Author	Scheduling	Reducing time	Optimization	Market based allocation	Methodology	Application
Islam *et al.*				●	Framework design	Computer systems
Mutz and Wolski				●	Vickrey Auction novel implementation	Computer systems
Mutz *et al.*				●	Proposing and evaluating new Mechanism	Computational settings

2.1 Batch Process Scheduling

Scheduling is a critical issue in batch process operation and is crucial for improving its performance. Many researchers contributed to this field making the scheduling problem one of the most studied problems in the optimization research community.

Page *et al.* [18] considered eight common heuristics along with the genetic algorithm (GA) evolutionary strategy to dynamically schedule tasks to processors in a heterogeneous distributed system. Scheduler operates dynamically, which means it allows tasks to arrive for processing continuously and considers variable system resources. Also, it utilized 8 heuristics which reduces probability of idling processors while waiting for a schedule to be generated. It was found that this approach generates more efficient schedules since it doesn't make prior assumptions about homogeneity or availability of resources. Also using multiple heuristics to generate schedules provides more efficient schedules than using each individual heuristic on its own.

Mendez *et al.* [19] presented different state of the art optimization methods for short term batch scheduling. They started by classifying batch scheduling problems such as: process layout itself (parallel or sequential, single or multiple stage.. etc.), demand pattern (due date, production target over time horizon), resource constraint (labor, utilities), cost associated with (equipment, utilities, changeover) and others; followed by classification of scheduling optimization models focusing on the following aspects: Time representation (whether schedule can take in predefined time point or at any moment), material balance (batch number and size), event representation (fixed time interval, variable global time points, etc..). Modeling aspects of optimization methods were presented and two practical problems were solved using different approaches to illustrate methods performance, highlight their strength and limitations and compare effectiveness and efficiency of these models.

Osman *et al*. [17] proposed a model for data batch scheduling where researchers allocated a number of tasks to capable resources to process these tasks or jobs within cut-off time while satisfying all constraints and predecessors conditions. The presented dynamic approach is called: Batch data Processes Scheduling (BDPS) algorithm. It is a dynamic iterative framework used for assigning required tasks to available resources taking into consideration the predecessors, constraints and priority of each job. The dynamic framework is characterized by constant change in task allocation to available resources to achieve maximum utilization and prevent idling any resource during any period of time. This work was adopted as a starting point in this thesis.

Lim and Cho [20] illustrated a method of process scheduling which adapts to users' preferences and aims to arrange CPU time to multiple processes for providing users with more efficient throughput. CPU time arrangement is quite hard due to processes having different purpose. Generally speaking, there are three purposes class: batch processes, interactive processes and real-time processes. Fuzzy inference systems (FIS), which are also called fuzzy rule-based systems determine process's features in run time, and classify their classes. Then, they model the preferences of users, and finally decide on the priority of each process using fuzzy inference with the information of process class and user's preferences. Although this paper considered minimizing batch process time, its main goal was satisfying user's preferences for the three processes class.

Xhafa and Abraham [21] dealt with Grid technologies which emerged in the first place to provide more computing power needed by the demanding scientific computing community. Scheduling in the majority of grid systems is a complicated problem - when compared to scheduling in classical parallel and distributed systems-, as well as an important mechanism due to: the diverse needs of grid-enabled applications, the different parameters that intervene scheduling and the multiple constraints and optimization criteria in a dynamic environment. They illustrated heuristic and meta-heuristic methods and found them to be appropriate for Grid scheduling alternating traditional scheduling techniques.

In Aida [22] a new aspect of the scheduling problem was tackled. Researchers investigated the effect of job size characteristics on job scheduling performance in a parallel computer system. In order to understand the issue, multiple job scheduling algorithms performance such as first come first served, longest job first, shortest job first, First-Fit and Backfilling were evaluated under various workload models each with a certain characteristic. The results showed that some scheduling algorithms were not affected by job size characteristics and showed best performance such as first fit scheduling. Also, certain job size characteristics affected performance of priority scheduling significantly.

Stoica *et al*. [23] described a scheduler based on the microeconomic paradigm for online scheduling of a set of parallel jobs in a multiprocessor system. Increasing system throughput, reducing response time, fairness in allocating system resources and providing user with control over the relative performances of his jobs were all considered. Every user has a savings account in which he/she receives money at a constant rate. To run a job, the user creates an expense account for that job to which he/she transfers money from his/her savings account. The job uses the funds in its expense account to obtain the system resources it needs. The share of the system resources allocated to the user is directly related to the rate at which the user receives money; the rate at which the user transfers money into a job expense account controls the job's performance.

Stoica and Potheny [24] considered the problem of scheduling online set of jobs on a parallel computer with identical processors. In this paper, they used simulation to compare three microeconomic policies with three variable partitioning policies, first come first served with and without reservation (FCFS and RES) and smallest cumulative demand first (SCDF). Under a systematically designed set of experiments, they show that the other scheduling policies can be considered as limiting cases of the microeconomic scheduling policy.

Agarwal and Kumar [25] considered bandwidth requirement as a Quality of Service (QoS) of a network and availability requirement as a QoS of a compute resource. An Availability aware QoS oriented Algorithm (AQuA) for task scheduling in grids has been proposed. The algorithm is evaluated and compared with QGMM (QoS Guided Min-Min) algorithm in the simulated grid environment. The experimental analysis and results showed that the AQuA algorithm is able to utilize the highly available resources in grid, which therefore increased the reliability to successfully execute applications without adversely affecting the makespan.

Andresen and McCune [26] presented a model for dynamically scheduling HTTP requests across clusters of processors, optimizing the use of client resources as well as the scattered processor nodes. They also presented a system, H-SWEB, implementing their techniques and showed experimental improvements which have been achieved through utilizing a global approach to scheduling requests. A discussion of the system architecture and implementation was provided, as well as a brief summary of the experimental results that have been achieved.

Islam *et al*. [27] proposed a new scheduling heuristic called Normalized Urgency which is based on the notion of prioritizing jobs based on their urgency and their processing times. They proved the optimality of their approach for certain restricted versions of the problem and experimentally compared their schemes against the existing schemes like First Price, First Reward etc. using real super computer center workloads. The results demonstrated that the proposed scheme leads to significantly higher revenue as compared to existing schemes while achieving better performance with respect to standard performance metrics such as slowdown and utilization.

2.2 Batch Process Optimization

There are several papers that addressed the application of batch process optimization in chemical engineering field such as factories and production lines. Srinivasan *et al*. [28] studied achieving batch process optimization in chemical industry in order to face the increased competition by reducing production cost, improving production quality, and meeting safety requirements and environmental regulations. They assumed the existence of an accurate model and they discussed the characterization of the optimal solution without considering uncertainty resulting from model mismatch and disturbances.

Srinivasan *et al*. [29] included uncertainty in their optimality study since accurate models of industrial processes are rarely available. Optimization strategies under uncertainty were overviewed and a novel scheme was proposed where optimality is achieved by tracking the necessary conditions. Instead of adopting model based optimization framework, researchers investigated the use of measurement as a way to optimize uncertain batch process.

Zhou *et al*. [30] developed a cost –optimal schedule scheme where an allocation network was combined and designed simultaneously. They played an essential role in determining the overall profit of the plant while fulfilling all requirements in batch production and satisfied concentration and flow rate constraints imposed at various times and locations in the water network. Batch process schedules and water – allocation networks were incorporated into a single mathematical programming model addressing the interaction between the two systems which results in a better overall optimization and design.

2.3 Reducing Batch Process Time Using Parallel System

Damodaran and Vélez-Gallego [31] developed a simulated annealing algorithm to evaluate the performance of batch systems in terms of total completion time and total weighted completion time. Research goal was to minimize the makespan of batch processing by allocating batches to identical parallel resources for processing. In other words, processing time is the issue here and the research objective is to minimize it. The Modified Delay heuristic (MD) and a Greedy Randomized Adaptive Search Procedure (GRASP) were proposed and used as a lower bond to compare the performance of the SA approach. Scheduling allocating resources method were not emphasized since that was not their major concern in their research.

Metha, *et al*. [32] proposed parallel query scheduling algorithm by dividing workload into batches and exploiting common operations within queries in a batch resulting in significant savings compared to single query scheduling techniques used in parallel database systems. Multiple parallel scheduling method is based on minimizing total execution time for a set of queries (batch) by integrating optimization and scheduling to come up with a plan for concurrent execution and run- time resource allocation. It is noted that common operations exist between different queries in multiple query workload. In case of independent scheduling, same operations are executed multiple times. Techniques developed by researchers in this paper identify common operations in batches of queries, schedule them in such a way that repetition is decreased- in other words shared operations will be executed once- which will reduce incremental load imposed by each query on the system, reduce total execution time and decrease the amount of memory used since one data copy is saved and shared leading to significantly improve performance by making additional memory available to other queries.

Grigoriev *et al*. [33] generalized the classical unrelated machine scheduling problem with the objective of minimizing the schedule makespan adding a resource – time tradeoff. In their paper, different variants of the problem were considered. The processing times of any job-machine pair can be reduced by utilizing a renewable resource, such as additional workers, that can be allocated to the jobs. In other words, a maximum number of units of a resource may be used to speed up the jobs, and the available amount of units of that resource must not be exceeded at any time. They used a two-phased LP rounding technique to assign resources to jobs and jobs to machines. The motivation of this study is to contribute to the production planning field where additional (overtime) workers can be allocated to specific tasks within production in order to reduce the production cycle time.

Bouganim *et al*. [34] studied data integration systems execution plans performance. In this paper, two important correlated problems that arise while processing queries in data integration systems were addressed: unpredictable delays in data delivery and memory limitation. Researchers proposed an execution strategy that reduces the query response time by concurrently executing several query fragments in order to overlap data delivery delays with the processing of these query fragments.

Ngubiri and Vliet [35] proposed a new approach for parallel job schedulers fairness evaluation. Fairness is an important aspect in queuing systems. Several fairness measures have been proposed in queuing systems in general and parallel job scheduling in particular. Generally, a scheduler is considered unfair if some jobs are discriminated while others are favored. The new fairness measuring approach was based on the fact that since jobs do not have the same resource requirements and arrive when the queue and system have different states, they are not expected to have the same performance in a fair set up. Every job, therefore, needs to have a time it would start at if scheduled by an ideal fair scheduler.

Arpaci-Dusseau and Culler [36] used proportional-share scheduler as a building-block and showed that extensions to the above scheduler for improving response time can still fairly allocate resources to a mix of sequential, interactive, and parallel jobs in distributed environment. Simple extensions were implemented and analyzed which provided better response-times for interactive jobs by giving those jobs their share of resources over a longer time-interval.

2.4 Economic Market Based Allocation and Evaluation Methods

Ferguson *et al*. [37] adopted the human economic model and implemented it on resource allocation in computer network system in order to limit the complexity of resource sharing algorithms by decentralizing the control of resources. Decentralization is clearly demonstrated in the fact that economic models consist of two types of agents: suppliers and consumers who attempt to achieve their goals independently and selfishly. A consumer attempts to optimize his individual performance criteria by obtaining the resources he requires and is not concerned with system wide performance. A supplier's main goal is to optimize his performance by increasing profit obtained from allocating his resources to consumers. Each supplier in the economic model owns a set of resources i.e. processors and charges for consumers for using those resources, meanwhile each consumer is endowed with money to buy needed resources. This clearly shows that money and pricing are the main ways to coordinate the selfish behavior of agents. It is worth mentioning that the resource price is determined based on supply and demand from consumers and that the pricing system ensures that a realizable allocation of resources is achieved.

Kuwabara *et al*. [38] presented market-based approach, where resources are allocated to activities through buying and selling of resources between agents and resource allocation in a multi-agent system which is achieved through associating agents with resources and activities. Agents associated with activities (activity agents or buyer) request agents at resources (resource agents or seller) for the resource they require, and resource agents decide whether requests are granted or not. Agents are given utility functions, and each agent takes actions independently of each other to maximize their utility. Agents may not know the global state of the system, and their actions are determined based on limited information of the system. In their model, buyers do not communicate with each other, and thus, a buyer does not know what actions the other buyers take; sellers do not communicate with each other, and thus, a seller does not know the resource prices at other sellers. The only possible communication in market model is between sellers and buyers to send resource requests (from buyers to sellers) indicating how much resource buyers require and resource price notifications (from sellers to buyers). This is the major difference between this model and economic model, which is considered open, which allowed for transparency of information regarding demand and pricing.

Chun and Culler [39] presented performance analysis of market based batch schedulers for clusters of workstations using user-centric performance metrics as the basis for system evaluation. Model built illustrates the allocation of computational resources through markets using auctions, where the cost of using a resource is directly related to supply and demand. Each user had a utility function for each job which measures value delivered as a function of execution time. With aggregated utility as the performance metric, explicit evaluations provide systems with extra information which is used to optimize for user value as opposed to system-centric performance metrics where users have little or no control over how their jobs are given priority in the queue. Pricing combined with charging creates feedback which causes users to balance the amount of work submitted to the system with the cost of obtaining associated resources.

Sairamesh *et al*. [40] proposed a new methodology based on economic models to provide Quality of Service (QoS) guarantees to competing traffic classes in packet networks. Priced and traffic classes compete for network resources and they buy them to satisfy their QoS needs. Researchers developed packet scheduling and admission policies to provide QoS guarantees based on available QoS and prices in the network. Yeo and Buyya [41] outlined a taxonomy that describes how market-based resource management systems can support utility-driven cluster computing. Cluster systems need to know the specific needs of different users so as to allocate resources according to their needs. Market-based resource management systems make use of real-world market concepts and behavior to assign resources to users. In this paper, the taxonomy is used to survey existing market-based resource management systems to better understand how they can be utilized.

Islam *et al*. [42] designed a framework that provides an admission control mechanism that only accepts jobs whose requested deadlines can be met and, once accepted, guarantees these deadlines. However, the framework is completely blind to the revenue these jobs can fetch for the supercomputer center. By accepting a job, the supercomputer center might relinquish its capability to accept some future arriving (and potentially more expensive) jobs. In other words, while each job pays an explicit price to the system for running it, the system may also be viewed as paying an implicit opportunity cost by accepting the job. Thus, accepting a job is profitable only when the job's price is higher than its opportunity cost. In this paper, they analyzed the impact that such opportunity cost can have on the overall revenue of the supercomputer center and attempted to minimize it through predictive techniques.

