Article

Efficient Dynamic Cost Scheduling Algorithm for Financial Data Supply Chain

Alia Al Sadawi [1], Abdulrahim Shamayleh [1,2,*] and Malick Ndiaye [1,2]

1 Engineering Systems Management Graduate Program, American University of Sharjah, Sharjah 26666, United Arab Emirates; g00047863@alumni.aus.edu (A.A.S.); mndiaye@aus.edu (M.N.)
2 Industrial Engineering, American University of Sharjah, Sharjah 26666, United Arab Emirates
* Correspondence: ashamayleh@aus.edu

Abstract: The financial data supply chain is vital to the economy, especially for banks. It affects their customer service level, therefore, it is crucial to manage the scheduling of the financial data supply chain to elevate the efficiency of banking sectors' performance. The primary tool used in the data supply chain is data batch processing which requires efficient scheduling. This work investigates the problem of scheduling the processing of tasks with non-identical sizes and different priorities on a set of parallel processors. An iterative dynamic scheduling algorithm (DCSDBP) was developed to address the data batching process. The objective is to minimize different cost types while satisfying constraints such as resources availability, customer service level, and tasks dependency relation. The algorithm proved its effectiveness by allocating tasks with higher priority and weight while taking into consideration customers' Service Level Agreement, time, and different types of costs, which led to a lower total cost of the batching process. The developed algorithm proved effective by testing it on an illustrative network. Also, a sensitivity analysis is conducted by varying the model parameters for networks with different sizes and complexities to study their impact on the total cost and the problem under study.

Keywords: financial data; supply chain management; data batching; scheduling; batching cost; parallel processing; optimization; multi-processing commitment

Citation: Al Sadawi, A.; Shamayleh, A.; Ndiaye, M. Efficient Dynamic Cost Scheduling Algorithm for Financial Data Supply Chain. *Algorithms* **2021**, *14*, 211. https://doi.org/10.3390/a14070211

Academic Editor: Frank Werner

Received: 18 June 2021
Accepted: 12 July 2021
Published: 14 July 2021

1. Introduction

In today's world, economies and commodity markets swing rapidly; personal, organizational, and business networks are becoming more interconnected, instrumented, and intelligent [1]. One of the important aspects of the business world is supply chain management. It works towards satisfying customers' requirements through the efficient use of resources resulting in reducing cost and increasing profit margin for companies [2]. With a highly competitive global market, supply chains have to be more responsive to customers' needs. Supply chain represents all stages of a process; usually, researchers focus on materials that are manufactured to become an end product; however, the supply chain covers a wider range that extends beyond physical assets. For instance, looking at the banking sector, their main supply chain is of financial data that needs to be processed using available resources which are software and hardware processors.

One of the main instruments used in business networks for the management of the financial data supply chain is data batch processing (DBP). It plays a critical role in daily operations carried out in most organizations in different business fields. Data batch processing can be defined as the execution of data files as input batches using the available resources and gather the resulted files as output batches while satisfying files priorities, predecessors constraints, and time constraints [3,4]. The execution is carried out on the mainframe computer that runs at a scheduled time or on an as-needed basis without user interaction and minimal or no interaction with a computer operator [3,4]. The main feature of data batch processing is its ability to handle an enormous amount of data files which makes it attractive and essential to many organizations that are constantly dealing with heavy business activities [5]. It contributes to the major part of the workload on mainframe computers that will often run a large number of business workflows with complex interrelations, requiring careful scheduling and prioritizing to ensure that all batch jobs run in the correct order and meet strict deadlines [3,4,6].

Our research considers the case of processing end-of-day (EOD) operations which include highly important and frequently used activities and services such as: preparing customers' bank statements; credit card processing bills; fraud detection data; merging the day′s transactions into master files; sorting data files for optimal processing; providing daily, weekly, monthly, and annual reports; invoices and bank statements; performing periodic payroll calculations; and applying interest to financial accounts [4]. Banks seek to elevate their level of customer service by optimizing their data supply chain scheduling. This supply chain consists of the bank, a service provider that arranges the processing of data, and processors providing company that rents and leases processors and software to service providers. Also, in our research, data supply chains adopt the technique of lot streaming which is a technique that splits a given data job, each consisting of similar files, into tasks to allow overlapping of successive operations in processing systems thereby reducing the processing makespan.

The proposed work addresses the financial data supply chain scheduling problem from an operational perspective by considering the scheduling at an individual job level. The problem can be viewed as a single-stage supply chain scheduling problem where jobs are arranged to be processed by mainframe processors that can be modeled as a series of flow shop machines. Processors are leased and rented from manufacturing and supplying company as needed by a third-party company that performs the data scheduling. Data arrives raw and unprocessed from banks to a service provider which needs to manage scheduling them for processing as per the available resources (software and hardware) in the most efficient way. After processing, jobs must be returned back to banks where they will be charged for the provided service.

In order to address financial data supply chain, this research covers all aspects of data batch processing, being that it is the tool that manages banks′ financial data supply chain scheduling. The scheduling aspects of DBP are job priorities, precedence relationships, constraints in addition to cost. It is important to understand that cost consideration in data batch processing is vital due to the extremely high cost of the resources involved. However, previous research has overlooked the cost despite its importance, and the focus was on the effectiveness of the scheduling component of the process. The high cost of software and hardware resources used in the batch process and the patent rights of companies that own the platform made it not only beneficial, but also essential for DBP users to reduce the cost associated with it. Therefore, this research intends to bridge such a gap and provides a comprehensive study that covers the important aspects of data batch processing. Thus, the main contributions of this work can be summarized as follows:

- Consider all types of costs related to the batch process which are the 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.
- Develop an iterative dynamic scheduling algorithm (DCSDBP) to optimize the data batch processing considering the different costs, availability of resources, and customer service level agreement (SLA) along with the rest of the batch process factors such as the clients′ priorities, tasks predecessors and time.
- Utilize the created optimization model to address the problem of controlling the availability of resources such as processors and software required to process input batches and balance the usage of current owned resources and the need to rent additional ones while maintaining the lowest possible cost for the whole data batch processing.

The rest of the paper is organized as follows: Section 2, the background related to this work is presented. The data batch algorithm is presented in Section 3. The illustrative example and sensitivity analysis are presented in Sections 4 and 5 respectively. Finally, the conclusions are presented in Section 6.

2. Literature Review

Supply chain management has another perspective when associated with the financial sector. An alternative concept appears that relates to information or data and supply chain known as Financial Information Supply Chain Management. The researchers in [7] emphasized the need to improve financial operating processes in order to optimize financial data supply chain management by reducing processing time, improving efficiency, and decreasing associated costs of equipment and computers.

Other studies reached an aligned outcome such as the case study by [8] to design, develop, and implement an inter-organizational system by the Reserve Bank of Australia (central bank) and other authorities to remodel data reporting by financial institutions in Australia. The research concluded that the complexity of data consumption patterns led to an increased interdependence within the financial information supply chain which requires developing data exchanges and commodity-like IT infrastructures.

Also, a study by [9] stated that multiple governments have implemented Integrated Financial Management Information Systems to improve effectiveness and streamline business processes. Authors determine the need for automated financial operations to improve data supply chain efficiency.

The authors of [7] believe that in order to improve financial performance, we need to utilize a tool from lean manufacturing. They state that although financial operations are not the same as manufacturing operations, they share more similarities than might be acknowledged. These similarities entitle the financial data supply chain to adopt batch processing and scheduling from lean manufacturing.

Batch processing is widely used because of its ability to meet business-critical functions. It is mainly applied when dealing with large, repeated jobs that can be carried out at a prescribed time. The batch process is complex posing a challenge to efficiently allocate the needed resources. In this section, related literature to approaches used in data batching is presented. Also, related work of using data batching in other fields is presented as well.

The scheduling problem was studied by many researchers since it is a critical issue in batch process operation and performance improvement. Page et al. [10] considered eight common heuristics along with the genetic algorithm (GA) evolutionary strategy to dynamically schedule tasks to processors in a heterogeneous distributed system; they found that using multiple heuristics to generate schedules provides more efficient schedules than using each heuristic on its own. Méndez et al. [11] classified and presented the different state-of-the-art optimization methods associated with the batch scheduling problem. Osman et al. [12] proposed a model for data batch scheduling; they presented a dynamic, iterative framework to assign the required tasks to available resources taking into consideration the predecessors, constraints, and priority of each job. Al Sadawi et al. [13] studied the data batch problem with the goal of minimizing costs and satisfying customer service level agreement. Lim and Chao [14] used fuzzy inference systems to model the preferences of users and decide on the priority of each schedule with the goal of providing users with more efficient throughput. Xhafa and Abraham [15] developed heuristic and metaheuristic methods to deal with scheduling in grid technologies which are considered more complicated than scheduling in classical parallel and distributed systems. Aida [16] evaluated multiple job scheduling algorithms performance to investigate the effect of job size characteristics on job scheduling in a parallel computer system. Stoica et al. [17] described a schedule based on the microeconomic paradigm for online scheduling of a set of parallel jobs in a multiprocessor system. The user was granted control over the performances of his jobs by providing him with a saving account containing an amount of money used to run the jobs. Stoica [18] considered the problem of scheduling an online set of jobs on a parallel computer with identical processors by using simulation to compare three microeconomic policies with three variable partitioning policies. Islam et al. [19] demonstrated higher revenue and better performance by using their proposed new scheduling heuristic called Normalized Urgency to prioritize jobs based on their urgency and their processing times.

A study by Damodaran and Vélez-Gallego [20] developed a simulated annealing algorithm to evaluate the performance of batch systems in terms of total completion time with the goal of minimizing the processing time, and Mehta et al. [21] proposed a parallel query scheduling algorithm by dividing the workload into batches and exploiting common operations within queries in a batch, resulting in significant savings compared to single query scheduling techniques. Grigoriev et al. [22] developed a two-phased LP rounding technique that was used to assign resources to jobs and jobs to machines where 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. Also, Bouganim et al. [23] studied execution plans performance of data integration systems where they proposed an execution strategy to reduce the query response time by concurrently executing several query fragments to overlap data delivery delays with the processing of these query fragments. Ngubiri and van Vliet [24] proposed a new approach for parallel job schedulers fairness evaluation where jobs are not expected to have the same performance in a fair set up when they do not have the same resource requirements and arrive when the queue and system have different states. Arpaci-Dusseau and Culler [25] used a 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 a distributed environment.

From an economic point of view, Ferguson et al. [26] adopted the human economic model and implemented it on resource allocation in a computer network system which resulted in limiting the complexity of resource sharing algorithms by decentralizing the control of resources. Additionally, Kuwabara et al. [27] presented a 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. Chun and Culler [28] presented a performance analysis of market-based batch schedulers for clusters of workstations using user-centric performance metrics as the basis for system evaluation. Also, Sairamesh et al. [29] proposed a new methodology based on economic models to provide Quality of Service (QoS) guarantees to competing traffic classes in packet networks. Yeo et al. [30] outlined a taxonomy that describes how market-based resource management systems can support utility-driven cluster computing; the taxonomy is used to survey existing market-based resource management systems to better understand how they can be utilized.

Islam et al. [31] 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. They analyzed the impact of job opportunity cost on the overall revenue of the supercomputer center and attempted to minimize it through predictive techniques. Mutz and Wolski [32] presented a novel implementation of the Generalized Vickrey Auction that uses dynamic programming to schedule jobs and computes payments in pseudo-polynomial time. Mutz et al. [33] 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. Tests indicated that the mechanism meets its theoretical predictions in practice and can be implemented in a computationally tractable manner.

Lavanya et al. [34] proposed two task scheduling algorithms for heterogeneous systems. Their offline and online scheduling algorithms aimed at reducing the overall makespan of task allocation in cloud computing environments. The algorithms' simulation proved that they outperformed standard algorithms in terms of makespan and cloud utilization. Minimizing makespan was the objective of the research conducted by Muter [35] which tackled single and parallel batch processing machine scheduling. The author presented a reformulation for parallel batch processing machines and proposed an exact algorithm to solve this problem. Also, Jia et al. [36] developed a mathematical model and a fuzzy ant colony optimization (FACO) algorithm to schedule parallel non-identical size jobs with fuzzy processing times. The batch processing utilized machines with different capacities and aimed at minimizing the makespan. Additionally, Li [37] proposed two fast algorithms and a polynomial time approximation scheme (PTAS) to tackle the problem of scheduling n jobs on m parallel batching machines with inclusive processing set restrictions and non-identical capacities. The research aimed at finding a non-preemptive schedule to minimize makespan.

Another study by Josephson and Ramesh [38] aimed at creating a task scheduling process by examining the various real times scheduling algorithm. The study presented a new algorithm for task scheduling in a multiprocessor environment. The authors used TORSCHE toolbox for developing real-time scheduling in addition to utilizing features of particle swarm optimization. The proposed algorithm succeeded in executing a maximum number of the process in a minimum time. Also, Ying et al. [39] investigated the Distributed No-idle Permutation Flowshop Scheduling Problem (DNIPFSP) with the objective of minimizing the makespan. The authors proposed an Iterated Reference Greedy (IRG) algorithm that was compared with a state-of-the-art iterated greedy (IG) algorithm, as well as the Mixed Integer Linear Programming (MILP) model on two benchmark problems showing promising results.

An energy-efficient flexible job shop scheduling problem (EFJSP) with transportation was the core of research by Li and Lei [40]. The authors developed an imperialist competitive algorithm with feedback to minimize makespan, total tardiness, and total energy consumption. The conducted experiments provided promising computational results, which proved the effectiveness of the proposed algorithm. Furthermore, a study addressing distributed unrelated parallel machines scheduling problem aiming at minimizing makespan in the heterogeneous production network was proposed by Lei et al. [41]. A novel imperialist competitive algorithm with memory was developed by authors and experiments were conducted to test the performance of where the computational results proved the effectiveness of the algorithm. A study aimed at optimizing the trade-off between the total cost of tardiness and batch delivery was conducted by Rahman et al. [42]. To achieve this goal, the authors proposed three new metaheuristic algorithms which are the Differential Evolution with different mutation strategy variation, a Moth Flame Optimization, and Lévy-Flight Moth Flame Optimization algorithm. The algorithms were validated through an industrial case study.

Luo [43] tackled the dynamic flexible job shop scheduling problem under new job insertions. The goal of the research was to minimize the total tardiness; therefore, the authors proposed a deep Q-network (DQN). The developed DQN was trained using deep Q-learning and numerical experiments confirmed the superiority and generality of DQN. Also, Yun et al. [44] suggested a genetic algorithm based energy-efficient design-time task scheduling algorithm for an asymmetric multiprocessor system. The proposed algorithm adaptively applies different generation strategies to solution candidates based on their completion time and energy consumption. Experiments proved that it minimized energy consumption compared to existing methods. Finally, Saraswati et al. [45] aimed at minimizing the total tardiness in batch completion time using the metaheuristic approach of simulated annealing. The programming was done using Python programming. The research case study scheduled batches to parallel independent machines where the results of data processing demonstrated reduced total tardiness.

As concluded from the above survey, none of the articles found in the literature covered all aspects of data batch processing and scheduling. While some researchers concentrated on fulfilling the time constraint, others aimed at scheduling the maximum number of tasks effectively; however, none of the studies found in the literature considered cost during the scheduling process of data batches. This work tackles the significantly effective and widely used data batching process covering all its aspects. It will focus 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. This work represents a major contribution to the literature since it comprises all aspects of the DBP including cost.

3. Data Batch Algorithm

This research tackles the financial data supply chain management for the banking and financial sector which mainly revolves around processing financial-related data using available resources. Those resources are usually software and hardware processors. A major tool used in the management of the financial data supply chain is data batch processing (DBP). Our study covers all aspects of data batch processing—such as job priorities, precedence task relation, and time constraints—in addition to different types of cost. It is vital to emphasize the importance of the cost aspect in data batch processing since the utilized resources′ costs are very high. Yet, previous research work has neglected considering DBP cost despite its significance. Therefore, this research is extremely important since it includes different types of DBP costs in addition to all other aspects in a scheduling algorithm.

The DBP which the financial sector relies heavily on has been bounded by the IT systems that drive banking businesses making them in severe need of rapid, efficient and effective processes to be able to focus on their clients holistically. They use DBP widely in their daily operations like processing end-of-day (EOD) jobs which include highly important and frequently used activities and services such as: preparing employees′ payroll, interest rate calculations, and customer′s bank statements, credit card processing bills, fraud detection data, and many others [4]. Banks usually outsource DBP to a third-party company (service provider) that takes charge of arranging and processing the data and aggregating the output as shown in Figure 1. The service provider company leases processors and software from a provider to perform the batch process for the bank based on a Service Level Agreement (SLA). The service provider usually operates after closing hours so there will be no more data entries or online intervention.

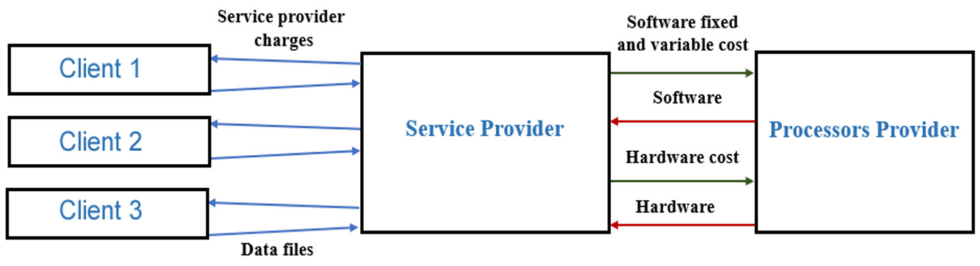

Figure 1. Data batching service network.