Mutz and Wolski [43] presented a novel implementation of the Generalized Vickrey Auction that uses dynamic programming to schedule jobs and compute payments in pseudo-polynomial time. Additionally, they have built a version of the PBS scheduler that uses this algorithm to schedule jobs, and have presented the results of their tests using this scheduler. Mutz *et al*. [44] proposed and evaluated the application of the Expected Externality Mechanism as an approach to solving the problem of efficiently and fairly allocating resources in a number of different computational settings based on economic principles. Their mechanism provides incentives for users to reveal information honestly about job importance and priority in an environment where batch-scheduler resource allocation decisions introduce "externalities" that affect all users. Tests indicated that the mechanism meets its theoretical predictions in practice and can be implemented in a computationally tractable manner.

2.5 Chapter Summary

After reviewing all the above literature, it was concluded that only few research studies considered operations cost. Most research mainly dealt with scheduling methods with the objective of minimizing the total execution time; hence that is the gap this research is trying to fill as explained further in the research objective and significance.

Chapter 3: Data Batch Model

3.1 Bank Model

Banking was used as an example to present the data batch processing and to develop and experiment our model with. It was selected due to the fact that banking sector is one of the major players in global economy. Financial institutions find themselves in need for rapid, efficient and effective processes in order to be able to focus on their clients in a holistic fashion [1]. In addition to that, historically banking processes have been bounded by the IT systems that drive them. Integration has been implemented through batch processing and data warehousing. Figure 6 illustrates the complex operations and working environment in financial companies (i.e. banks) [1].

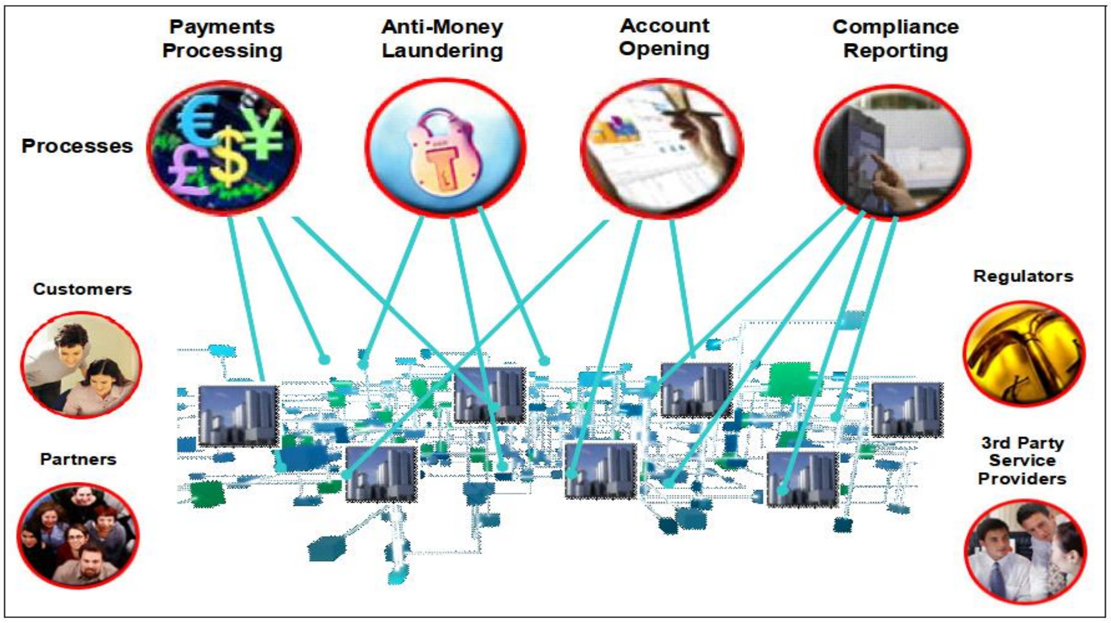

Figure 6: Banks complex processes [1]

The bank has a significant amount of data that need to be processed. Usually a data processing takes place on daily basis but sometimes on monthly basis depending on its type. Examples of the data that need to be handled: employee’s pay roll, interest calculations, customer’s bank statements, credit card companies processing bills and fraud data detection.

Banks usually outsource data processing due to the quantity and diversity of data and lack of resources (i.e. processors). A private company takes charge of arranging and processing the data, and aggregating the output; this process is well known as “data batching”. Therefore, the proposed model will be developed from that third party view; this third party is the data batch process service provider in a form of a private company leasing processors and software from the only source “IBM” and performing the batch process for the bank under a service level agreement (*SLA*). As previously mentioned the (*SLA*) is the contract that states all rules and conditions concerning the data batch process and governing the relationship between the client (i.e. bank) and the service provider (i.e. private company).

Data arrives from the bank daily after closing hour (5:00 p.m.). The processing occur after closing hour because they need to be stable (with no online intervention) so processors will be able to handle the process. Although a lot of processors can handle both online and data batching processes at the same time, the above mentioned reason shows why this method is rarely adopted.

Batch processing begins with scheduling batches; this consist of gathering data in batches and assigning each batch to the appropriate processor. Scheduling data files shall take into consideration job priorities, predecessors and other constraints. This means that jobs won’t be processed simultaneously but rather as per their schedule. The size of each batch depends on the number and size of data files included in it, where each data file may consist of multiple records. Hence, the time taken to process any data file depends on its size and on the number of records included in it.

Scheduling is a very important step since it affects the success of the entire process. The schedule’s size and complexity depend on the size and number of batches, number and type of processors, kind of tasks that need to be performed on each batch (whether there is a lot I/O read /write job or not). Also, scheduling methods vary from First Come First Served (FCFS), Longest Case First (LTF) and other algorithm [17]. Constraints and predecessors of each task are very important issues that should be considered.

There are five types of costs associated with the data batching process:

First: Leasing cost as per the agreement between processors host like IBM and the service provider. This is the basic cost of leasing the necessary resources(processors in this case) to implement the data batching process. Usually the leasing contract specifies hardware rental charges along with the period, rules and other details. Terms and conditions can’t be changed unless both parties agree upon it.

Second: Software cost, which is the cost of using the necessary software from IBM needed to execute the batch process.

Third: Extra processors cost, which is the cost of renting additional processors from IBM. This happens in the case of job overload from the bank side. Obviously rental cost will be higher than leasing cost since it was not negotiated in the first place or maybe not included in the original contract.

Fourth: Penalty cost. Clearly this is the fine that the service provider has to pay in case of failing to execute the data batch process as per the *SLA*.

Fifth: The opportunity cost that represents the cost of not utilizing the resources.

The company attempts to execute the tasks using the available resources with maximum utilization to reduce cost and increase their profit (from bank charges) while meeting the *SLA* (service level agreement). Assuming multiprocessing method will be adopted meaning that several processors will be running simultaneously.

It is assumed that the company has a set of processors. Each processes one task at a time. Also processor reservation is not allowed. Jobs are waiting in queues. The dispatcher release them for processing according to their priorities and precedence.

No pre-assignment to processors is allowed [17]. In addition, with multiple processing, any task can be divided among more than one processor for execution. It is worth mentioning here that a certain processor can execute one job at a time. [17]. After all batches are processed, the resulted output batches shall be collected continuously (as soon as they are ready), and the batching process should be considered to have generated the required outcome either as a software or in printed form where it will be delivered back to the bank first thing in the next morning.

3.2 Mathematical Model Formulation

In this research we study the significant problem of effective data batching process. The main goal is to optimize this process and increase its utility while minimizing cost. In previous work by Osman et al. [17], a dynamic iterative scheduling framework was presented in which they allocated tasks to resources to process within cut-off time while satisfying all constraints and predecessors conditions. The dynamic framework is characterized by constant change in task allocation to available resources to achieve maximum utilization and to prevent idling any resource during any period

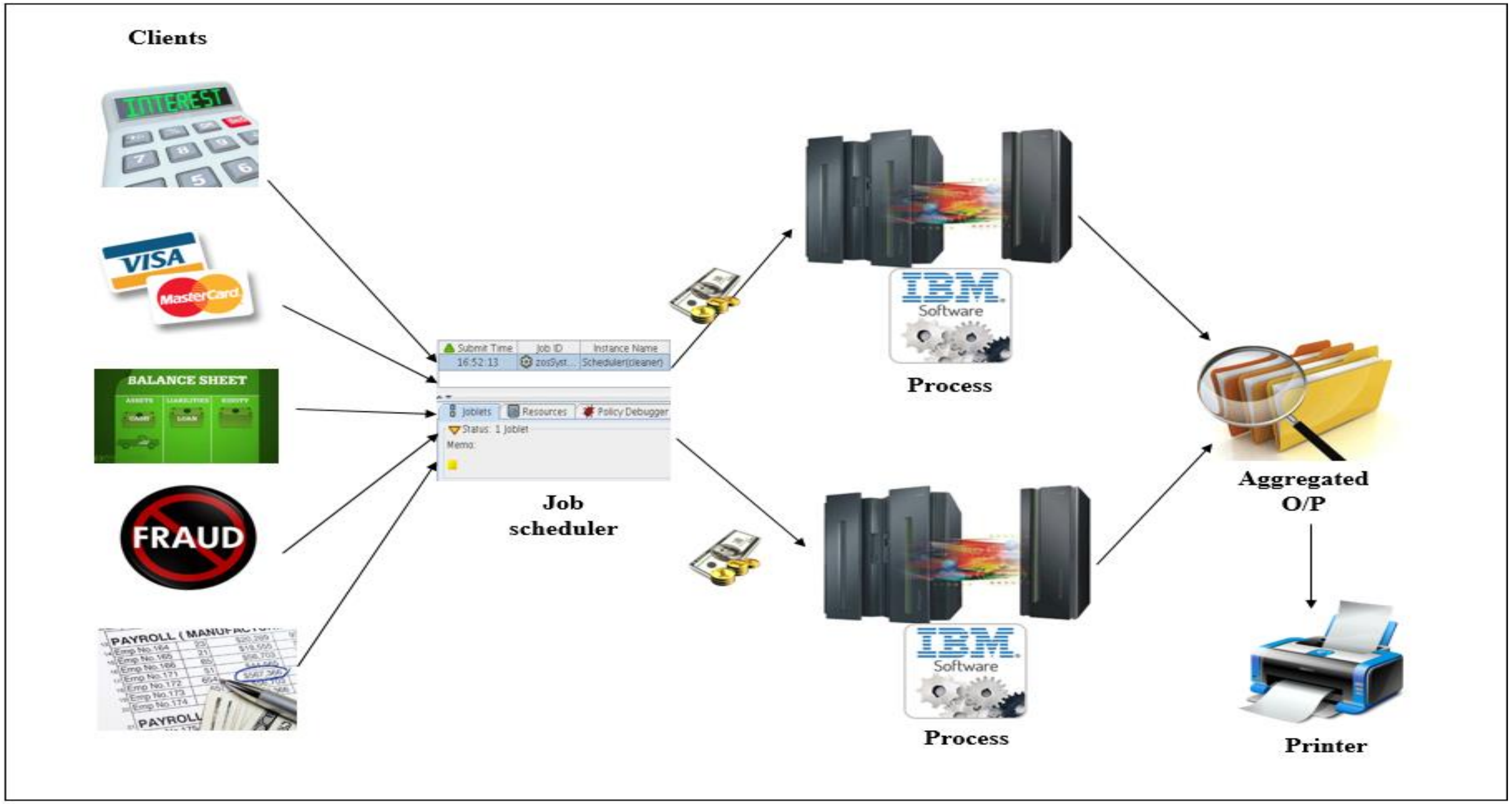

Figure 7: Typical Data Batch Processing System for bank model

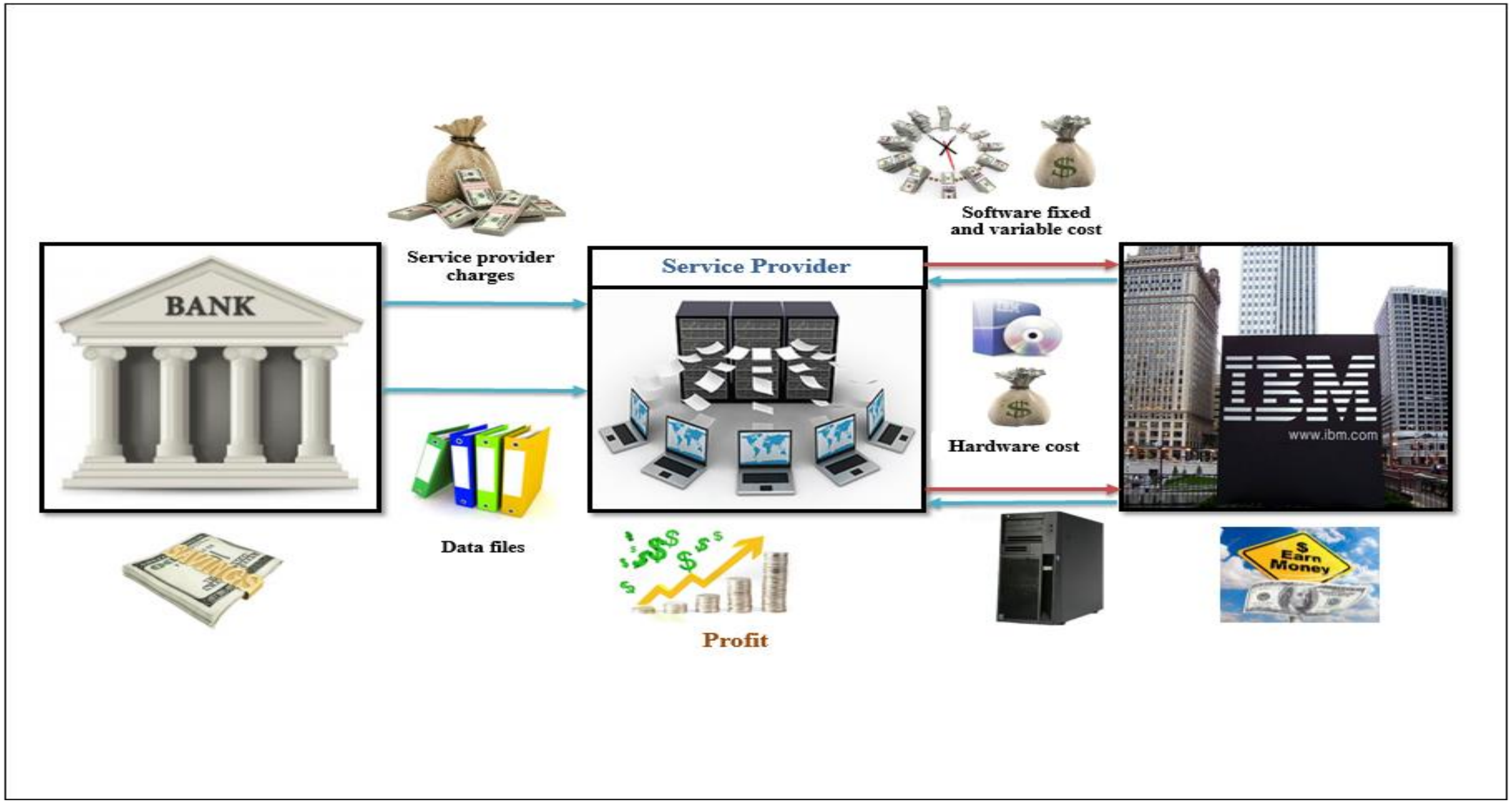

Figure 8: Service provider position with respect to IBM and client

of time. The work presented in this thesis proposes an algorithm for dynamic iterative scheduling to satisfy all data batch processing aspects while minimizing five types of processing costs which are: processors' basic leasing cost, software basic leasing cost, additional resources rental cost in case of overload, penalty cost of failing to execute the batch process as per the *SLA* and the opportunity cost representing the cost of idling a resource for any period of time due to inefficient task allocation. The objective is to minimize the aforementioned costs while satisfying a set of related predecessors, priority and other vital constraints within cutoff time. This resulted in achieving the research goal of maximizing batch process effectiveness and feasibility. The developed algorithm is called Dynamic Cost scheduling for Data Batch Process (DCSDBP). The (DCSDBP) steps are:

1. Preparatory stage

This involves setting up the initial data. The initial data consist of information about the basic and extra processors to state their availability, subset of data files ready for processing, extra processor utilization parameter which is set to zero indicating that no extra processor is used at the beginning of allocation process, data files processing parameters which are set to zero meaning that files are not being processed yet and time loop is initialized where time is set to zero. The end of this stage indicates the start of the DCSDBP algorithm.