DBP begins with scheduling data, gathered in batches by type, to the available processor. The data files are scheduled while taking into consideration job priorities, predecessors and other constraints meaning that jobs will not be processed simultaneously but rather as per their schedule. The objective is to execute the tasks using the available resources to improve utilization, reduce cost and increase their profit while meeting the SLA. Each job consists of one or more executable data files. It is considered the surface layer of batch systems. A job consists of multiple tasks; therefore, the network of tasks is a detailed network view of jobs. In other words, a job is defined as a collection of tasks that are used to perform a computation. The developed dynamic scheduling algorithm considers cost types associated with the data batch scheduling which are: servers and software 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 (SLA), and the opportunity cost representing the cost of idling a resource for any period of time due to inefficient task allocation.

The data batching dynamic algorithm will be presented next. The algorithm will efficiently decide on resources allocation for the service provider to minimize the total operational cost. The service provider will batch process data for its clients one at a time according to a specific service agreement with that client.

3.1. Dynamic Cost Scheduling Algorithm for Data Batch Processing (DCSDBP)

Usually, given the current business practice dealing with such techniques, certain market-based rules govern the implementation of these scheduling processes. Therefore, the following assumptions underpinning the proposed algorithm are applied:

- The service-providing company lease a fixed number of data processors.
- Reserving processors is not allowed at the beginning or during the batch process scheduling. A processor is immediately acquired once a decision is made to rent it.
- The characteristics of data files that will be processed as batches such as size and priorities are specified at the beginning of the scheduling process.
- Hardware and software costs are incurred for processing data.
- The hardware cost is a fixed amount.
- Software cost will have fixed and variable cost amounts; the variable cost is charged based on usage per unit time.
- Additional processors can be rented anytime during the execution; however, costs will be higher, 1.25 times, than the currently leased processors′ hardware and software fixed cost. Also, a software variable cost is charged per unit time of usage. These assumptions are derived from practices in the field.
- Additional processors are rented only if there is a chance of not meeting the SLA.
- When an additional processor is acquired, it will be accounted for from the time it is acquired until the end of the batch window.
- Whenever an additional processor is rented at any time unit T, the endorsed fixed hardware and software costs will be calculated from the time of renting until the end of the batch process.
- Each leased or additionally rented processor executes a single task at a time.
- Different tasks can be processed in parallel. Parallel processing can occur only if there is no restriction due to priorities and predecessors relations provided in advance.
- A single data file containing multiple tasks can be multi-processed on multiple processors at the same time if it is allowed by the tasks predefined predecessors relations and the hardware and software resources are available.
- The iteration clock time unit is estimated by the time it takes to process the smallest size unit of the data file.
- The total DBP time is a multiple of the iteration unit time. Files are allocated to processors until a predetermined SLA time is reached. A penalty cost is imposed on each delay unit of time if the SLA is exceeded. The total penalty cost is calculated by multiplying the time delay by the penalty cost per unit time.
- 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.

3.1.1. Indices

i,j	Data file.
k	Leased processor
r	Rented processor.
T	Clock discrete time

3.1.2. Problem Parameters

I	Set of data files.
$I\prime^T$	Subset of the data files that are available for processing at any time T.
n_i	Required processing time for data file i.
SLA	Batch process time as agreed on in the SLA.
K	Set of all leased processors.
R	Set of rented processors.
V^T	Number of rented extra processors that can be acquired at any discrete time T.
l_{ij}^T	Binary parameter equal to 1 if data file j immediately precedes data file i, and 0 otherwise (parameters of precedence/dependency matrix).
α_i^T	Data file weight based on precedence/dependency matrix.
β_i	Data file scheduling priority provided by the client.
l_{ij}^T	Precedence/dependency matrix at each period T.
BW	Available batch process window.
Csf	Leased processor software fixed leasing cost.
Csv	Variable leasing cost per unit time of a processor software.
Ch	Leased processor hardware cost.
$Cesf$	Fixed rental cost per unit time of additional processor software.
$Cesv$	Variable rental cost per unit time of additional processor software.
Ceh	Rental cost per unit time of additional processor hardware.
Cp	Penalty cost per unit time for failing to execute the batch process as per the SLA.
e_i	Number of times file i can be multi processed.
TCp	Delay time total penalty cost.
T_H	Total cost of renting one additional processor at any time unit T.
D^T	Available files total multiprocessing ei 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.

3.1.3. Problem Variables

P_k	Binary variable equal to1 if processor *k* is available and 0 otherwise.
W_r	Binary variable equal to1 if processor *r* is available 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.
q_i^T	Number of times data file *i* has been processed. It is incremented by one every time data file *i* being processed.
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 $I\prime^T$.
ES_i^T	Early start of file *i*.
LS_i^T	Late start of file *i*.
S_i^T	Slack (the 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.

3.1.4. Problem Decision Variables

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* is allocated to extra processor *r*, and 0 otherwise.

The DCSDBP algorithm is illustrated in Figure 2 and the step-by-step description of the algorithm is as follows.

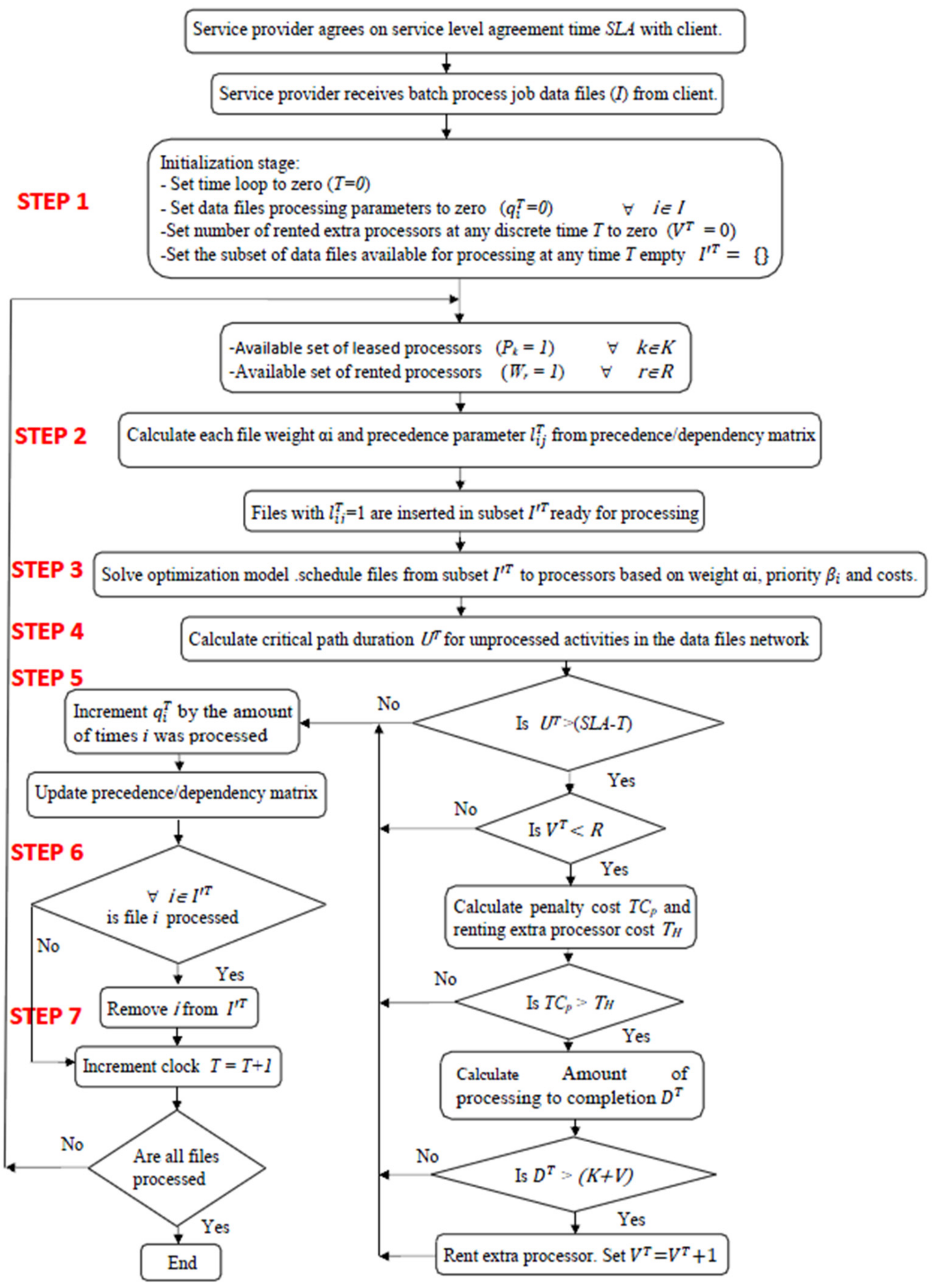
Service provider agrees on service level agreement time SLA with client.
Service provider receives batch process job data files (I) from client.
STEP 1
Initialization stage:
- Set time loop to zero (T=0)
- Set data files processing parameters to zero (q_i^T=0) ∀ i∈I
-Set number of rented extra processors at any discrete time T to zero (V^T = 0)
-Set the subset of data files available for processing at any time T empty I'^T = {}
-Available set of leased processors (P_k = 1) ∀ k∈K
-Available set of rented processors (W_r = 1) ∀ r∈R
STEP 2
Calculate each file weight αi and precedence parameter l_{ij}^T from precedence/dependency matrix
Files with l_{ij}^T=1 are inserted in subset I'^T ready for processing
STEP 3
Solve optimization model .schedule files from subset I'^T to processors based on weight αi, priority β_i and costs.
STEP 4
Calculate critical path duration U^T for unprocessed activities in the data files network
STEP 5
Increment q_i^T by the amount of times i was processed
No
Is U^T>(SLA-T)
Yes
Update precedence/dependency matrix
No
Is V^T< R
Yes
STEP 6
∀ i∈I'^T is file i processed
No
Calculate penalty cost TC_p and renting extra processor cost T_H
Yes
No
Is TC_p > T_H
STEP 7
Remove i from I'^T
Yes
Increment clock T = T+1
Calculate Amount of processing to completion D^T
No
Are all files processed
No
Is D^T > (K+V)
Yes
Yes
End
Rent extra processor. Set V^T=V^T+1

Figure 2. Dynamic Cost Scheduling Algorithm for Data Batch Processing (DCSDBP).