2. Setting up data files subset

Continuing on the data files subset from stage one, the files are assigned parameters and weights based on their precedence obtained from dependency matrix.

3. **Solve an iterative network optimization model**

Solve an optimization model in the algorithm to allocate data files to the available processors taking into consideration file weight and priority while minimizing different types of cost for each time unit. The model ensures that all assumptions are fulfilled through its constraints.

4. **Update utilized extra processors**

Solve a mathematical model to estimate the completion time. At this stage, it will be checked if an extra processor should be rented at this time unit to avoid penalty cost.

The model will calculate the critical path duration for the rest of unprocessed activities in the data files network at each time unit, and will compare it to the remaining time till the end of the batch process time that is agreed on in the *SLA*. At each time unit, a trade-off between the cost of renting new extra processor and the penalty cost is made and according to that the model decides whether to rent a new processor or incur a penalty cost. This trade-off is made under the condition that sufficient files are available for processing in case of renting new processor.

5. **Update the availability of files**

At this stage, each data file processing parameter is incremented by the amount of times it was processed .When a file is fully processed, then it gets removed from the subset for the following time unit and the rest of the process.

6. **Termination condition is checked**

When all files are processed, then the program shall stop and the model accomplishes its purpose, else the model will go to step 7.

7. **Update iteration clock**

Increment iteration clock by one time unit in order to allocate the rest of data files to available processors for processing.

Steps 2 through 7 of the (DCSDBP) will be repeated until all tasks are allocated to available resources and there are no more jobs waiting in the queue.

Before presenting the model, the model assumptions, parameters, and decision variables will be introduced.

3.2.1 Model assumptions

The optimization model in this research is developed under the following set of assumptions:

1. A fixed number of basic processors are leased by the service providing company
2. Hardware and software costs are incurred by basic processors. The hardware cost is a fixed amount while the software cost will have fixed and variable cost amounts. The variable cost is charged per unit time of usage.

3. Extra processors can be rented in case of exceeding the *SLA*; however, costs will be higher than the cost of basic processors. It is assumed to be 1.25 times higher than the basic processor hardware and software fixed leasing cost.
4. Hardware and software costs are incurred by the additional processors. The hardware cost is a fixed amount while the software cost will have a fixed and variable cost amounts. The variable cost is charged per unit time of usage.
5. Files are allocated to processors until a predetermined cutoff time is reached, after which the batch process shall continue but penalty cost is imposed on each delay unit of time. The total penalty cost is calculated by multiplying the time delay amount by the penalty cost per unit time.
6. Parallel processing is allowed (processors work simultaneously).
7. Processor reservation is not allowed.
8. Each basic or extra processor processes one task at a time.
9. Limited and predetermined number of data files are being processed.
10. Files can be multi-processed at the same time.
11. The iteration clock time unit is estimated by the time it takes to process the smallest size unit of data file.
12. The batch process total time is a multiple of the iteration unit time.
13. When an extra processor is acquired, it is utilized for the rest of the batch window.
14. Extra processors can be rented one at a time per iteration.
15. Whenever an extra processor is rented at any time unit *T* throughout the batch process, the endorsed fixed hardware and software costs will be calculated from the time of renting till the end of the batch process.
16. At the beginning of each time unit *T*, files are allocated and resources are captured. However, at the end of the time period *T*, all resources are freed and ready for the next time period *T*.

3.2.2 Indices

The following are the indices used in the mathematical model:

i,j	Data file.
k	Basic processor
r	Extra processor

3.2.3 Problem parameters

The following are the parameters of the developed mathematical model:

T	Clock discrete time.
I	Number of all data files.
I'^T	The subset of data file that are available for CI processing at any time *T*.
n_i	Number of processing times required for data file *i*.
q_i^T	Number of times data file *i* has been processed; it is incremented by one every time data file *i* being processed.
SLA	Batch process time as agreed on in the *SLA*.
K	Number of all basic processors.
R	Maximum number extra processors that can be rented in case of exceeding *SLA*.
V^T	Number of rented extra processors files can acquire at any discrete time *T*.
α_i	The data file weight based on precedence/dependency matrix.
β_i	The data file scheduling priority.
BW	Available batch process window.
C_{sf}	Basic processor software fixed leasing cost.
C_{sv}	Basic processor software variable leasing cost given per unit time.
C_h	Basic processor hardware leasing cost.
C_{esf}	Extra processor fixed software rental cost per unit time.
C_{esv}	Extra processor variable software rental cost given per unit time.
C_{eh}	Extra processor hardware rental cost per unit time.
C_p	Penalty cost per unit time of failing to execute the batch process as per the *SLA*.
e_i	Number of time file *i* can be multi processed.
TC_p	Total penalty cost for the total delay time.
T_H	Total cost of renting one extra processor at any time unit *T*.
D^T	Available files total multiprocessing e_i at any time *T*.
T^r	Clock discrete time at which extra processor r is rented.
END	End of batch process.
TBC	Total batch process cost.

3.2.4 Problem decision variables

The followings are the decision variables of the developed mathematical model:

l_{ij}^T	Binary variable equal to 1 if data file *j* immediately precedes data file *i*, and 0 otherwise (parameters of precedence/dependency matrix).
P_k	Binary variable equal to1 if processor *k* is available to receive data file, and 0 otherwise.
W_r	Binary variable equal to1 if processor *r* is available to receive data file, and 0 otherwise.
f_i^T	Binary variable equal to 1 if data file *i* is available for processing at discrete time *T*, and 0 otherwise.
X_{iK}^T	Binary variable equal to 1 if data file *i* allocated to processor *k*, and 0 otherwise.
Y_{ir}^T	Binary variable equal to 1 if data file *i* allocated to extra processor *r*, and 0 otherwise.
A^T	Binary variable equal to 1 if $T > SLA$, and 0 otherwise.
U^T	Critical path at time *T* for each file *i* included in the subset I'^T.
ES_i^T	Early start of file *i*.
LS_i^T	Late start of file *i*.
S_i^T	Slack (difference between early start and late start) of file *i*.
O_i^T	Binary variable equal to1 if $n_i - q_i^T > 0$, and 0 otherwise.

3.2.5 Algorithm steps and model formulation

The main steps of the DCSDBP algorithm are detailed as follows:

Given:

l_{ij}^T, n_i $\quad\forall\ i$ and $j(i \neq j)$

1. Preparatory and initialization stage:

- $I'^T = \{\}$
- Set $q_i^T = 0$ $\quad\forall\ i \in I$ (1)
- Set $P_k = 1$ $\quad\forall\ k \in K$ (2)
- Set $W_r = 1$ $\quad\forall\ r \in R$ (3)
- Set $V^T = 0$ (4)
- Set $T = 0$ (5)

2. Setup data file subset I'^T:

- If $\sum_{j=1}^{J} l_{ij}^T = 1$ $\quad\forall\ i$

  Then:

  - $f_i^T = 1$ (6)

– $I'^{T} = I'^{T} + \{i\}$ (7)

– Set $\alpha_i^T = \sum_{\theta=1}^{I} l_{\theta i}^T \quad \forall\ i$ (8)

3. Using the above mentioned assumptions and notations, the objective function can be written as follows:

$$\text{Min } (Z_T) = \sum_{i=1}^{I'}\sum_{k=1}^{K} \left(\frac{Csv + \left(\frac{Csf + Ch}{BW}\right)}{\beta i\ \alpha_i}\right) X_{iK}^{T} + \sum_{k=1}^{K} \left(\frac{Csf + Ch}{BW}\right)\left(1 - \sum_{i=1}^{I'} X_{iK}^{T}\right) +$$

$$\sum_{i=1}^{I'}\sum_{v=1}^{V} \left(\frac{Cesv + Cesf + Ceh}{\beta i\, \alpha_i}\right) Y_{ir}^{T} + \sum_{v=1}^{V} (Cesf + Ceh) \left(1 - \sum_{i=1}^{I'} Y_{ir}^{T}\right) +$$

$$A^T * Cp * (T\text{-}SLA). \tag{9}$$

The objective function (9) is to minimize data file allocation cost while taking into consideration priority, weight and criticality of each file included in each subset at any time *T*.

The first term considers cost, weight α_i and priority β_i at any time unit *T*. The second term considers the opportunity cost of not utilizing the basic processors at any time unit *T*. The third term handles the cost, weight α_i and priority β_i at any time unit *T*. The fourth term considers the opportunity cost of not utilizing the extra processors at any time *T*. the last term is concerned with the penalty cost of exceeding the *SLA*.

Subject to:

- **Basic and extra processors allocation and availability constraint.**

$$\sum_{k=1}^{K} X_{ik}^{T} + \sum_{r=1}^{V} Y_{ir}^{T} \le M_i f_i^T \qquad \forall\ i \in I'^{T}$$

where $M_i = min\{e_i, n_i - q_i^T, K+V\}$ (10)

Constraint (10) ensures that the total file allocation for any file *i* at a certain time *T* doesn't exceed neither file multiprocessing e_i nor file required remaining processing n_i nor total number of available basic and extra processors (*K* ,*V*).

- **Over processing constraint.**

$q_i^T \le n_i \qquad \forall\ i \in I$ (11)

Constraint (11) ensures that files are not being over processed i.e. processed more than required.

- **Single file allocation constraints.**

$$\sum_{i=1}^{I'} X_{ik}^{T} \leq P_{k}^{T} \qquad \forall\ k \in K \qquad (12)$$

$$\sum_{i=1}^{I'} Y_{ir}^{T} \leq W_{r}^{T} \qquad \forall\ r \in R \qquad (13)$$

Constraint (12) ensures that exactly one data file is allocated to a single basic processor.

Constraint (13) ensures that exactly one data file is allocated to a single extra processor.

- **Binary constraints.**

$$X_{ik}^{T} = 0 \quad or \quad 1 \qquad \forall\ i \in I \text{ and } k \in K \qquad (14)$$

$$Y_{ir}^{T} = 0 \quad or \quad 1 \qquad \forall\ i \in I \text{ and } r \in R \qquad (15)$$

Constraint (14) declares that the decision variable X_{ik} is binary, meaning that a file i either be assigned to basic processor or not.

Constraint (15) declares that the decision variable Y_{ir} is binary, meaning that a file i either be assigned to extra processor or not.

4. Update utilized extra processors:

- Calculate U^T = duration of the remaining critical path at time T [Subroutine]
- $ES_i^T = \text{Max}\ \{ES_j^T + (n_i - q_i^T)\}$ $\forall\ i$ and $j \in I$ where $(i \neq j)$ and $\alpha_i <= \alpha_j$
- $ES_i^T = 0$ $\forall\ i = 0$
- $LS_i^T = \text{Min}\ \{LS_j^T - (n_i - q_i^T)\}$ $\forall\ i$ and $j \in I$ where $(i \neq j)$ and $\alpha_i <= \alpha_j$
- $S_i^T = LS_i^T - ES_i$
- $U^T = \sum_{i=1}^{I} (n_i - q_i^T)$ $\forall\ i \in I$ and $S_i^T = 0$ (16)
- [End of Subroutine].
- If $U^T \geq SLA - T$

  Then:
  - $TC_p = C_p * (U^T - (SLA - T))$ (17)

  Else:
  - $TC_p = 0$ (18)
- If $U^T \geq SLA - T$

Then:

– $T_H = (C_{esf} + C_{eh} + C_{esv})^* U^T$ (19)

Else:

– $T_H = 0$ (20)

– $D^T = \sum_{i=1}^{I'} e_i - q_i^T \quad \forall\ i \in I'^T$ and $e_i > 1$ +

$\sum_{i=1}^{I'} e_i \quad \forall\ i \in I'^T$ and $e_i = 1$ (21)

– If $TC_p \geq T_H$ *and* $D^T > (K+V)$

Then:

– $V^T = V^T + 1$ (22)

– $T^r = T$ (23)

– $V^T \leq R$ (24)

5. Update f_i^T Matrix:

$-q_i^T = q_i^T + \sum_{k=1}^{K} X_{ik}^{T} + \sum_{r=1}^{R} Y_{ir}^{T} \quad \forall\ i \in I'^T$ (25)

– If $O_i^T = 1$

Then:

– $f_i^{T+\Delta} = 1$ (26)

Else:

– $f_i^{T+\Delta} = 0$ (27)

And:

– $l_{ji}^{T+\Delta} = 0 \quad \forall\ j$ (28)

– If $q_i^T = n_i$

Then:

– $\sum_{j=1}^{J} l_{ij}^{T+\Delta} = 0 \quad \forall\ i$ (29)

And:

– $\sum_{i=1}^{I} l_{ji}^{T+\Delta} = 0 \quad \forall\ j$ (30)

6. Check termination condition:

– If $\sum_{i=1}^{I} q_i^T = \sum_{i=1}^{I} n_i$ (31)

Then:

– Stop.