Step 1: Preparatory and Initialization Stage

This step involves preparing the initial data which consists of the subset of data files that are ready for processing and the availability of leased and rented. We also set:

1. The extra processors′ utilization parameter to zero indicating that no extra processor is used at the beginning of the allocation process.
2. The data files processing parameters to zero meaning that files are not being processed yet. In addition to that, the time loop is initialized where time is set to zero. At the end of the step, we initialize all data needed to start the iterative algorithm.

$$I\prime^{T} = \{\} \tag{1}$$

Set

$$P_k = 1 \; \forall \; k \in K \tag{2}$$

Equation (2) indicates that leased processors are ready since the binary variable P_k is set to 1.

Set

$$Wr = 1 \; \forall \; r \in R \tag{3}$$

Equation (3) indicates that rented processors are ready since the binary variable W_r is set to 1.

Set

$$V^T = 0 \tag{4}$$

Equation (4) indicates that the number of rented extra processors acquired at this time period V^T is zero.

Set

$$q_i^T \; = \; 0 \; \forall \; i \in I \tag{5}$$

Equation (5) indicates that a data file i have never been processed since q_i^T is set to zero.

Set

$$T = 0 \tag{6}$$

Equation (6) indicates the beginning of the time loop of the algorithm where the time T is set to zero.

Step 2: Set of Files Available for Processing

We assign parameters and weights to the subset of data files available for processing $I\prime^T$ based on their precedence obtained from the dependency matrix.

If:

$$\sum_{j=1}^{J} l_{ij}^T \; = \; 1 \; \forall \; i \tag{7}$$

Then set:

$$f_i^T = 1 \tag{8}$$

$$I\prime^T = I\prime^T + \{i\} \tag{9}$$

$$\alpha_i^T = \sum_{\theta=1}^{I} l_{\theta i}^T \; \forall \; i \tag{10}$$

α_i^T is the weight for each data file. It is calculated using the precedence/dependency matrix by finding the number of files that depend on file i.

This step will indicate the set of data files available for processing which will serve as input for step 3 to start the scheduling of available files on the available processors.

Step 3: Allocation of Files to Processors

In this step, the algorithm allocates files to processors. The model presented in this step will allocate files with the objective function (11) of minimizing data file allocation cost while taking into consideration priority, weight and criticality of each file included in each subset at any time T.

$$\begin{aligned} \text{Min (Z T)} \; = \; & \sum_{i=1}^{I'} \sum_{k=1}^{K} \left(\frac{Csv + \left(\frac{Csf + \; Ch}{BW} \right)}{\beta_i \; \alpha_i^T} \right) X_{iK}^T + \sum_{k=1}^{K} \left(\frac{Csf+Ch}{BW} \right) \left(1 - \sum_{i=1}^{I'} X_{iK}^T\right) + \sum_{i=1}^{I'} \sum_{v=1}^{V} \left(\frac{Cesv+Cesf+Ceh}{\beta_i \; \alpha_i^T} \right) Y_{ir}^T + \\ & \sum_{v=1}^{V} (Cesf + Ceh) \left(1 - \sum_{i=1}^{I'} Y_{ir}^T\right) + A^T * \; Cp \; * \; (T - SLA) \end{aligned} \tag{11}$$

The first term considers the basic processors leasing cost, weight α_i^T and priority β_i at any time unit T. The second term considers the opportunity cost of not utilizing the leased processors at any time unit T. The third term handles the cost of renting additional processors, weight α_i^T and priority β_i at any time unit T. The fourth term considers the opportunity cost of not utilizing the additionally rented processors at any time *T*. the last term is concerned with the penalty cost of exceeding the *SLA*. The $\beta_i\ \alpha_i^T$ used in terms 1 and 3 ensures that files with higher priority and weight are scheduled first. *BW* is the available time during which processing can take place. It is used in Equation (11) to find what the fixed cost of resources per unit time is.

Subject to:

$$\sum_{k=1}^{K} X_{ik}^T + \sum_{r=1}^{V} Y_{ir}^T \le M_i f_i^T\ \forall\ i \in I\prime^T \text{ where M}i = \min\{e_i, n_i - q_i^T,\ K+V\} \tag{12}$$

The constraint in Equation (12) is concerned with leased and additionally rented processors allocation and their availability to ensure that the total file allocation does not exceed either file multiprocessing *ei* or file required remaining processing n_i or the total number of available basic and extra processors (*K*,*V*) for any file *i* at a certain time *T*.

$$q_i^T \le n_i\ \forall\ i \in I \tag{13}$$

The constraint in Equation (13) ensures that the number of times a file is processed is less or equal to the needed processing time.

$$\sum_{i=1}^{I'} X_{ik}^T \le P_k^T\ \forall\ k \in K \tag{14}$$

The constraint in Equation (14) ensures that exactly one data file is allocated to a single leased processor.

$$\sum_{i=1}^{I'} Y_{ir}^T \le W_r^T\ \forall\ r \in R \tag{15}$$

The constraint in Equation (15) ensures that exactly one data file is allocated to a single additionally rented processor.

$$X_{iK}^T\ 0 \text{ or } 1\ \forall\ i \in I \text{ and } k \in K \tag{16}$$

The constraint in Equation (16) declares that the decision variable X_{ik} is binary, meaning that file *i* either be assigned to a leased processor or not.

$$Y_{ir}^T = \quad 0 \quad \text{or} \quad 1 \quad \forall\ i \in I \text{ and } r \in R \tag{17}$$

The constraint in Equation (17) declares that the decision variable Y_{ir} is binary, meaning that a file *i* either be assigned to an additionally rented processor or not.

Step 4: Update Utilized Extra Processors

At this stage, we check if an additional processor should be rented at this time unit to avoid delay and penalty cost. This step involves calculating the critical path duration for the rest of the unprocessed activities in the data files network at each time unit and compare it to the remaining time until the end of the batch process time that is agreed on in the *SLA*. A trade-off between the cost of renting an additional processor and the penalty cost is made, and according to that, it will be decided whether to rent a new processor or incur a penalty cost. Renting a new processor is subject to the condition that the total number of data files available for processing is higher than the total number of rented basic and extra processors.

Critical path duration

$$\text{Calculate } U^T = \text{duration of the remaining critical path at time } T.ES_i^T = \text{Max} \{ES_j^T + (n_i - {q_i}^T)\}\ \forall\ i \text{ and } j\ \in\ I \text{ where } (i \ne j) \text{ and} \alpha_i <= \alpha_j \tag{18}$$

$$ES_i^T = 0 \, \forall \, i = 0 \tag{19}$$

$$LS_i^T = \text{Min}\,\{LS_j^T - (n_i - q_i{}^T)\} \, \forall \, i \text{ and } j \in I \text{ where } (i \neq j) \text{ and} \alpha_i <= \alpha_j \tag{20}$$

$$S_i^T = LS_i^T - ES_i \tag{21}$$

$$U^T = \sum_{i=1}^{I} (n_i - q_i{}^T) \, \forall \, i \in I \text{ and } S_i^T = 0 \tag{22}$$

In the above equations the slack S_i^T for each file i is calculated and the critical path for the network U^T is found for the files with slack equal to zero (S_i^T = 0).

Checking critical path duration against SLA

If

$$U^T \geq SLA - T \tag{23}$$

Then

$$TC_p = C_p * (U^T - (SLA - T)) \tag{24}$$

Else

$$TC_p = 0 \tag{25}$$

The above equations calculate the penalty cost TC_p for all cases of critical path duration against *SLA*.

Cost of renting extra processor in case there is a delay

If

$$U^T \geq SLA - T \tag{26}$$

Then

$$T_H = (Cesf + Cesh + Cesv) * U^T \tag{27}$$

Else

$$T_H = 0 \tag{28}$$

The above equations calculate the extra processor renting cost T_H in case of a delay.

Amount of processing to completion

$$D^T = (\textstyle\sum_{i=1}^{I'} e_i - q_i^T) \, \forall \, i \in I\prime^T \text{ and } e_i > 1 + (\textstyle\sum_{i=1}^{I'} e_i^T) \, \forall \, i \in I\prime^T \text{ and } e_i = 1 \tag{29}$$

Decision on renting extra processor

If

$$TC_p \geq T_H \text{ and } D^T > (K + V) \tag{30}$$

Then

$$V^T = V^T + 1 \tag{31}$$

$$T^r = T \tag{32}$$

$$V^T \leq R \tag{33}$$

In the above equations, the number of rented extra processor V^T is increased by one if the penalty cost TC_p is greater than the extra processor renting cost T_H.

In this step, we are calculating the maximum completion time using CPM without considering splitting because the decision to multi-process a job is not taken yet. A job multi-processing means it can be split if needed to minimize the completion time of the batch process. CPM is not the only tool in this study to make the decision. Looking at Equations (29) and (31), the decision to rent an additional processor is based on another criterion. D was introduced which, as per Equation (29), ensures that no extra processor shall be rented unless there is an available task ready to be allocated to it. This ensures that no additional processor will be rented unless it will be used.

Step 5: Update the Availability of Files

Each data file processing parameter is incremented by the number of times it was processed. When a file is fully processed, then it is removed from the subset for the following time unit and the rest of the process.

Update f_i^T Matrix:

$$q_i^T = q_i^T + \sum_{k=1}^{K} X_{ik}^T + \sum_{r=1}^{R} Y_{ir}^T \ \forall\, i \in \ I\prime^T \tag{34}$$

If

$$O_i^T = 1 \tag{35}$$

Then

$$f_i^{T+\Delta} = 1 \tag{36}$$

Else

$$f_i^{T+\Delta} = 0 \tag{37}$$

And

$$l_{ji}^{T+\Delta} = 0 \ \forall\, j \tag{38}$$

If

$$q_i^T = n_i$$

Then

$$\sum_{j=1}^{J} l_{ij}^{T+\Delta} = 0 \ \forall\, i \tag{39}$$

And

$$\sum_{i=1}^{I} l_{ji}^{T+\Delta} = 0 \ \forall\, j \tag{40}$$

This step will update the file availability to determine if any file needs further processing or all files are completed.

Step 6: Check Termination Condition

When all files are processed, then the algorithm shall stop, else the model will go to step 7.

If

$$\sum_{i=1}^{I} q_i^T = \sum_{i=1}^{I} n_i \tag{41}$$

Then

Stop.

If

$$T > SLA \tag{42}$$

Then

$$A^T = 1 \tag{43}$$

Else

$$A^T = 0 \tag{44}$$

Step 7: Update Clock

Increment iteration clock by one time unit until the DCSDBP algorithm allocates all files.

$$T = T+1 \tag{45}$$

Go to Step 2

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.

The Total batch process cost is updated as follows:

$$
\begin{aligned}
TBC &= (C_{sf} + C_h\,) * K + \sum_{T=1}^{END} \sum_{i=1}^{I'^T} \sum_{k=1}^{K} C_{sv} * X_{ik}^T \\
&+ \sum_{T=1}^{BW} \sum_{i=1}^{I'^T} \sum_{k=1}^{K} \frac{(C_{sf}+C_h\,)}{BW} * (1 - X_{ik}^T) \\
&+ \sum_{r=1}^{V} (C_{esf} + C_{eh})(END - T^r) + \sum_{T=1}^{END} \sum_{i=1}^{I'^T} \sum_{r=1}^{V} C_{esv} * Y_{ir}^T \\
&+ \sum_{r=1}^{V} \sum_{i=1}^{I'^T} \sum_{T=T^r}^{END} (C_{esf} + C_{eh})(1 - Y_{ir}^T) + A^{END} *Cp *(T^{END} - SLA)
\end{aligned} \tag{46}
$$

The total cost and the completion time will be calculated at the end of the DCSDBP algorithm. The total batch process cost (C) = leased processors fixed hardware cost + leased processors fixed software cost+ leased processors variable software cost + leased processors opportunity cost + rented processors variable software cost + rented processors fixed software cost + rented processors hardware cost + rented processors opportunity cost + penalty cost.

4. Illustrative Example

In this section, we present a numerical example to illustrate the proposed DCSDBP algorithm. Consider a network of 15 data files with the precedence relations as shown in Figure 3.

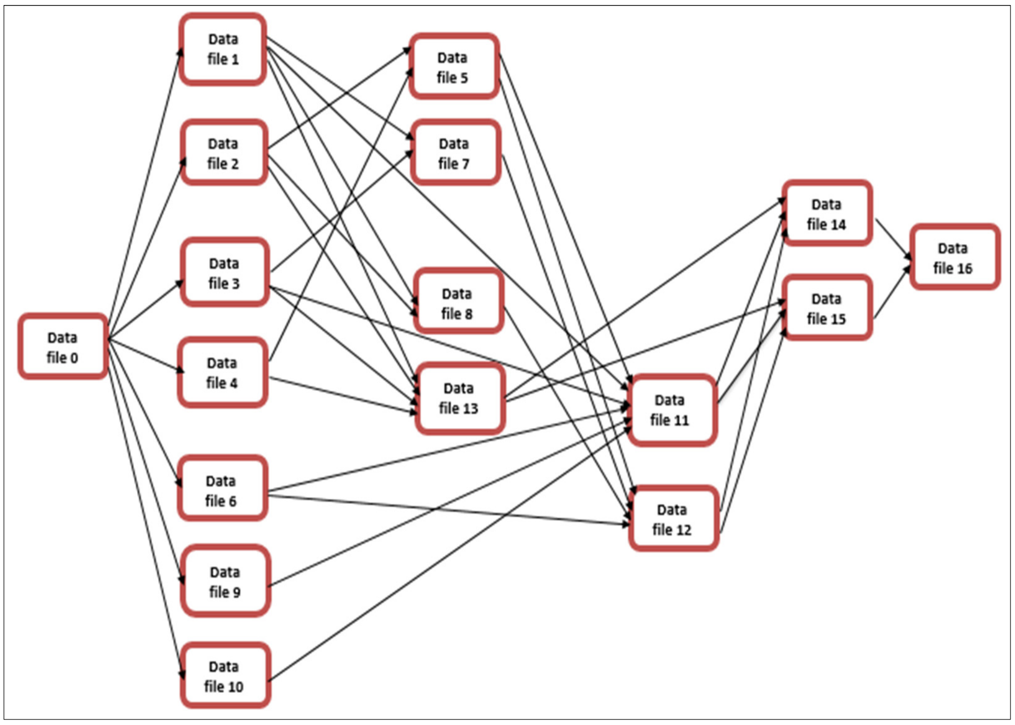

Figure 3. Illustrative example.

The precedence relationships are fixed relationships provided by the client that the service provider must use in processing the files; therefore, there cannot be any deadlocks relation. Table 1 presents the different parameters' values; these values are either given directly such as file's multiprocessing *ei*, processing time *ni* and priority *βi*, or derived from the files precedence relation such as file's weight *αi*ˆ*T* and precedence parameter *Lij*ˆ*T*. The following data were also used: there are two available leased processors; maximum number of available processors than can be rented is 5, *SLA* = 18 time units; batch window

BW = 22 time unit; leased processor fixed software cost *Csf* = \$10; leased processor hardware cost *Ch* = \$100; leased processor variable software cost per time unit *Csv* = \$2; rented processor fixed software cost *Cesf* = \$12.5; rented processor Hardware Cost *Ceh* = \$125; rented processor variable software cost per time unit *Cesv* = \$2.5; penalty cost per time unit in case of exceeding the *SLA* is *Cp* = \$200.

Table 1. Model parameters at *T* = *0*.

File	e_i	n_i	β_i	$\alpha_i^{T=0}$	L_{ij}
0	0	0	1	1	1
1	3	9	2	6	1
2	1	1	3	5	1
3	2	8	2	5	1
4	3	9	4	4	1
5	1	3	2	4	3
6	1	1	5	4	1
7	1	3	2	3	3
8	1	8	2	3	3
9	2	4	2	3	1
10	2	8	2	3	1
11	1	1	1	4	7
12	6	12	1	4	5
13	3	9	1	4	5
14	1	2	2	2	4
15	1	4	2	2	4
16	0	0	1	1	1

The algorithm was programmed using LINGO 15.0 x64. The Lingo output shows that 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, which means that ready files subset $I\prime^T$ is empty. At the beginning of each time unit *T*, the precedence parameters l_{ij}^T for all files are checked to determine which files are ready to be processed. All files having precedence parameter l_{ij}^T = 1 are considered ready and inserted to the ready files subset $I\prime^T$. 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 algorithm started at *T* = 0 with files 1, 2, 3, 4, 6, 9, and 10 ready for processing; therefore, they were inserted into the ready files subset $I\prime^T$. At *T* = 0 files 4 and 6 were processed by leased processor 2 and 1 respectively because these files have the highest calculated weight $\alpha_i^{T=0}$ and predetermined priority β_i while the rest of ready files were shifted for the next time unit. At *T* = 1, the values of precedence parameter l_{ij}^T are updated for all files, so the ones that have not been processed or not finished processing still have the value of 1 and are consequently still included in the ready files subset $I\prime^T$ 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 $I\prime^T$ since it is considered ready. Also, an additional processor V^T was rented and utilized at that time unit because the criteria of renting a new 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. The decision of renting a new processor is made since the cost of renting a new processor *TH* was found to be less than the penalty cost *TCp* at that time unit and also since there are ready files for the next time unit that exceeds the total number of available processor. During *T* = 1 file 4 was processed by leased processors 1 and 2, as well as by the additionally rented processor 1.