– If $T > SLA$

Then:

– $A^T=1$ (32)

Else

– $A^T=0$ (33)

Else

7. Update clock:

– $T= T+1$ (34)

Go to Step 2

Total cost (C) =Basic processor fixed hardware cost + Basic processor fixed software cost+ Basic processor variable software cost + Basic processor opportunity cost + Extra processor variable software cost +Extra processor fixed software cost + Extra processor hardware cost + Extra processor opportunity cost + penalty cost.

Total cost is calculated as follows:

$$TBC = \left(C_{sf} + C_h\right) * K + \sum_{T=1}^{END} \sum_{i=1}^{I'^T} \sum_{k=1}^{K} C_{sv} * X_{ik}^T$$

$$+ \sum_{T=1}^{BW} \sum_{i=1}^{I'^T} \sum_{k=1}^{K} \frac{\left(C_{sf} + C_h\right)}{BW} * (1 - X_{ik}^T)$$

$$+ \sum_{r=1}^{V} \left(C_{esf} + C_{eh}\right)(END - T^r) + \sum_{T=1}^{END} \sum_{i=1}^{I'^T} \sum_{r=1}^{V} C_{esv} * Y_{ir}^T$$

$$+ \sum_{r=1}^{V} \sum_{i=1}^{I'^T} \sum_{T=T^r}^{END} \left(C_{esf} + C_{eh}\right)(1 - Y_{ir}^T)$$

$$+ A^{END} * Cp * (T^{END} - SLA) \quad (35)$$

3.2.6 Illustrative example

The developed algorithm was tested using an example of a network with fifteen data files. Figure. 9 shows the used network which was generated using RanGen 2 [45]. The example was solved under the following assumptions:

- Basic resources=2 processors.
- Maximum available extra resources to be used in case of exceeding the *SLA*= 5 processors.
- Service Level Agreement, *SLA* = 18 time units.
- Batch Window, BW=22 time unit.
- Basic Processor Fixed Software Cost, *Csf* = \$ 10.
- Basic Processor Hardware Cost, *Ch* =\$ 100.
- Basic Processor Variable Software Cost per time unit, *Csv* = \$ 2.
- Extra Processor Fixed Software Cost, *Cesf* = \$ 12.5.
- Extra Processor Hardware Cost, *Ceh* = \$ 125.
- Extra Processor Variable Software Cost per time unit, *Cesv* = \$ 2.5
- Penalty Cost per time unit in case of exceeding the *SLA*, *Cp* = \$ 200

The client usually provides the network information in an excel file while the service provider prepares the required input figures and sets the lingo interface in order to run the program and determine the optimized solution. Table 2 represents the given initial precedence matrix needed to calculate data file's weight α_i and precedence parameter l_{ij}^T which are shown in Table 3. Files priorities β_i , processing time n_i , multiprocessing e_i are detailed in Table 4. These parameters depend on the type of network and are provided by the client before starting the data batch process.

After obtaining the required allocation result the final costs were calculated. An interface between Lingo and Excel was established to read the input information and to provide the output from the algorithm.

The results of running the program on the example are shown in Table 5. The files were processed in 20 time units, while 4 extra processors were rented, and the batch process exceeded the *SLA* by 2 time units, which indicates that penalty cost was imposed.

Once the batch process started, the program sets all initialization conditions. This means that ready files subset I'^T is empty. At the beginning of each time unit *T*,

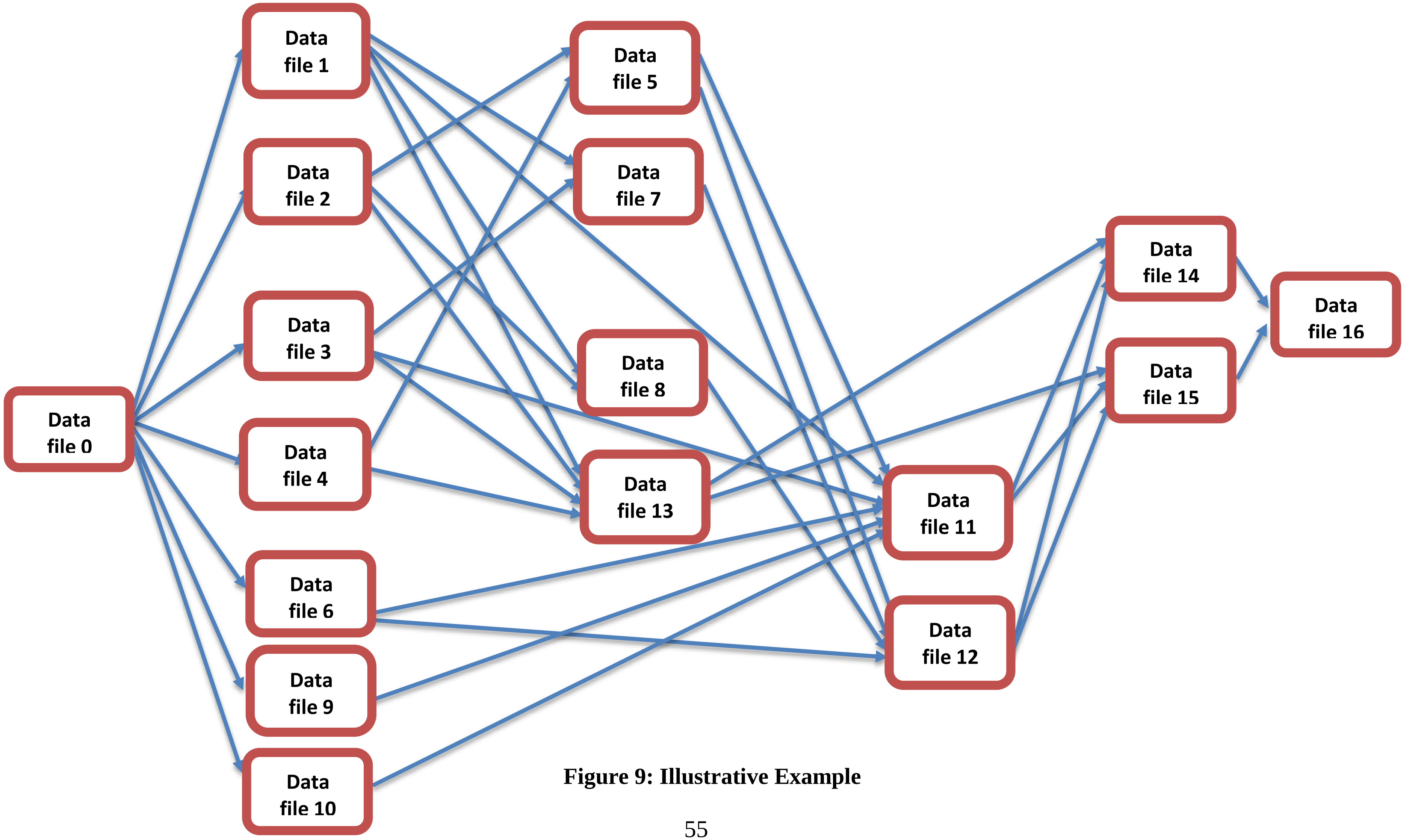

Figure 9: Illustrative Example

Table 2: Precedence Matrix for l_{ij} at T=0

L	j																
i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
11	0	1	0	1	0	1	1	0	0	1	1	1	0	0	0	0	0
12	0	0	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0
13	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Each data file's weight and precedence parameter is derived from Table 2. Table 3 shows the weights *αi* and precedence *Lij* for each file while clients given parameters are shown in Table 4.

Table 3: Data file initial weight *αi* and precedence parameter *Lij* at *T*=0

File	*αi*	*Lij*
0	1	1
1	6	1
2	5	1
3	5	1
4	4	1
5	4	3
6	4	1
7	3	3
8	3	3
9	3	1
10	3	1
11	4	7
12	4	5
13	4	5
14	2	4
15	2	4
16	1	1

Table 4: Files scheduling priority, multiprocessing e_i and processing time *ni* at *T*=0

File	*ei*	*ni*	*βi*
0	0	0	1
1	3	9	2
2	1	1	3
3	2	8	2
4	3	9	4
5	1	3	2
6	1	1	5
7	1	3	2
8	1	8	2
9	2	4	2
10	2	8	2
11	1	1	1
12	6	12	1
13	3	9	1
14	1	2	2
15	1	4	2
16	0	0	1

Table 5: Allocation result for Example 1.

T	Files														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0				k:2		k:1									
1				k:1, 2 R:1											
2		k:2		k:1 R: 1,2											
3	k:1, 2 R:3			R:1, 2											
4	R:1, 3, 4		k:1 R:2		k:2										
5	R:1, 3, 4		k:1 R:2		k:2										
6			R:1, 2		R:4			k:2	k:1 R:3						
7			R:1, 4					R:2	R:3	k:1, 2					
8							R:3	R:2	R:4	k:1 R:1			k:2		
9							R:1	R:2		R:3, 4			k:1 ,2		
10							R:1	R:2		R:3, 4			k:1 ,2		
11								R:1			R:2		k:1 ,2 R:3		
12								k:2					k:1		
13								k:2							
14												k:1, 2 R:1, 2, 3,4			
15												k:1, 2 R:1, 2, 3,4			
16														k:2	k:1
17														k:2	k:1
18															k:2
19															k:2

the precedence parameter l_{ij}^{T} for all files is checked to determine which files are ready to be processed at that time unit. All files having precedence parameter l_{ij}^{T}=1 are considered ready and inserted to the ready files subset I'^{T}.

Files 1, 2, 3,4,6,9 and 10 were ready for processing at *T*=0 therefore they were inserted into the ready files subset I'^{T}. At *T*=0 files 4 and 6 were processed by basic processor 2 and 1 respectively because these files have the highest calculated weight αi and predetermined priority β_i while the rest of ready files were shifted for the next time unit. Also since file 6 needed only one processing time unit n_i then by the end of the first iteration this file is considered ready.

At *T*=1, the values of precedence parameter l_{ij}^{T} are updated for all files, so the ones that haven't been processed or not finished processing still have the value of 1 and are consequently still included in the ready files subset I'^{T}while file 6 which has a processing time n_i of 1 has the value of l_{ij}^{T} =0 and no longer exist in the subset I'^{T}since it is considered ready. Also an extra processor $V^{T}$was utilized at that time unit because the criteria of renting a new extra processor was satisfied. It was found that the critical path of the remaining network activities exceeds the *SLA*, so the program needs to take an action to try to avoid the delay. Since the cost of renting a new processor T_H was found to be less than the penalty cost TC_p at that time unit and also since there are ready files for the next time unit that exceeds the total number of available processor then the decision of renting a new extra processor was taken. During *T*=1 file 4 was processed by basic processors 1 and 2 as well as by extra processor 1.

At *T*=2 another extra processor was rented based on the above explained mechanism. File 4 continued to be processed by basic processors 1 and extra processors 1 and 2. It is worth mentioning that file 4 has a multiprocessing of *ei* =3. Also file 2 was processed by basic processor 2.

At *T*=3 a third extra processor was rented and file 1 was processed by basic processors 1 and 2 and extra processor 3 according to it's multiprocessing criteria. File 4 was processed by extra processors 1 and 2 and by that it is considered ready.

At *T*=4 the forth extra processor was rented and used to process file 1 along with extra processors 1 and 3 while extra processor 2 and basic processor 1 were used to process file 3 . File 5 was processed by basic processor 2.

T=5 had the same files allocations as *T*= 4. It is noted that the program didn't rent any more extra processor starting from *T*=5 onwards which means it utilized 4 out of the 5 available extra processors and that is of course based on the renting mechanism.

At *T*=6 files 3, 5, 8 and 9 were processed by the available basic and extra processors as demonstrated in Table 5. For the rest of time units till time unit *T*=19 and files till file 15, Table 5 provides a clear allocations result.

It is worth mentioning that files 0 and 16 are start and end files with processing time n_i =0 which means they won't be allocated to any processor. The reason of their existence is to start and end the network for critical path calculation purposes.

The batch process cost calculation was based on applying the input figures mentioned at the beginning of this section and the allocation results shown in Table 5 in equation (35) of the mathematical model. Table 6 demonstrates the detailed costs.

Table 6 clarifies the cost of allocating files to basic processor and that includes hardware and software fixed costs as well as software variable cost. Similarly, extra processors allocating cost which includes hardware and software fixed costs as well as software variable cost is shown. Then the opportunity costs associated with each processor type is calculated. Penalty cost is determined at the end of the batch process and total cost is found by summing all costs for each time unit.

In Table 6 column (1) represents the time unit *T*, Column (2) shows the total number of existing basic processors *K*. However, Column (3) identifies how many basic processors are actually being acquired at each time unit *T*, column (4) calculates the basic variable cost C_{sv} while the fixed software C_{sf} and hardware C_h costs are calculated at the end because they are not related to time .Basic opportunity cost is determined in column (5) as per equation (35). For extra processor, column (6) shows number of extra processors V^T rented at that time unit *T*, Column (7) represents the number of extra processors actually being acquired at each time unit *T*. In column (8) and based on model assumption 14 which states that Extra processors are paid for from the time unit *T* they are rented onwards, extra processors total allocation cost is calculated using all types of costs(extra fixed hardware cost C_{eh}, extra fixed software cost C_{esf} and extra variable software cost C_{esv}). It is worth mentioning that in case of an extra processor being rented but not utilized due to unavailability of ready files, the total allocation cost will equal extra fixed hardware cost C_{eh} plus extra fixed software cost C_{esf} while extra variable software cost C_{esv} will not exist since the processor is not being utilized. Extra processor opportunity cost is found in column (9) using extra fixed hardware cost C_{eh} plus extra fixed software cost C_{esf} . Finally column (10) sums all the above costs for each time unit T .

At the end of Table 6 basic fixed costs are added, they represent fixed hardware cost C_h plus fixed software cost C_{sf} for each basic processor k . Also as mentioned above penalty cost is calculated based on number time units the processing is delayed beyond the *SLA*. Since basic fixed hardware cost C_h and fixed software cost C_{sf} are calculated per unit time by dividing them by BW , remaining opportunity cost for basic processor exists in case the batch process time END is less than batch window BW . It represents the opportunity cost of the basic processors for the time units between end of batch process *END* and *BW*.

From Table 6 we can see that from *T*= 0 till *T*=17, the basic allocation cost was showing the utilization of both basic processors, which explains the zero basic processors opportunity cost during the same period. However, at *T*=18 and 19 one basic processor was utilized and that ended up in variable allocation cost for one processor and opportunity cost for the other.

It is noticed that from *T*=0 up to *T*=10, every rented extra processor was utilized which ended in zero extra processor opportunity cost. After *T*=10 some rented extra processors were not utilized due to unavailability of ready files such as at *T*=11 where 3 out of 4 rented extra processors were utilized. Also at *T*=12,13,16,17,18 and 19 non of the rented extra processors were utilized due to unavailability of ready files.

The above illustrative example proves the effectiveness of the developed optimization algorithm in allocating files to processors. It managed to allocate highly prioritized and weighted files before the ones with lower priority and weight. Also, the algorithm will advise renting the necessary number of extra processors to accomplish the batch process goal while trying to minimize all types of costs. In the illustrated example, the program rented 4 extra processors to achieve minimum real batch process time which is 20 time units. Although that exceeds the *SLA* specified time of 18 time units, it is the minimum possible execution time for this network due to network logic and predecessors’ relations.

Table 6: Data Batch Process Cost for Example 1.

(1) T	(2) Basic processors	(3) No. of Acquired Basic processors	(4) Basic Allocation Variable Cost $	(5) Basic Oppo. Cost $	(6) Rented Extra Processors	(7) No. of Acquired Extra Processors	(8) Extra Total Allocation Cost $	(9) Extra Oppo. Cost $	(10) Total Cost per time period $
0	2	2	4.00	0.00	0	0	0.00	0.00	4.00
1	2	2	4.00	0.00	1	1	8.75	0.00	12.75
2	2	2	4.00	0.00	2	2	17.50	0.00	21.50
3	2	2	4.00	0.00	3	3	26.25	0.00	30.25
4	2	2	4.00	0.00	4	4	35.00	0.00	39.00
5	2	2	4.00	0.00	4	4	35.00	0.00	39.00
6	2	2	4.00	0.00	4	4	35.00	0.00	39.00
7	2	2	4.00	0.00	4	4	35.00	0.00	39.00
8	2	2	4.00	0.00	4	4	35.00	0.00	39.00
9	2	2	4.00	0.00	4	4	35.00	0.00	39.00
10	2	2	4.00	0.00	4	4	35.00	0.00	39.00
11	2	2	4.00	0.00	4	3	32.50	6.25	42.75
12	2	2	4.00	0.00	4	0	25.00	25.00	54.00
13	2	1	2.00	5.00	4	0	25.00	25.00	57.00
14	2	2	4.00	0.00	4	4	35.00	0.00	39.00
15	2	2	4.00	0.00	4	4	35.00	0.00	39.00
16	2	2	4.00	0.00	4	0	25.00	25.00	54.00
17	2	2	4.00	0.00	4	0	25.00	25.00	54.00
18	2	1	2.00	5.00	4	0	25.00	25.00	57.00
19	2	1	2.00	5.00	4	0	25.00	25.00	57.00
									795.25

Penalty cost $	**400.00**
Basic processors fixed costs $	**220.00**
Remaining opportunity cost for basic processors $	**20.00**
Batch Process Total Cost $	**1,435.25**

3.2.7 Benchmark

The same example was ran by the Batch data Processes Scheduling (BDPS) algorithm presented in Osman *et al*. The results showed that files were processed in 41 time units, and the batch process exceeded the *SLA* by 23 time units, which indicates that penalty cost was imposed. Also the total cost of the data batch process was $ 5,184.00.

Comparing the results of running the same example by the two algorithms indicates that the Dynamic Cost scheduling for Data Batch Process (DCSDBP) algorithm developed in this research performed the data batch process in a shorter time and achieved a lower total batch process cost .

Chapter 4: Sensitivity Analysis

The sensitivity analysis performed on the developed model will be presented in this chapter and the results will be analyzed. Sensitivity analysis was performed in two parts. The first part was conducted on the illustrative example network where different parameters were changed and the result of running the program was demonstrated. However, in part two of the sensitivity different networks with varying sizes and complexities were analyzed to study the performance of the developed algorithm on them.

4.1 Single Network Analysis

4.1.1 Varying number of available extra processors

Five extra processors were available to be rented in the illustrative example in case of exceeding the *SLA*. Part of the proposed algorithm was deciding how many of those processors to rent to minimize total cost and batch total processing time while satisfying files priorities and predecessors' relations. After running the program it was found that four out of five extra processors were rented to finish the batch process.

In this part of the sensitivity analysis, different values of the number of available extra processor will be tested and the results of running the program in each case will be presented to show how many extra processors the program decides to rent.

- In case of having four extra processors available to be used in the batch process, the results were:

Table 7: Batch process results in case of four Available extra processors

Dimension	result	unit
Batch time	20	time units
Program run time	3	seconds
Extra resources	4	servers
Total batch process cost	1,435.25	$
SLA deadline	Exceeded by 2	time units

It is noted from the above table that the results are the same as the case of 5 available extra processors since the same number of extra servers were rented.

- The results of having three available extra processors in the batch process:

Table 8: Batch process results in case of three Available extra processors

Dimension	result	unit
Batch time	21	time units
Program run time	4	seconds
Extra resources	3	servers
Total batch process cost	1,475.00	$
SLA deadline	Exceeded by 3	time units

In this case less processors were utilized which ended up in longer batch process time and higher total cost. Also, *SLA* was exceeded by more time units.

- In case of having two extra processors available to be used in the batch process, the results were:

Table 9: Batch process results in case of two Available extra processors

Dimension	result	unit
Batch time	24	time units
Program run time	4	seconds
Extra resources	2	servers
Total batch process cost	1,952.00	$
SLA deadline	Exceeded by 6	time units

In this case, minimizing utilized processors led to more delay in batch process time and increased total cost. *SLA* was also exceeded by 6 time units.

- The results of having one available extra processor are:

Table 10: Batch process results in case of one Available extra processors

Dimension	result	unit
Batch time	30	time units
Program run time	6	seconds
Extra resources	1	servers
Total batch process cost	3,014.75	$
SLA deadline	Exceeded by 12	time units

In the last case of having only one extra processor, the batch process had the highest delay and total cost. From the above it is concluded that the program rents extra processors only when needed to perform the batch process with minimum execution time and total cost.

Figure 10 and Figure 11 are a graphical summery of the above findings with respect to batch processing time and total cost respectively.

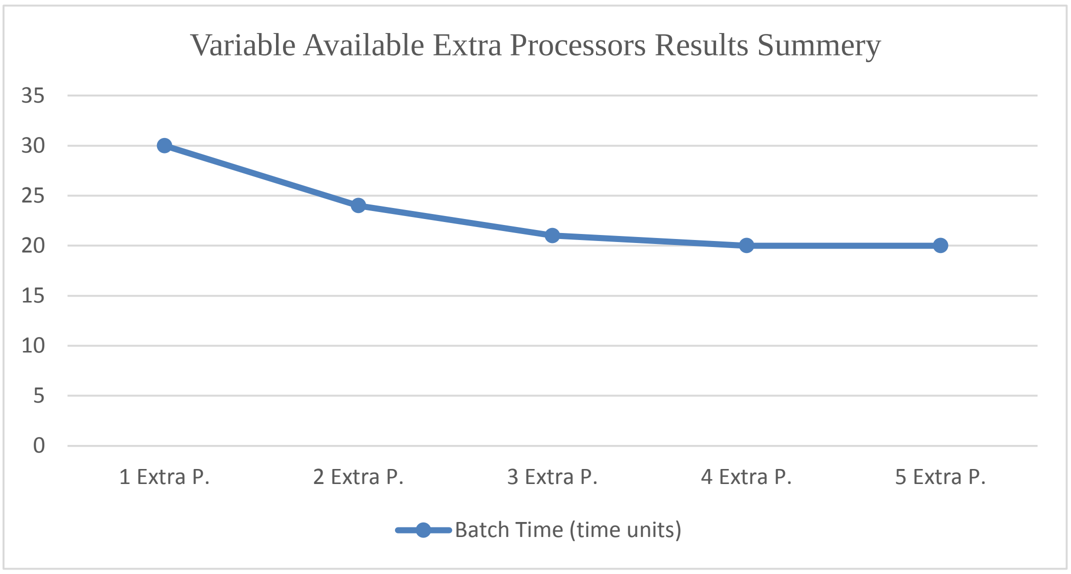

Figure 10: Varying available extra processors vs. Batch time

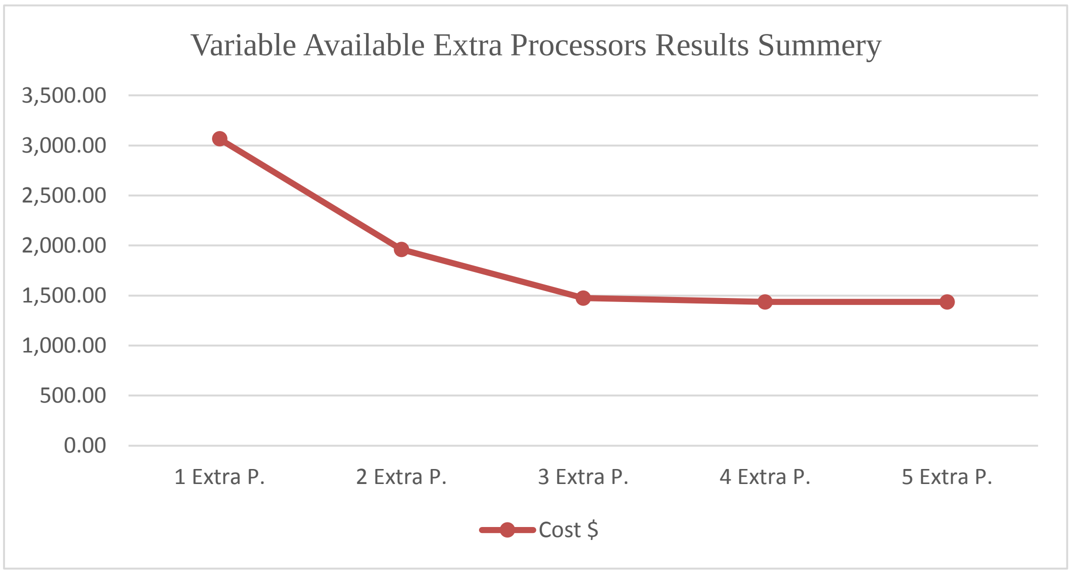

Figure 11: Varying available extra processors vs. Cost

4.1.2 Changing *SLA* value

Originally it was assumed that the *SLA* is 18 time units but the program couldn't finish allocating files to available resources before the 20th time unit. This is due to network logic and predecessors relation. That is why although an extra fifth server was available, the program didn't rent it because that will not minimize the total batch process time in order to meet the specified *SLA*. In the following, different *SLA* values were assumed while other parameters kept the same:

- Assuming the *SLA* value equals to 19 time units, the results were:

Table 11: Batch process results in case of *SLA* = 19 time units

Dimension	result	unit
Batch time	20	time units
Program run time	3	seconds
Extra resources	4	servers
Total batch process cost	1,235.25	$
SLA deadline	Exceeded by 1	time units

Clearly when increasing the value of *SLA*, the result will be lower penalty cost leading to lower total cost.

- Assuming the *SLA* value equals to 17 time units, the results were:

Table 12: Batch process results in case of *SLA* = 17 time units

Dimension	result	unit
Batch time	20	time units
Program run time	3	seconds
Extra resources	4	servers
Total batch process cost	1,635.25	$
SLA deadline	Exceeded by 3	time units

Clearly when decreasing the value of *SLA* while the network can't be processed in a shorter time due to network precedence relations, the result will be higher penalty cost leading to higher total cost.

- Assuming the *SLA* value equals to 16 time units, the results are shown in Table 13:

When *SLA* was further decreased, penalty cost and consequentially total cost increased while batch process time remained the same.

Table 13: Batch process results in case of *SLA* = 16 time units

Dimension	**result**	**unit**
Batch time	20	time units
Program run time	3	seconds
Extra resources	4	servers
Total batch process cost	1,835.25	$
SLA deadline	Exceeded by 4	time units

- Assuming the *SLA* value equals to 20 time units, the results were:

Table 14: Batch process results in case of *SLA* = 20 time units

Dimension	**result**	**unit**
Batch time	20	time units
Program run time	3	seconds
Extra resources	4	servers
Total batch process cost	1,035.25	$
SLA deadline	met	-

The effect of the *SLA* is clear in this case where the deadline is met and no penalty cost is imposed. This led to the lowest total batch process cost. It is recommended when deciding on the *SLA* time between the client and the service provider that both sides study the data batch network carefully to decide on the right *SLA* terms that serve both sides and achieve the goal of the batch process.

Figure 12 and Figure 13 are a graphical summery of the above findings with respect to batch processing time and total cost respectively.