At *T* = 2 another additional processor was rented based on the above-explained mechanism. File 4 continued to be processed by leased processors 1 and rented processors 1 and 2 since it has a multiprocessing of $e_i = 3$. Also, file 2 was processed by leased processor 2. At $T = 3$ an additional third processor was rented and file 1 was processed by leased processors 1 and 2 and rented processor 3 due to its multiprocessing criteria. File 4 was processed by rented processors 1 and 2 and by that it is considered ready. At $T = 4$ the forth additional processor was rented and used to process file 1 along with rented processors 1 and 3 while rented processor 2 and leased processor 1 were used to process file 3. File 5 was processed by leased processor 2. $T = 5$ had the same files allocations as $T = 4$. It is noted that the program did not rent any more extra processor starting from $T = 5$ onwards which means it utilized 4 out of the 5 processors available for renting and that is based on the renting mechanism. The same steps were executed on all files until the end of processing at $T = 19$ when all files were processed.

The batch process cost calculation was based on using Equation (46) by using the input values listed earlier in this section and the allocation results from Lingo output; Table 2 demonstrates the detailed costs. Also, Table 2 clarifies the cost of allocating files to leased processor and that includes hardware and software fixed costs as well as software variable cost. Similarly, rented processors allocation 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 2, column (1) represents the time unit T, column (2) shows the total number of existing leased processors K, column (3) identifies how many leased processors are actually being acquired at each time unit T, column (4) calculates the leased processors variable cost Csv while the fixed software Csf and hardware Ch costs are calculated at the end because they are not related to time. Leased processors opportunity cost is determined in column (5) as per Equation (27). For rented processors, column (6) shows the number of additionally rented processors V^T at that time unit T, column (7) represents the number of rented processors actually being acquired at each time unit T. In column (8), and based on model assumption 14 which states that rented processors are paid for from the time unit T they are rented onwards, rented processors total allocation cost is calculated using all types of costs (extra fixed hardware cost Ceh, extra fixed software cost $Cesf$ and extra variable software cost $Cesv$). It is worth mentioning that in case of an additional processor being rented but not utilized due to unavailability of ready files, the total allocation cost will equal fixed hardware cost Ceh plus fixed software cost $Cesf$ while the variable software cost $Cesv$ will not exist since the processor is not being utilized. Rented processors opportunity cost is found in column (9) using fixed hardware cost Ceh plus fixed software cost $Cesf$. Finally, column (10) sums all the above costs for each time unit T.

At the end of Table 2 leased fixed costs are added, they represent fixed hardware cost Ch plus fixed software cost Csf for each leased processor k. Also as mentioned above penalty cost is calculated based on number time units the processing is delayed beyond the *SLA*. Since leased processors fixed hardware cost Ch and fixed software cost Csf 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 leased processors for the time units between the end of batch process END and BW.

From Table 2, we can see that from $T = 0$ until $T = 17$, the basic allocation cost was showing the utilization of both leased processors, which explains the zero value for opportunity cost of leased processors during the same period. However, at $T = 18$ and 19 one leased processor was utilized and that ended up in variable allocation cost for one processor and opportunity cost for the other. It can be also noticed that from $T = 0$ up to $T = 10$, every rented processor was utilized which ended in zero opportunity cost. After $T = 10$ some rented processors were not utilized due to unavailability of ready files such as at $T = 11$ where 3 out of 4 rented processors were utilized. Also at $T = 12$, 13, 16, 17, 18, and 19 none of the rented processors were utilized due to the unavailability of ready files.

Table 2. DBP cost summary.

	Leased Processors				Rented Processors				(10) Total Cost/Time Period $
(1) *T*	(2) No. of P	(3) No. of Acquired P	(4) Allocation Variable cost $	(5) Oppo. Cost $	(6) No. of P	(7) No. of Acquired P	(8) Total Allocation Cost $	(9) Oppo. Cost $	
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
					Total dynamic cost $ for all periods				**795.25**
					Penalty cost $				**400.00**
					Leased processors fixed costs $				**220.00**
					Remaining opportunity cost for Leased processors $				**20.00**
					Batch Process Total Cost $				**1435.25**

The above illustrative example shows the effectiveness of the developed algorithm in allocating files to processors. The algorithm 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 minimizing cost. 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.

The illustrative example shows that the batch processing is performed using a set of assumptions and constraints under which the problem makes the best decision at that time unit. Decisions are made dynamically based on the current status and previous decisions. This decision process ascertains that the best decision is taken at each iteration. We do not define the global network (of all feasible assignments of jobs to servers) to perform a direct optimization on it, we rather generate the subnetwork of feasible assignments at each iteration (Step 3 of the algorithm). This sequence of optimum decision-making ensures that the solution at the end is global. Not being optimal would mean that a better solution exists at some stage in the optimization process, which would contradict stage 3 as built.

5. Sensitivity Analysis

The whole research was motivated by working with a company in the field, therefore, whatever assumptions and constraints applied to the algorithm are based on practice. Given the initial results obtained from the illustrative example, they were promising and could be easily extended to cover other networks.

Sensitivity analysis was performed in two parts. The first part was conducted on the illustrative example network by changing different parameters. In part two, the algorithm performance was tested using networks with varying sizes and complexities.

5.1. Parameters Variation Analysis

Different parameters were tested and the summary of results total cost and process time are shown in Figures 4 and 5, respectively.

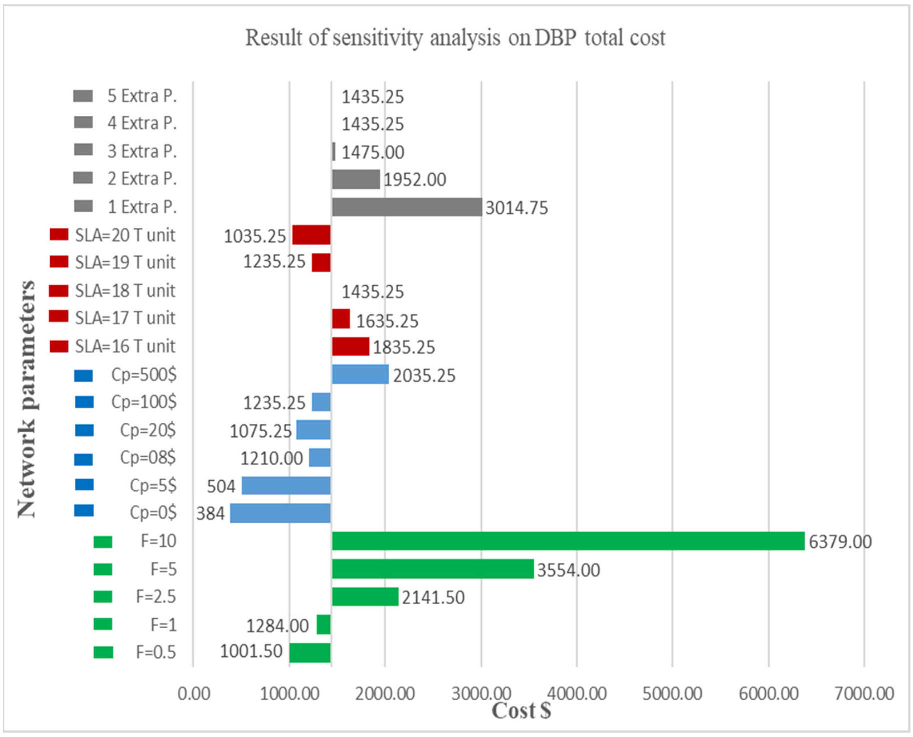

Figure 4. Result of sensitivity analysis on DBP total cost.

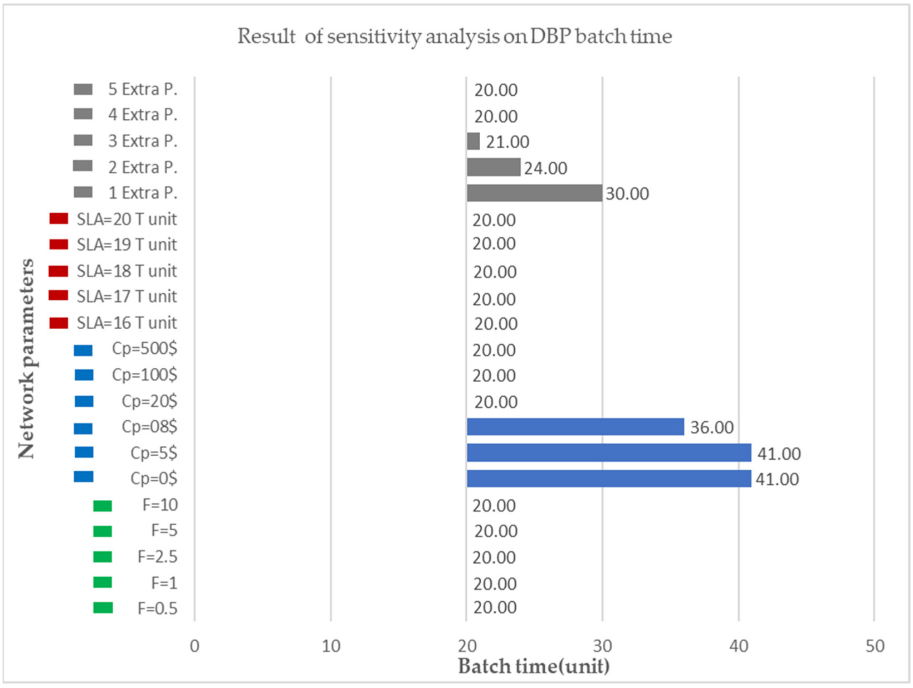

Figure 5. Result of sensitivity analysis on DBP batch time.

5.1.1. Varying Number of Processors Available to Rent

In the case of having four processors available for rent to be used in the batch process, the results were the same as the case of five available processors since the same number of processors were actually rented as in the illustrative example results shown earlier. However, having three available processors to rent resulted in a longer batch process time and higher total cost in addition to that, the SLA was exceeded by more time units. Having two processors available for rent resulted in more delay in batch process time, increased the total cost, and the SLA was exceeded by 6 time units. The batch process had the highest delay and total cost in the last case of having only one available processor to rent. The previous results shows that the algorithm is renting additional processors only when needed in order to perform the batch process with minimum execution time and total cost.

5.1.2. Changing SLA Value

Different values for the SLA were tested. The results show the lower the SLA value, the higher the penalty cost will be or the need to rent additional processors to avoid delay and the opposite in case of higher SLA. The values tested are for SLA = 16, 17, 19, and 20. It is recommended that, when deciding on the SLA time between the client and the service provider, 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.

5.1.3. Varying Penalty Cost per Time Unit

Different penalty cost values were assumed and the algorithm was tested accordingly. The results showed that the lowest batch process cost is obviously obtained when the penalty cost is set to zero. In this case, there will not be any trade-off between cost of renting an additional processor and any other cost; however, the program needed more time to run since the completion time to process doubled. In the case of the penalty, cost equals to \$5 same results were obtained as in the case of zero penalty cost. Three extra processors instead of four were used with a penalty cost equal to 8 and the total batch process time was higher; however, the total cost is less. In cases of penalty cost of \$20 and \$100 per time unit, 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. Increasing the penalty cost further to \$500 will not affect the way files are allocated to servers since the program will not 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 rented processor simply because of the network$'$s activity relations. In other words, due to precedence constraints jobs are processed only when they are available. Therefore, even if more resources are available, the duration will not be reduced since the resources will not be utilized; i.e., paying more money for resources may not reduce the completion time.

5.1.4. Changing the Processor Rental Costs

When increasing the renting cost of an additional processor to 2.5 times the leased processor cost, the program rented the same number of processors to finish the batch process because the penalty cost is still higher relatively. It only ended up increasing the total batch cost, while the same number of additional processors were utilized. Increasing the processors renting cost to 5 times the leased processor cost did not stop the program from renting additional processors to finish the batch process because the penalty cost is still relatively higher. However, the total batch cost increased while the same number of additional processors were utilized. To prove the efficiency of the algorithm, the case of equal costs for leased and rented processors was tested, the algorithm rented only what it needed from available additional processors to finish the batch process. Finally, in the case of setting costs ratio to 0.5, the total price went lower but the files-to-processors allocation stayed the same.

The above findings and analysis prove that when the algorithm reaches the optimum solution, it does not 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.

5.2. Jobs Network Size Analysis

In the second part of the sensitivity analysis, the developed algorithm was tested by varying the number of jobs and the network complexity. Different network sizes and relationship complexities were generated using a random network generator RanGen2 software [46]. Several data for different network sizes of 15, 25, 50, and 100 jobs were generated. Each network size was also generated based on precedence complexity measure by an indicator of complexity "I2 index" which is a measure of the closeness of a network to a serial or parallel network based on the number of progressive levels. If I2 = 0 then the network activities are all in parallel, meaning no relation between them, while if I2 = 1 then the network activities are all in series [47]. The average program run time for the complete DCSDBP Algorithm for each network is shown in Table 3. The results demonstrate that the algorithm is efficient since the program run time is relatively low and acceptable.

Table 3. Run times for variant network sizes with different complexities.

No. of Network Activities	Complexity Index I2	Run Time (s)
15	0.2	2
	0.5	3
	0.8	3
25	0.2	5
	0.5	2
	0.8	4
50	0.2	11
	0.5	13
	0.8	14
100	0.2	33
	0.5	34
	0.8	34

6. Conclusions

The financial data supply chain is of huge importance to the banking sector. It impacts financial institutes′ performance and customer service. Therefore, it is of great necessity to manage the scheduling of the financial data supply chain. The main tool utilized in the financial data supply chain is data batch processing. Batch scheduling and processing are extremely important because they are widely used in service industries to track tasks and data continuously. However, there is a lack of an efficient scheduling solution for data batch scheduling which creates a major issue for the financial sector. The goal of this work is to develop an iterative Dynamic cost scheduling for DBP (DCSDBP) algorithm that includes all aspects of DBP. Different types of costs associated with the batch process were taken into consideration to develop the iterative scheduling algorithm. While the algorithm worked towards minimizing these costs, it aimed at the same time to allocate files based on their weight and priority without violating network predecessors′ relations. Also, the algorithm tries to satisfy the time limit specified in the SLA. The developed algorithm proved its effectiveness in allocating data files to available resources while satisfying priority and predecessors constraints in addition to maintaining the minimum possible cost, keeping in mind the SLA time limit. 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 along with priority, weight, predecessor, and time factors, which led to a more effective allocation and a lower total batch process cost. It can also be seen from the results that renting more processors does not necessarily mean that the batch process will be performed in a shorter time because network logic relations or ′predecessors′ relations′ govern the process′s total time. The decision of whether or not to rent a new processor and when to do so is very important since it will affect the whole batch process in terms of file allocations to processors and total processing cost and time. Our research has positive implications on the performance and customer service of financial institutes and banks that choose to adopt it. It optimizes the scheduling of the data batch process which reflects on a more efficient and reliable financial data supply chain and through better management. The algorithm was developed under certain assumptions; while we tried to generalize it to cover batch processing, there are some limitations that could be addressed in future researches. For instance, it was assumed that resources costs are the same for leased processors, this could be generalized to assume different costs or even different costs as a future research direction. Also, additional research could be done on cases where renting additional processors has varying costs. Future researchers might also work on developing the additional processors renting mechanism to allow more than one processor to be rented for each time unit in case that serves the total completion time and maintains a low cost. In addition, one of our basic assumptions is that processors reservation is not allowed, which can be researched further to test the case where processor reservation is allowed and how it can be implemented; in addition to its impact on the different aspects of the batch process such as total process time, total cost and basic and extra processor utilization.

Author Contributions: A.A.S.: conceptualization, methodology, software programming, validation, and writing the original draft. A.S. and M.N.: conceptualization, methodology, and finalizing the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The authors received no specific funding for this work.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable. This article does not contain any studies with human participants or animals performed by any of the authors.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors have no conflict of interest to declare.

References

1. Jensen, C.T. Smarter Banking, Horizontal Integration, and Scalability. 2010. Available online: http://www.redbooks.ibm.com/redpapers/pdfs/redp4625.pdf (accessed on 15 June 2021).
2. Chang, Y.-C.; Chang, K.-H.; Chang, T.-K. Applied column generation-based approach to solve supply chain scheduling problems. *Int. J. Prod. Res.* **2013**, *51*, 4070–4086. [CrossRef]
3. Bullynck, M. What Is an Operating System? A Historical Investigation (1954–1964). In *Reflections on Programming Systems: Historical and Philosophical Aspects*; Philosophical Studies Series 2542–8349; Springer International Publishing: Cham, Switzerland, 2018; pp. 49–79.
4. Microsoft-Corporation. Batch Applications—The Hidden Asset. 2006. Available online: http://download.microsoft.com/download/4/1/d/41d2745f-031c-40d7-86ea-4cb3e9a84070/Batch%20The%20Hidden%20Asset.pdf (accessed on 15 June 2021).
5. Antani, S. Batch Processing with WebSphere Compute Grid: Delivering Business Value to the Enterprise. 2010. Available online: http://www.redbooks.ibm.com/redpapers/pdfs/redp4566.pdf (accessed on 16 June 2021).
6. Barker, M.; Rawtani, J. *Practical Batch Process Management*; Elsevier: Burlington, VT, USA, 2005.
7. Matyac, E. Financial Supply Chain Management. *Str. Financ.* **2015**, *96*, 62–63.
8. Fahy, M.; Feller, J.; Finnegan, P.; Murphy, C. Co-operatively re-engineering a financial services information supply chain: A case study. *Can. J. Adm. Sci.* **2009**, *26*, 125–135. [CrossRef]
9. Mbaka, A.; Namada, J. Integrated financial management information system and supply chain effectiveness. *Am. J. Ind. Bus. Manag.* **2019**, *9*, 204–232. [CrossRef]
10. Page, A.J.; Keane, T.M.; Naughton, T.J. Multi-heuristic dynamic task allocation using genetic algorithms in a heterogeneous distributed system. *J. Parallel Distrib. Comput.* **2010**, *70*, 758–766. [CrossRef] [PubMed]
11. Méndez, C.A.; Cerdá, J.; Grossmann, I.E.; Harjunkoski, I.; Fahl, M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Comput. Chem. Eng.* **2006**, *30*, 913–946. [CrossRef]
12. Osman, M.S.; Ndiaye, M.; Shamayleh, A. Dynamic scheduling for batch data processing in parallel systems. In Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems-Volume 1: ICORES, Angers, France, 6–8 March 2014; pp. 221–225.