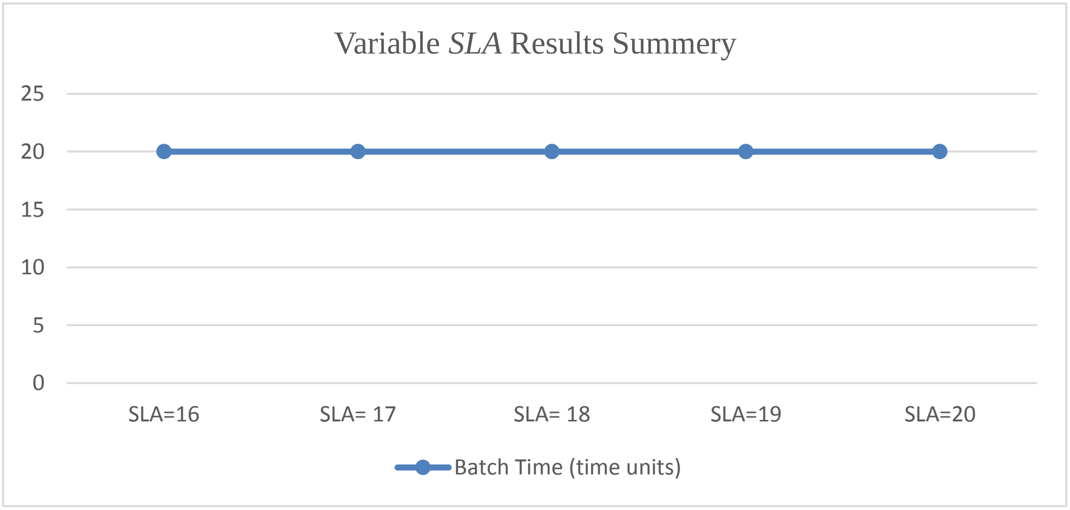

Figure 12: Varying *SLA* vs. Batch time

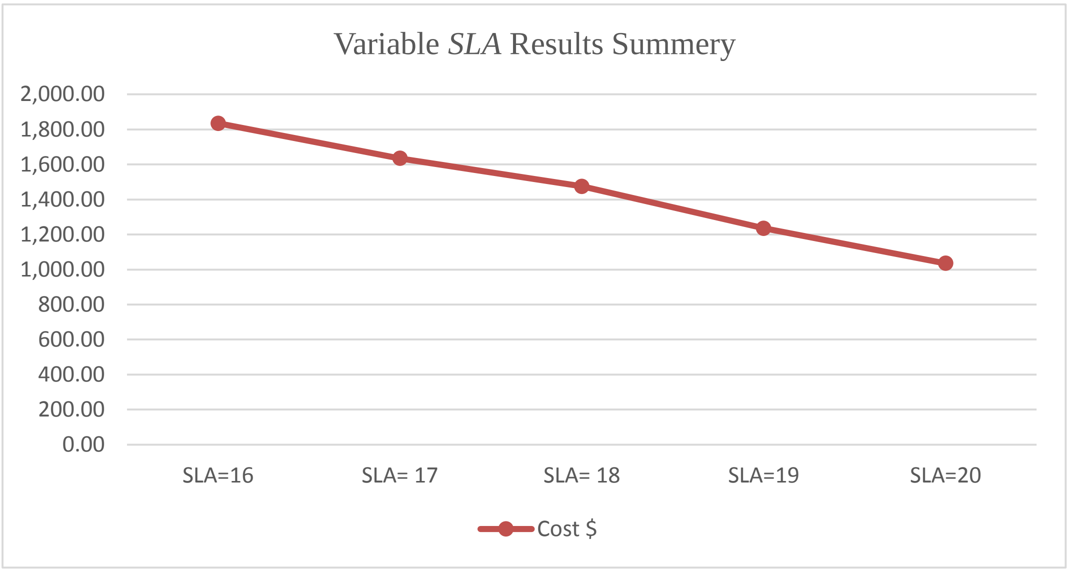

Figure 13: Varying *SLA* vs. Cost

4.1.3 Varying penalty cost per time unit

Penalty cost of $200 per unit time of delay was assumed in the illustrative example. To study the effect of penalty cost factor on the batch process results, different penalty cost values were assumed and the algorithm was tested accordingly.

- Set penalty cost to zero, the results were:

Table 15: Batch process results in case of *Cp* = $ 0 per time unit

Dimension	result	unit
Batch time	41	time units
Program run time	8	seconds
Extra resources	0	servers
Total batch process cost	384	$
SLA deadline	Exceeded by 20	time units

The lowest batch process cost is obviously obtained when there is no penalty cost. In this case there won't be any trade-off between renting extra processor cost and any other cost. Since one of the goals of this algorithm is minimizing cost, then it will choose not to rent any extra processors because in this way, the delay in processing the data files has no price and that is clearly shown in the low total batch cost. Also the program needed more time to run since it's batch process time was doubled.

- Set penalty cost to $ 5 per unit time, the results were:

Table 16: Batch process results in case of *Cp* = $ 5 per time unit

Dimension	result	unit
Batch time	41	time units
Program run time	8	seconds
Extra resources	0	servers
Total batch process cost	504	$
SLA deadline	Exceeded by 20	time units

The program chooses not to rent any extra processors. This is exactly the same results of the case of zero processors but with a higher total cost since the penalty cost here is not only low but extremely low.

- Set penalty cost to $ 8 per unit time, the results were:

Table 17: Batch process results in case of *Cp* = $ 8 per time unit

Dimension	result	unit
Batch time	36	time units
Program run time	8	seconds
Extra resources	3	servers
Total batch process cost	1,210.00	$
SLA deadline	Exceeded by 16	time units

In this case, the program chose to rent three extra processors instead of four and extend the total batch process time and pay more penalty cost. However, the total cost is less. This is due to the new penalty cost used in the trade –off decision mechanism in renting extra processor.

- When assuming penalty cost per time unit =$ 20, the results were:

Table 18: Batch process results in case of *Cp* = $ 20 per unit time

Dimension	result	unit
Batch time	20	time units
Program run time	8	seconds
Extra resources	4	servers
Total batch process cost	1,075.25	$
SLA deadline	Exceeded by 2	time units

In this case the same number of extra processors were rented as of the case of Penalty cost per unit time = $ 200 but with a less total bath cost.

- Set penalty cost to $ 100 per unit time, the results were:

Table 19: Batch process results in case of *Cp* = $ 100 per unit time

Dimension	result	unit
Batch time	20	time units
Program run time	4	seconds
Extra resources	4	servers
Total batch process cost	1,235.25	$
SLA deadline	Exceeded by 2	time units

In this case the same number of extra processors were rented as of the case of Penalty cost per unit time = $ 200 but with a less total bath cost.

- Set penalty cost to $ 500 per unit time, the results were:

Table 20: Batch process results in case of *Cp* = $ 500 per unit time

Dimension	result	unit
Batch time	20	time units
Program run time	3	seconds
Extra resources	4	servers
Total batch process cost	2,035.25	$
SLA deadline	Exceeded by 2	time units

Increasing the penalty cost further will not affect the way files are allocated to servers since the program won't be able to squeeze the batch time more even if the trade-off between renting more extra processors and penalty cost goes in favor of utilizing another extra server simply because of the network's activity relations.

Figure 14 and Figure 15 are a graphical summery of the above findings with respect to batch processing time and total cost respectively.

4.1.4 Changing the extra server costs

The extra sever has three types of costs as previously explained. It is assumed in our model that they represent 1.25 of the basic server's costs. In the following analysis, the effect of various extra to basic servers' costs will be demonstrated.

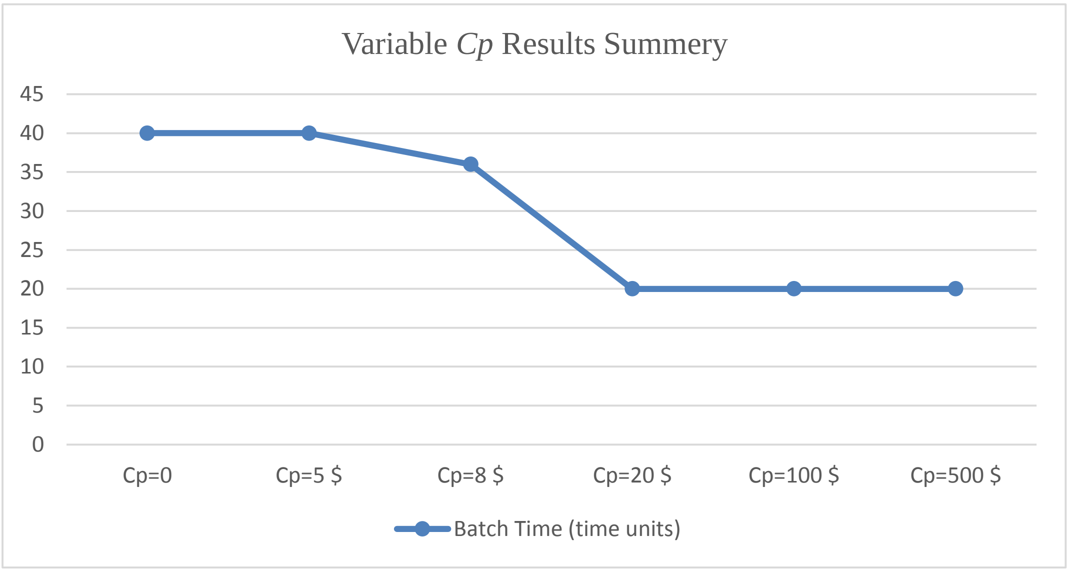

Figure 14: Varying *Cp* vs. Batch time

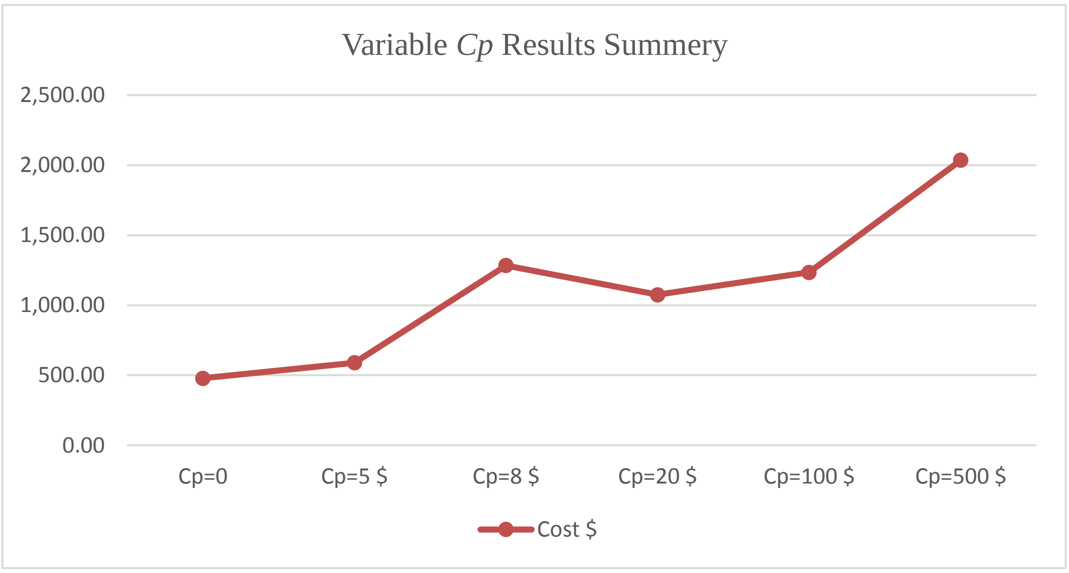

Figure 15: Varying *Cp* vs. Cost

- Set costs ratio to 2.5, the results were:

Table 21: Batch process results in case of extra to basic servers cost ratio = 2.5

Dimension	result	unit
Batch time	20	time units
Program run time	4	seconds
Extra resources	4	servers
Total batch process cost	2,141.50	$
SLA deadline	Exceeded by 2	time units

When increasing the renting cost of extra processor to 2.5 times the basic leasing cost, the program rented the same number of extra servers to finish the batch process because the penalty cost is still higher relatively. It is only ended up in increasing the total batch cost while same number of extra servers were utilized.

- Set costs ratio to 5, the results were:

Table 22: Batch process results in case of extra to basic servers cost ratio = 5

Dimension	**result**	**unit**
Batch time	20	time units
Program run time	4	seconds
Extra resources	4	servers
Total batch process cost	3,554.00	$
SLA deadline	Exceeded by 2	time units

Increasing the renting cost of extra processor to 5 times the basic leasing cost didn't stop the program from renting extra servers to finish the batch process because the penalty cost is still higher relatively. However, the total batch cost increased while same number of extra servers were utilized.

- Set costs ratio to 10, the results were:

Table 23: Batch process results in case of extra to basic servers cost ratio = 10

Dimension	**result**	**unit**
Batch time	20	time units
Program run time	5	seconds
Extra resources	4	servers
Total batch process cost	6,379.00	$
SLA deadline	Exceeded by 2	time units

Same results were obtained in this case as the case of the cost factor =5, in which it is still cheaper to rent extra servers than paying penalty cost.

- Set costs ratio to 1, the results are show in Table 24:

To prove the efficiency of the program, the case of equal costs for basic and extra servers was tested, the program rented only what it needed from available extra servers to finish the batch process.

Table 24: Batch process results in case of extra to basic servers cost ratio = 1

Dimension	result	unit
Batch time	20	time units
Program run time	5	seconds
Extra resources	4	servers
Total batch process cost	1,284.00	$
SLA deadline	Exceeded by 2	time units

- Set costs ratio to 0.5, the results were:

Table 25: Batch process results in case of extra to basic servers cost ratio = 0.5

Dimension	result	unit
Batch time	20	time units
Program run time	5	seconds
Extra resources	4	servers
Total batch process cost	1,001.50	$
SLA deadline	Exceeded by 2	time units

Here the total price went lower but the files to servers' allocation stayed the same. The above findings and analysis prove that when the program reaches the optimum solution, it doesn't utilize any unnecessary resources since it is part of the objective function to minimize all types of costs while trying to meet *SLA* deadline and satisfy all priorities and constraints. This proves that the purpose of developing this model is served to a high extent and the batch process is being utilized in a more efficient way. Figure 16 and Figure 17 are a graphical summery of the above findings with respect to batch processing time and total cost respectively.

4.2 Multiple Networks Analysis

In this section, we tested the developed algorithm by running networks with different sizes and complexities generated by RanGen2 [45]. Four networks' sizes were selected; each with three different complexities and the run time for each network in a group was recorded.

One indicator of complexity is I2 index. It is a measure of the closeness of a network to a serial or parallel graph based on the number of progressive levels. If I2=0 then the network activities are all in parallel, while if I2=1 then the network activities are all in series. [46]

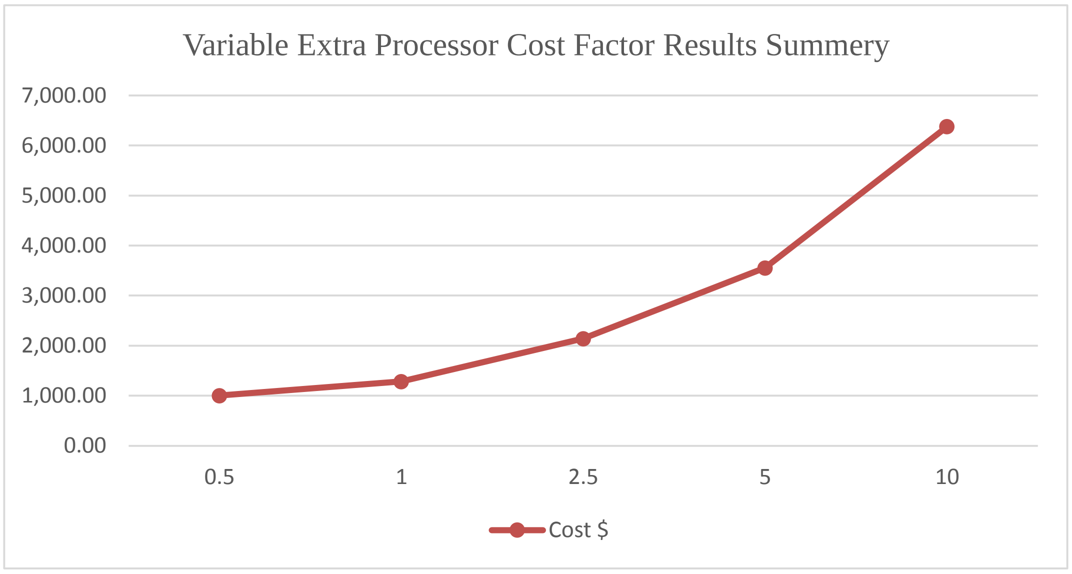

Figure 16: Varying extra processor cost factor vs. Batch time

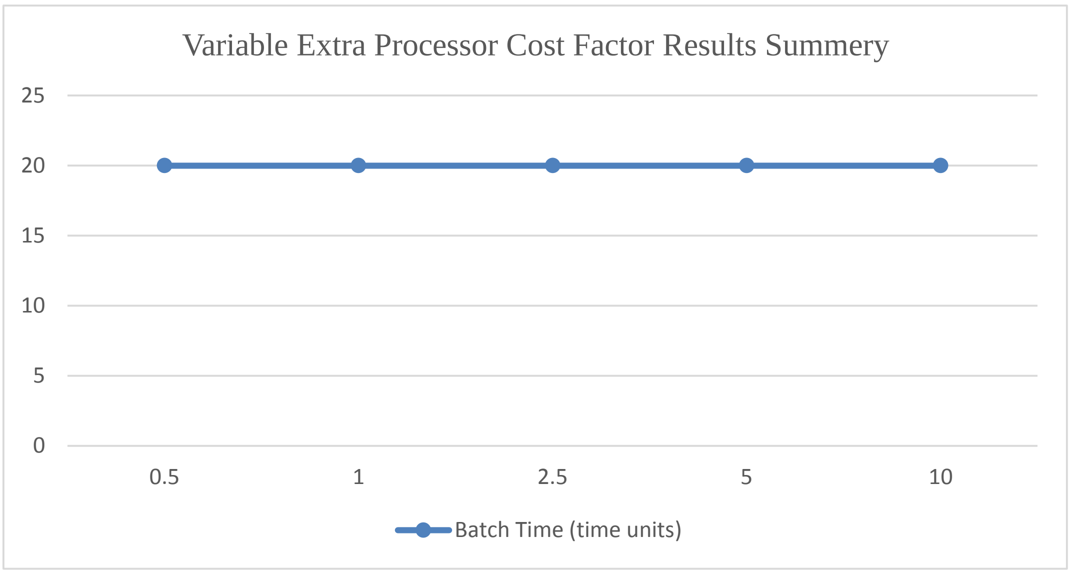

Figure 17: Varying extra processor cost factor vs. Cost

4.2.1 Fifteen activities networks

The average program run time for each network is shown in the below Table 26

Table 26: Run times for fifteen activities network with different complexities

Complexity index I2	Run time (seconds)
0.2	2
0.5	3
0.8	3

4.2.2 Twenty-five activities networks

The average program run time for each network is shown in the below Table 27

Table 27: Run times for twenty-five activities network with different complexities

Complexity index I2	Run time (seconds)
0.2	5
0.5	2
0.8	4

4.2.3 Fifty activities networks

The average program run time for each network is shown in the below Table 28

Table 28: Run times for fifty activities network with different complexities

Complexity index I2	Run time (seconds)
0.2	11
0.5	13
0.8	14

4.2.4 Hundred activities networks

The average program run time for each network is shown in the below Table 29

Table 29: Run times for hundred activities network with different complexities

Complexity index I2	Run time (seconds)
0.2	33
0.5	34
0.8	34

It is clear from the above demonstration that the program run time is relatively low. However, it depends on the networks' sizes and complexity.

Chapter 5: Conclusion and Future Research

5.1 Conclusion

This research addressed the problem of data batch process optimization. The importance of data batch process and the wide applications of batch process, whether in data files or in general manufacturing and industries, made it essential to improve it's performance to the maximum possible level. This work aimed to include all aspects of data batch process in an iterative algorithm called DCSDBP. The developed algorithm proved its effectiveness in allocating data files to available resources while satisfying priority and predecessors constraints while maintaining the minimum possible cost keeping in mind the *SLA* time limit.

The five types of associated batch process costs which are basic hardware and software fixed and variable costs, extra hardware and software fixed and variable costs, and penalty cost were taken into consideration in the objective function. While the algorithm worked towards minimizing these costs, it aimed at the same time to allocate files based on their weight and priority while following the network predecessors' relations. At the same time the algorithm tried to commit to the time limit specified in *SLA*.

This work represents a major contribution to the literature since it includes all aspects of the data batch process. It was clearly demonstrated in the literature review chapter that non of the presented papers took batch process cost into consideration. And while most researchers dealt with one aspect of the data batch process, this research is filling the gap and considered an important reference in data batch process total optimization.

After coding the developed algorithm using Lingo, a number of networks were used to test it. It was concluded from the results that it is more effective to include all types of costs in one optimization model along with priority, weight, predecessor and time factors which led to a more effective allocation and a lower total batch process cost. The results showed that files with higher priority and weight get allocated before the ones with lower priority and weight, and when performing sensitivity analysis, the obtained allocation result had the lowest cost, bearing in mind maintaining other process related factors. It is also concluded that renting more extra processors doesn't necessary mean that the batch process will be performed in a shorter time because there are other factors that govern the process total time and these are the network logic relations or ‘predecessors’ relations’. The decision of whether or not to rent a new extra processor and when to do so is very important. It will affect the whole batch process in terms of file to processor allocation and total batch process cost and time. These processors are costly and if they are rented, they will be paid for till the end of the batch process, so the service provider has to be careful in negotiating their prices with the sole owner of these processors since these prices will affect when and how many extra processors the algorithm will rent based on the trade-off criteria developed within the algorithm. Another conclusion is related to the penalty cost since it is the other side of the trade -off criterion that governs the extra processor renting mechanism. That is why it has to be negotiated thoroughly, especially when looking at the batch process total cost and how much it is affected by the penalty cost per unit time delay agreed on in the *SLA*.

5.2 Limitations and future research direction

The model was developed under certain assumptions and while we tried to generalize it to cover most batch process networks, there might be a place for more research to include wider cases such as assuming different costs for basic processors or even different costs for extra processors. Also, future researchers might work on developing the extra processors renting mechanism to allow for more than one processor to be rented for each time unit in case that serves the total completion time and maintains a low cost.

One of our basic assumptions is that processors reservation is not allowed. This is another area for research where scholars can test the need for processor reservation, how it can be done, and what type of impact it has on the batch process different aspects such as total process time, total cost and basic and extra processors utilization.

References

[1]T. Jensen (2009) *Smarter Banking, Horizontal Integration, and Scalability* [online] available at: http://www.redbooks.ibm.com/redpapers/pdfs/redp4625.pdf [Accessed 1 April 2015].

[2]IBM Knowledge Center (2013) Mainframes Working After hours: *Batch processing* [online] available at:http://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zmainframe/zconc_batchproc.htm [Accessed 1 March 2015].

[3]IBM Knowledge Center (2010) *Roles in the mainframe world* [online] available at: http://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zmainframe/zconc_ITroles.htm?lang=en [Accessed 1 March 2015].

[4]Mainframe Migration Alliance (2006) *Batch Applications—The Hidden Asset* [online] available at:http://download.microsoft.com/download/4/1/d/41d2745f-031c-40d7-86ea-4cb3e9a84070/Batch%20The%20Hidden%20Asset.pdf [Accessed 1 March 2015].

[5]High Performance Computing Center North (HPC2N) *What is a batch system* [online] available at:https://www.hpc2n.umu.se/support/beginners_guide [Accessed 1 March 2015].

[6]IBM Knowledge Center (2010) *Batch and online transaction processing* [online] available at: http://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zmainframe/zconc_mfworklds.htm?lang=en [Accessed 3 March 2015].

[7]IBM Software Services (2009) *Designing Batch Applications* [online] available at: file:///C:/Users/Lenovo/Downloads/attachment_14548593_Designing_Batch_Applications.pdf [Accessed 20 March 2015].

[8]Business dictionary (2015) *Batch processing* [online] available at: http://www.businessdictionary.com/definition/batch-processing.html [Accessed 5 March 2015].

[9]I. Sommerville (2008) *Batch data processing systems* [online] available at: http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Architecture/AppArch/BatchDP.html [Accessed 5 March 2015].

[10]TransFirst,LLC (2015) *Batch or Batch Processing* [online] available at: http://www.transfirst.com/resources/glossary [Accessed 5 March 2015].

[11]Tutorialspoint (2015) *Batch operating system* [online] available at: http://www.tutorialspoint.com/operating_system/os_types.htm [Accessed 10 March 2015].

[12]"IBM 360/370/3090/390". Lars Poulsen (2015) *History of Operating Systems* [online] available at:http://www.beagle-ears.com/lars/engineer/comphist/ibm360.htm [Accessed 10 March 2015].

[13]J. Rehman (2012) *What are advantages and disadvantages of batch processing systems* [online] available at: http://www.itrelease.com/2012/12/what-are-advantages-and-disadvantages-of-batch-processing-systems/ [Accessed 20 March 2015].

[14]IBM Knowledge Center (2010) *Clustering technique: Parallel Sysplex* [online] available at:http://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zmainframe/zconc_clusterPlSys.htm?lang=en [Accessed 20 March 2015].

[15]G. Mca (2009) *Distributed and parallel processing* [online] available at: http://www.codeproject.com/Articles/35671/Distributed-and-Parallel-Processing- [Accessed 1 April 2015].

[16]Oracle (2015) *Batch failure and recovery* [online] available at: https://docs.oracle.com/html/E38272_01/Content/Batch%20Processor/Batch_failure_and_recovery.htm [Accessed 1 April 2015].

[17]M. Osman, M. Ndiaye and A. Shamayleh. "Dynamic scheduling for batch data processing in parallel systems," *International Conference on Operations Research and Enterprise Systems,* 2014.

[18]A. Page, T. Keane and T. Naughton. "Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system," *Journal of Parallel and Distributed Computing* 70:758-766, 2010.

[19]C. Mendez, J. Cerda, I. Grossmann. "State-of-the-art review of optimization methods for short-term scheduling of batch processes," *Computers and Chemical Engineering.* 30:913-946, 2006.

[20]S. Lim and S. Cho. "Intelligent os process scheduling using fuzzy inference with user models,*" Lecture Notes in Computer Science* 4570:725-734, 2007.

[21]F. Xhafa and A. Abraham. "Computational models and heuristic methods for Grid scheduling problems," *Future Generation Computer Systems* 26:608-621, 2010.

[22]K. Aida. "Effect of job size characteristics on Job scheduling performance," *Lecture Notes in Computer Science* Volume, 1911, pp 1-17, 2000.

[23]I. Stoica, H. Abdel-Wahab and A. Pothen. "A Microeconomic Scheduler for Parallel Computers," *Institute for Computer Applications in Science and Engineering (ICASE),* 1995.

[24]I. Stoica and A. Pothen. "A Robust and Flexible Microeconomic Scheduler for Parallel Computers," *3rd International Conference on High Performance Computing*, 1996.

[25]A. Agarwal and P. Kumar. "Multidimensional QOS Oriented Task Scheduling in Grid Environments," *International Journal of Grid Computing & Applications (IJGCA)* Vol.2, No.1, March 2011.

[26]D. Andresen and T. McCune. “Towards a Hierarchical Scheduling System for Distributed WWW Processor Clusters,” *The Seventh International Symposium on High Performance Distributed Computing*, 1998.

[27]M. Islam, G. Kanna and P. Sadayappan. “Revenue Maximization in Market-Based Parallel Job Schedulers,” Ohio State University Library, 2008.

[28]B. Srinivasan, S. Palanki and D. Bonvin. “Dynamic optimization of batch processes, I. Characterization of the nominal solution,” *Computers and Chemical Engineering*, 27 1 _/26, 2002.

[29]B. Srinivasan, S. Palanki, E. visser and D. Bonvin. “Dynamic optimization of batch processes II. Role of measurements in handling uncertainty,” *Computers and Chemical Engineering,* 27_/44, 2002.

[30]R. Zhou, L. Li, W. Xiao and H. Dong. “Simultaneous Optimization of batch process schedules and water – allocation network,” *Computers and Chemical Engineering,* 33 (2009) 1153–1168, 2008.

[31]P. Damodaran and M. Vélez-Gallego. “A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times,” *Expert Systems with Applications*, 39:1451-145, 2012.

[32]M. Mehta, V. Soloviev and D. DeWitt. “Batch scheduling in parallel database systems,” *9th International Conference on Data Engineering* 400-410, 1993.

[33]A. Grigoriev, M. Sviridenko and M. Uetz. “Unrelated parallel machine scheduling with resource dependent processing times,” *METEOR Research Memorandum*, Vol. 033, 2005.

[34]L. Bouganim, F. Fabret, C. Mohan and P. Valduriez. “Dynamic query scheduling in data integration systems,” *Proceedings of 16th International Conference on Data Engineering*, (Cat. No.00CB37073), 2000.

[35]J. Ngubiri and M. Vliet. “A Metric of Fairness for Parallel Job Schedulers,” *Concurrency Computat.:Pract. Exper*. 2008; 05:1–7 Prepared using cpeauth.cls [Version: 2002/09/19 v2.02], 2008.

[36]A. Arpaci-Dusseau and D. Culler. “Extending Proportional-Share Scheduling to a Network of Workstations,” *Conference of Parallel and Distributed Processing Techniques and Applications*, 1997.

[37]D. Ferguson, C. Nikolaou, J. Sairamesh and Y. Yemini. “Economic Models for Allocating Resources in Computer Systems,” *Market-based control: a paradigm for distributed resource allocation,* P. 156-183, 1996.

[38]K. Kuwabara, Y. Nishibe, T. Ishida and T. Suda. “An Equilibratory Market- Based Approach for Distributed Resource Allocation and Its Applications to Communication Network Control,” *Market-based control: a paradigm for distributed resource allocation*, P. 53-73, 1996

[39]Chun and Culler. “User-centric Performance Analysis of Market-based Cluster Batch Schedulers,” *CCGRID '02 Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid* , P. 30, 2002.