13. Al Sadawi, A.; Shamayleh, A.; Ndiaye, M. Cost Minimization in Data Batch Processing. In Proceedings of the 6th International Conference on Industrial Engineering and Operations Management, Kuala Lumpur, Malaysia, 8–10 March 2016.
14. Lim, S.; Cho, S.-B. *Intelligent OS Process Scheduling Using Fuzzy Inference with User Models*; Springe: Berlin/Heidelberg, Germany, 2007; pp. 725–734.
15. Xhafa, F.; Abraham, A. Computational models and heuristic methods for Grid scheduling problems. *Future Gener. Comput. Syst.* **2010**, *26*, 608–621. [CrossRef]
16. Aida, K. Effect of Job Size Characteristics on Job Scheduling Performance. In *Lecture Notes in Computer Science*; Springe Science & Business Media: Berlin/Heidelberg, Germany, 2000; pp. 1–17.
17. Stoica, I.; Abdel-Wahab, H.; Pothen, A. A microeconomic scheduler for parallel computers. In Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, Santa Barbara, CA, USA, 25 April 1995; pp. 200–218.
18. Stoica, I.; Pothen, A. A robust and flexible microeconomic scheduler for parallel computers. In Proceedings of the 3rd International Conference on High Performance Computing (HiPC), Trivandrum, India, 19–22 December 1996; pp. 406–412.
19. Islam, M.; Khanna, G.; Sadayappan, P. *Revenue Maximization in Market-Based Parallel Job Schedulers*; Technical Report; Ohio State University: Columbus, OH, USA, 2008; pp. 1–13.
20. Damodaran, P.; Vélez-Gallego, M.C. A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times. *Expert Syst. Appl.* **2012**, *39*, 1451–1458. [CrossRef]
21. Mehta, M.; Soloviev, V.; DeWitt, D.J. Batch scheduling in parallel database systems. In Proceedings of the IEEE 9th International Conference on Data Engineering, Vienna, Austria, 19–23 April 1993; pp. 400–410.
22. Grigoriev, A.; Sviridenko, M.; Uetz, M. Unrelated Parallel Machine Scheduling with Resource Dependent Processing Times. In Proceedings of the Integer Programming and Combinatorial Optimization: 11th International IPCO Conference, Berlin, Germany, 8–10 June 2005; pp. 182–195.
23. Bouganim, L.; Fabret, F.; Mohan, C.; Valduriez, P. Dynamic query scheduling in data integration systems. In Proceedings of the 16th International Conference on Data Engineering, San Diego, CA, USA, 28 February–3 March 2000; pp. 425–434.
24. Ngubiri, J.; van Vliet, M. A metric of fairness for parallel job schedulers. *Concurr. Comput. Pract. Exp.* **2009**, *21*, 1525–1546. [CrossRef]
25. Arpaci-Dusseau, A.; Culler, D. Extending Proportional-Share Scheduling to a Network of Workstations. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, NV, USA, 30 June–3 July 1997.
26. Ferguson, D.; Nikolaou, C.; Sairamesh, J.; Yemini, Y. *Economic models for allocating resources in computer systems. Market-Based Control: A Paradigm for Distributed Resource Allocation*; World Scientific: Singapore, 1996; pp. 156–183.
27. Kuwabara, K.; Ishida, T.; Nishibe, Y.; Suda, T. An Equilibratory Market-Based Approach for Distributed Resource Allocation And Its Applications To Communication Network Control. In *Market-Based Control: A Paradigm for Distributed Resource Allocation*; World Scientific: Singapore, 1996; pp. 53–73.
28. Chun, B.N.; Culler, D.E. A Malleable-Job System for Timeshared Parallel Machines. In Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, Washington, DC, USA, 21–24 May 2002; p. 30.
29. Sairamesh, J.; Ferguson, D.F.; Yemini, Y. An approach to pricing, optimal allocation and quality of service provisioning in high-speed packet networks. In Proceedings of the INFOCOM'95, Boston, MA, USA, 2–6 April 1995; pp. 1111–1119.
30. Yeo, C.S.; Buyya, R. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Softw. Pract. Exp.* **2006**, *36*, 1381–1419. [CrossRef]
31. Islam, M.; Balaji, P.; Sabin, G.; Sadayappan, P. Analyzing and Minimizing the Impact of Opportunity Cost in QoS-aware Job Scheduling. In Proceedings of the 2007 International Conference on Parallel Processing (ICPP 2007), Xi'an, China, 10–14 September 2007; p. 42.
32. Mutz, A.; Wolski, R. Sc-International Conference for High Performance Computing. Efficient auction-based grid reservations using dynamic programming. In *2008 SC-International Conference for High Performance Computing, Networking, Storage and Analysis*; IEEE: Piscataway, NJ, USA, 2008; pp. 1–8.
33. Mutz, A.; Wolski, R.; Brevik, J. Eliciting honest value information in a batch-queue environment. In *2007 8th IEEE/ACM International Conference on Grid Computing*; IEEE: Piscataway, NJ, USA, 2007; pp. 291–297.
34. Lavanya, M.; Shanthi, B.; Saravanan, S. Multi objective task scheduling algorithm based on sla and processing time suitable for cloud environment. *Comput. Commun.* **2020**, *151*, 183–195. [CrossRef]
35. Muter, İ. Exact algorithms to minimize makespan on single and parallel batch processing machines. *Eur. J. Oper. Res.* **2020**, *285*, 470–483. [CrossRef]
36. Jia, Z.; Yan, J.; Leung, J.Y.; Li, K.; Chen, H. Ant colony optimization algorithm for scheduling jobs with fuzzy processing time on parallel batch machines with different capacities. *Appl. Soft Comput.* **2019**, *75*, 548–561. [CrossRef]
37. Li, S. Parallel batch scheduling with inclusive processing set restrictions and non-identical capacities to minimize makespan. *Eur. J. Oper. Res.* **2017**, *260*, 12–20. [CrossRef]
38. Josephson, J.; Ramesh, R. A novel algorithm for real time task scheduling in multiprocessor environment. *Clust. Comput.* **2019**, *22*, 13761–13771. [CrossRef]
39. Ying, K.-C.; Lin, S.-W.; Cheng, C.-Y.; He, C.-D. Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems. *Comput. Ind. Eng.* **2017**, *110*, 413–423. [CrossRef]

40. Li, M.; Lei, D. An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times. *Eng. Appl. Artifi. Intell.* **2021**, *103*, 104307. [CrossRef]
41. Lei, D.; Yuan, Y.; Cai, J.; Bai, D. An imperialist competitive algorithm with memory for distributed unrelated parallel machines scheduling. *Int. J. Prod. Res.* **2020**, *58*, 597–614. [CrossRef]
42. Rahman, H.F.; Janardhanan, M.N.; Poon Chuen, L.; Ponnambalam, S.G. Flowshop scheduling with sequence dependent setup times and batch delivery in supply chain. *Comput. Ind. Eng.* **2021**, *158*, 107378. [CrossRef]
43. Luo, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. *Appl. Soft Comput. J.* **2020**, *91*, 106208. [CrossRef]
44. Yun, Y.; Hwang, E.J.; Kim, Y.H. Adaptive genetic algorithm for energy-efficient task scheduling on asymmetric multiprocessor system-on-chip. *Microprocess. Microsyst.* **2019**, *66*, 19–30. [CrossRef]
45. Saraswati, D.; Sari, D.K.; Kurniadi, S.P. Minimizing total tardiness in parallel machines with simulated annealing using python. *Int. J. Adv. Sci. Technol.* **2019**, *29*, 645–654.
46. Vanhoucke, M.; Coelho, J.; Debels, D.; Maenhout, B.; Tavares, L.V. An evaluation of the adequacy of project network generators with systematically sampled networks. *Eur. J. Oper. Res.* **2008**, *187*, 511. [CrossRef]
47. Higgins, J.P.T.; Thompson, S.G.; Deeks, J.J.; Altman, D.G. Measuring inconsistency in meta-analyses. *BMJ* **2003**, *327*, 557. [CrossRef]