[40]J. Sairamesh, D. Ferguson and Y. Yemini . “An Approach to Pricing, Optimal Allocation and Quality of Service Provisioning In High-Speed Packet Networks,” proceedings of the INFOCOM, 1995.

[41]C. Yeo and R. Buyya . “A taxonomy of market-based resource management systems for utility-driven cluster computing,” *Journal Software—Practice & Experience*, Volume 36 Issue 13, P.1381-1419, 2006.

[42]M. Islam, P. Balaji, G. Sabin and P. Sadayappan. “Analyzing and Minimizing the Impact of Opportunity Cost in QoS-aware Job Scheduling,” *International Conference on Parallel Processing*, 2007.

[43]A. Mutz and R. Wolski. “Efficient Auction-Based Grid Reservations using Dynamic Programming,” *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, Article No.16, 2008.

[44]A. Mutz , R. Wolski and J. Brevik. “Eliciting Honest Value Information in a Batch-Queue Environment,” *8th IEEE/ACM International Conference on Grid Computing*, 2007.

[45]M. Vanhoucke, J. Coelho, D. Debels, B. Maenhout, and L.V. Tavares. “An evaluation of the adequacy of project network generators with systematically sampled networks”, *European Journal of Operational Research*, Volume 187, Issue 2, 2008.

[46]J. P. T. Higgins, S. G. Thompson, J. J. Deeks and D. G. Altman. “Measuring inconsistency in meta-analyses,” *BMJ : British Medical Journal*, Volume 327, Issue 7414, 2003.

Appendix A

Lingo Code

```
MODEL:
SETS:
clock/z1..z40/;!time loop;
file:m,f,e,n,q,remain,alpha,beta,Lij,O,ES, LS,
SLACK,EF,LF,d;!define all file related parameters;
nestedfile(file,file):rowcol;!files readiness parameter
matrix;
processor:P;!P=1 if k is available otherwise 0;
extra:W,V;!w=1 if R is available otherwise 0;
assign1(file,processor):X;!allocating files to basic
processors;
assign2(file,extra):Y;!allocating files to extra processors;
PRED( file, file);!predecessors matrix;
ENDSETS
DATA:

file,PRED,n,e,beta,processor,extra,rowcol,maxtime,SLA,BW,csf,c
h,Csv,Cesv,Cesf,Ceh,Cp@=OLE')C\:Users\Lenovo\Desktop\lingo
programs 2,'2\
'file','PRED','n','e','beta','processor','extra','rowcol','max
time','SLA','BW','Csf','Ch','Csv','Cesv','Cesf','Ceh','Cp;('
!establish the excel link to import input data;
ENDDATA
submodel main:
;!If AP =1 then data batch process exceeded SLA time and
penelty cost is imposed,  0 otherwise;

Min=@sum(assign1(I,K)|lij(i)#eq#1:((Csv+((Csf+Ch)/Bw))/(alpha(
i)*beta(i)))*X(I,K))+
@sum(processor(K):((Csf+Ch)/BW)*(1-@sum(file(i)|lij(i)#eq#1
:x(i,k))))+
@sum(assign2(I,R)|lij(i)#eq#1 #and# v(r)#eq#1
:((Cesv+Cesf+Ceh)/(alpha(i)*beta(i)))*Y(I,R))+
@sum(extra(R)|v(r)#eq#1 :(Cesf+Ceh)*(1-
@sum(file(I)|lij(i)#eq#1 :Y(I,R))))+AP*Cp*(time- SLA);
!obj fun. : min costs of assigning files to extra and basic
processors,opportunity costs and penalty cost
 while satisfting priority , weight and time constraints;

BASIC= @size(processor);
sub = @sum(file(i)|i #GT#1 #and# i#LT#@size(file)  #and#
lij(i)#eq#1 :e(i));

fix=@if(sub #GE#Basic,BASIC,sub);

@sum(assign1(I,k) :x(i,k))= fix ;


@for(file(i)|i#eq#1:@sum(assign1(i,k):x(1,k))=0);!ensure that
start files are not being assigned;
```

```
@for(file(i)|i#eq#@size(file);!ensure that finish files are
not being assigned;

@for(file(i)|i#eq#1:;!ensure that start files are not being
assigned;
@for(file(i)|i#eq#@size(file);!ensure that finish files are
not being assigned;


@for(file(i):;!ensures that files are not being over
processed;

@for(file(i):;!determine how many times a files has been
processed at each time unit T;

@for(file(i):
;!total file allocation for any file i at a certain time T
doesn't exceed neither file multiprocessing ei nor file
required processing ni nor total number of available basic and
extra processors (K ,V);

@for(file(I):
;  !total file allocation for any file i at a certain time T
doesn't exceed neither file multiprocessing ei nor file
required processing ni nor total number of available basic and
extra processors (K ,V);


@for(processor(K):
@sum(file(I):X(I,K))<=P(K));!ensures that one file is allocted
to each available basic processor;

@for(extra(R):
@sum(file(I):Y(I,R))<=W(R));!ensures that one file is
allocated to each available extra processor;

@for(file(I):@bin(f(I)));!data file availability, f(i)=1 when
file is ready,0 otherwise;

@for(assign1(I,K):@bin(X(I,K)));!binary constraint for
allocating to basic processors;
@for(assign2(I,R):@bin(Y(I,R)));!binary constraint for
allocating to extra processors;
exe=@sum(file(i):q(i));!no. of times a file been executed;
req=@sum(file(i):n(i));!no. of times a file need to be
executed;
;!If AP =1 then data batch process exceeded SLA time and
penelty cost is imposed,  0 otherwise;

endsubmodel

submodel cpm:!finding the critical path for the remaining
unprocessed files in the network;
@for(file(i):remain(i)=n(i)-q(i));
```

```
@FOR( file( J)| J #GT# 1:
ES( J) = @MAX( PRED( a, J): ES( a) + remain( a))
);
@FOR( file( a)| a #LT# LTASK:
LS(a) = @MIN( PRED( a, J): LS( J) - remain( a));
);
@FOR( file( a): SLACK( a) = LS( a) - ES( a));
ES( 1) = 0;
LTASK = @SIZE( file);
LS( LTASK) = ES( LTASK);
@for(file(a):
EF(a)= ES(a)+remain(a));
@for(file(a):

LF(a)= LS(a)+remain(a));

MIN=path;
path=EF(LTASK) -ES(1);
@for(pred(a,J)|slack(a)#eq#0 #and# slack(j)#eq#0 :
ES(J) >=ES(a) +remain(a));

;!finding the value of the penalty cost in case the critical
path exceed the SLA time;

;!finding the value of renting one additional extra processor
in case the critical path exceed the SLA time;

endsubmodel

calc:
basic=@size(processor);
used_extra=0;!initialization, no extra processors are rented;
time=0;

@for(file(i)|i #GT#1 #and#
i#LT#@size(file):!initialization,all files are unprocessed ;
q(i)=0);
q(1)=0;
q(@size(file))=0;

@for(processor(k):
p(k)=1);!initialization,basic processors are ready;

@for(extra(R):
W(r)=1);!initialization,extra processors are ready;

@for(extra(R):
v(r)=0);!initialization,no extra processors are rented;


@SET('TERSEO', 2); ! Ask for Terse output, (0 = default);

@solve (main);!solve main program;
```

```
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','u0x') = x;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','ch') = ch;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','csf') = csf;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','csv') = csv;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','ceh') = ceh;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','cesf') =
cesf;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','cesv') =
cesv;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','used_extra')
= used_extra;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','BW') = BW;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','SLA') = SLA;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','basic') =
basic;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','u0y') = y;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','u0time') =
time;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','cp') = cp;
!establish excel link to transfer output results to excel
file;

@WRITE('Batch process cost minimization.'
,@NEWLINE(1));!writing the lingo output allocation report;
  @WRITE('          ',  @NEWLINE(1));

    @WRITE('    Time  : ',time,     @NEWLINE(1));
    @WRITE('    Ch    : ',Ch,     @NEWLINE(1));
    @WRITE('    Csf   : ',Csf,     @NEWLINE(1));
    @WRITE('    Csv   : ',Csv,     @NEWLINE(1));

  @WRITE(' Basic file allocation :',@NEWLINE(1), '      File
Basic Processor     ', @NEWLINE(1));
  @FOR( assign1(i,k) | X(i,k) #GT# 0.5:
    @WRITE('       ',file(i),'                ', processor(k),
@NEWLINE(1));
      );
  @WRITE('          ',  @NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));

    @WRITE('    Ceh    : ',Ceh,     @NEWLINE(1));
    @WRITE('    Cesf   : ',Cesf,     @NEWLINE(1));
    @WRITE('    Cesv   : ',Cesv,     @NEWLINE(1));
@WRITE(' Utilized Extra Processors :',used_extra,@NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));

  @WRITE(' Extra file allocation :',@NEWLINE(1), '      File
Extra Processor     ', @NEWLINE(1));
  @FOR( assign2(i,R) | y(i,R) #GT# 0.5:
    @WRITE('       ',file(i),'                ', extra(R),
@NEWLINE(1));
      );
  @WRITE('          ',  @NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));

@solve (cpm);!solve the submodel to find the critical path;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-
b.XLS','critical_path') = path;


@for(clock(z):
@IFC( exe #LT#req:!check termination condition;
time= z;!increment clock ;

@solve (main);

@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS',clock(z)+'x')
= x;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS',clock(z)+'y')
= y;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-
b.XLS',clock(z)+'time') = time;
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-
b.XLS',clock(z)+'used_extra') = used_extra;

@WRITE('Batch process cost minimization.' ,@NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));

    @WRITE('    Time : ',time,     @NEWLINE(1));
    @WRITE('    Ch   : ',Ch,     @NEWLINE(1));
    @WRITE('    Csf  : ',Csf,     @NEWLINE(1));
    @WRITE('    Csv  : ',Csv,     @NEWLINE(1));

  @WRITE(' Basic file allocation :',@NEWLINE(1), '      File
Basic Processor     ', @NEWLINE(1));
  @FOR( assign1(i,k) | X(i,k) #GT# 0.5:
    @WRITE('       ',file(i),'               ', processor(k),
@NEWLINE(1));
      );
  @WRITE('          ',  @NEWLINE(1));

  @WRITE('          ',  @NEWLINE(1));

    @WRITE('    Ceh   : ',Ceh,     @NEWLINE(1));
    @WRITE('    Cesf  : ',Cesf,     @NEWLINE(1));
    @WRITE('    Cesv  : ',Cesv,     @NEWLINE(1));

@WRITE(' Utilized Extra Processors :',used_extra,@NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));

  @WRITE(' Extra file allocation :',@NEWLINE(1), '      File
Extra Processor     ', @NEWLINE(1));
  @FOR( assign2(i,R) | y(i,R) #GT# 0.5:
    @WRITE('       ',file(i),'               ', extra(R),
@NEWLINE(1));
      );
  @WRITE('          ',  @NEWLINE(1));

  @WRITE('          ',  @NEWLINE(1));
  @WRITE('          ',  @NEWLINE(1));

@solve (cpm);
@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-
b.XLS',clock(z)+'critical_path') = path;

ENDCALC

DATA:
@OLE')C\:Users\Lenovo\Desktop\lingo programs 2,'2\
'used_extra')=V;
@OLE')C\:Users\Lenovo\Desktop\lingo programs 2,'2\
'Real_time')=time;

@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS',
'Real_time')=time;

@OLE( 'C:\Users\Lenovo\Desktop\outputex.2-b.XLS','sum_extra')
=used_extra;

@TEXT() = @TABLE(V);

ENDDATA
END
```

Appendix B

Network Allocation Example

T	Basic processors	No. of Acquired Basic processors	File processed by basic processors	Rented Extra Processors	No. of Acquired Extra Processors	File processed by extra processors
0	2	2	1	0	0	0
1	2	2	1	1	1	2
2	2	2	1	1	1	2
3	2	2	1	1	1	2
4	2	2	2	1	1	2
5	2	2	4,7	2	2	3
6	2	2	3	3	3	3,4
7	2	2	3	3	3	3,4
8	2	2	5	3	3	5,9
9	2	2	5	3	3	5,9
10	2	2	6	3	2	6,9
11	2	2	6	3	2	6,9
12	2	2	6	3	2	6,9
13	2	2	9,12	4	4	8,12
14	2	2	9,12	4	4	8,12
15	2	2	10,11	4	4	11,12
16	2	2	11,12	4	1	11
17	2	2	11	4	0	0
18	2	2	11	4	0	0
19	2	2	13	5	5	14
20	2	2	13	5	5	14
21	2	2	15	5	5	15
22	2	2	18	5	5	16,18
23	2	2	17,18	5	5	17,18
24	2	2	17,18	5	1	17
25	2	2	20	5	5	19,20
26	2	2	21	5	5	19,21
27	2	2	21	5	3	19,21
28	2	2	22,24	5	2	22
29	2	2	23,24	5	0	0
30	2	2	23,24	5	0	0
31	2	2	23,24	5	0	0
32	2	2	23,24	5	0	0
33	2	1	24	5	0	0
34	2	1	24	5	0	0
35	2	1	24	5	0	0

Network Allocation Example (Continued)

T	Basic processors	No. of Acquired Basic processors	File processed by basic processors	Rented Extra Processors	No. of Acquired Extra Processors	File processed by extra processors
36	2	1	24	5	0	0
37	2	1	24	5	0	0
38	2	2	25	5	3	25

Vita

Alia Hasan Al Sadawi was born in 1977, in Baghdad, Iraq and moved to the United Arab Emirates in 1980. She was educated in private schools and received a scholarship from Al Shola Private School where she graduated with honors.

In 2000, she graduated from Ajman University of Science and Technology, her degree was a Bachelor of Science in Electrical and Electronics Engineering. After Graduation she worked in Honeywell Middle East limited as Electronics Engineer then in Johnson Control International as Design and Estimation Engineer. Later she joined Nahas Intertrade as Electronics and Sales Engineer. Her last position was Communication Division Director in Unified Technologies. After many years in practical field she decided to go back and study for her masters. She joined the Engineering Systems Management Program at the American University of Sharjah where she was a Graduate Teaching Assistant.

Mrs. Alia participated in the Sixth Industrial Engineering and Operation Management conference IEOM in Kuala lumpur, Malaysia in 2016. Where she presented a paper related to Batch Process Optimization.