VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

SOFTWARE ENGINEERING STUDY PROGRAM

Workload Scheduling Algorithm for a Heterogeneous System of 3D Printers

Master's thesis

Author:	Artsiom Tserashkovich
Supervisor:	prof. dr. Aistis Raudys
Reviewer:	asist. dr. Vytautas Valaitis

Vilnius – 2022

SUMMARY

The research aims to solve the problem of automating the scheduling process in the additive manufacturing domain. The goal of this research is to introduce the workload scheduling algorithm that going to schedule incoming jobs based on different parameters and aspects of jobs and printers that are in a system. Printers in a system are heterogeneous, it means that each printer has a different type, possibilities and settings for executing requested printing jobs from users.

As a solution, the research provides 3 scheduling algorithms that are adapted to the additive manufacturing domain and meet specific constraints and rules of this domain. Also, each algorithm is based on the priorities mechanism that could be parametrized by users – choosing better quality against faster execution time.

As a result, each algorithm was simulated using different simulation’s data sets that described different situations that an algorithm could face during real-world scheduling. All results were analyzed and characterized. Based on the results was chosen Backfilling Queue with Previous Jobs algorithm that could be applied for scheduling workload in a heterogeneous system of 3D printers.

Keywords: scheduling, 3D printing, additive manufacturing, heterogeneous systems.

TABLE OF CONTENTS

INTRODUCTION

3D printing technology is increasingly being used in everyday life as a tool for creating objects [BSh19]. Initially, 3D printers were not particularly common equipment. But, the development of this technology was able to significantly simplify the maintenance of this equipment and it was the reason for the massive use of 3D printers, including for personal purposes. As a result of the service market, it became possible to order printing of various models on an individual basis.

When providing 3D printing services, it is rational to consider a printer factory - the system of a set of printers. It should be noted that today is hard to provide a set of printers with the same printing parameters. Each of them has its settings and printing aspects, and they have to be considered before the printing job is executed. Currently, humans analyze the printer model and, based on the model parameters, assign a print job to a suitable printer from the printer pool. When the number of models increases, the need for human resources also increases. To solve this problem it is necessary to introduce an algorithm for automatization of human scheduling work based on parameters of a model and a profile of printer in a heterogeneous system. This research work is aimed to introduce the printing workload algorithm for a printer factory and automized scheduling.

Currently, scheduling algorithms are mostly used in the Computer Science field. Nowadays CS's researches in automation scheduling of workload mostly focused on how to schedule tasks in computer systems [Min07]. The same as in the additive manufacturing domain, computer systems are could be heterogeneous and there are also scheduling algorithms for such types of systems. But these algorithms are oriented to computer hardware - on the computing characteristic of the node/processor in the system. These values are used to make the correct decision for scheduling tasks aimed to reducing execution time.

Generally, under a heterogeneous system of 3D printers is understood the system which combines a set of printers with different types, possibilities and settings for executing requested printing jobs from users. The printing scheduling algorithm possible will be based on the other list of characteristics. Besides, the main goal of the workload algorithm will be not only to speed up the printing time but also to take into account the possible quality of the resulting printed model. So printers systems and computer systems follow the same approaches but have their own goals. As a result, the development of the algorithm could be based on already implemented approaches and experience from the computer science sphere (the scheduling of the workload at computer systems), but can't be applied to the additive manufacturing domain without some rework.

To identify what exact aspects and constraints have to be solved during the development of the scheduling algorithm the additive manufacturing domain should be analyzed. Nowadays there are researchers of printing service systems and they contain general architecture, description of components and contracts that are used by components for intercommunication, but they miss the definition of scheduling workload in the system.

Additionally, already developed scheduling algorithms have to be analyzed in this research work. Some of them could be applied or a general structure of such algorithm could be reused if it is suitable for the scheduling jobs in heterogeneous systems of 3D printers.

To be applied in a heterogeneous system, an algorithm has to provide accurate decisions in the scheduling of printing jobs based on different parameters of printers in the system and incoming printing jobs. The algorithm has to choose one of the most suitable printers from the list based on printing model parameters, printer profile, and printing preferences: time and/or quality coefficient. As a result, will be provided the description of algorithms which will contain required steps and formulas presented in the algorithms. Based on this algorithm the developer will have the opportunity to develop the scheduler component for a heterogeneous system of 3D printers.

A scheduling algorithm will require a priority mechanism to make a decision on where a job will be scheduled. As was mentioned previously, the final decision about a schedule cannot be made only based on time value. The suitability of a job for a specific printer always must be considered. Additionally, a priority mechanism has to provide users a possibility to define a value of preference between a schedule with better quality (printer resolution) or faster execution of a requested printing job.

The introduction of the algorithm will be done in such way: different algorithms will be introduced and evaluated using a simulation environment. Simulations have to be as close as possible to the realworld workload that heterogeneous systems of 3D printers are faced every day. Simulations data sets have to be introduced to evaluate systems in different situations. It will show how algorithms handle different edge cases. Also, algorithms should be compared with other scheduling algorithms that are not adjusted to work in the additive manufacturing domain. Such comparison will help into understanding how priority mechanism consider different aspects of 3D printing.

Goal and Tasks

The goal of the research is to introduce the workload scheduling algorithm for a heterogeneous system of 3D printers that will automate current manual scheduling process and will schedule (assign execution place) incoming jobs based on job’s and printer's parameters.

Tasks that have to be done to achieve research goal:

1. Define the structure of components and mathematical models based on researches of printing service systems.
2. Analyze the domain to identify aspects and constraints of scheduling in the additive domain.
3. Analyze researches related to scheduling algorithms for overview of different implementations.
4. Develop priority scheduling mechanism that will calculate priority based on users preference of better quality or faster execution time for jobs schedules.
5. Introduce and describe different versions of workload scheduling algorithms that will succeed constraints and use developed priority mechanism.
6. Evaluate (simulate) introduced algorithms using simulation environment, analyze simulation results and based on analysis recommend one of the algorithm.

1. Overview of printing service system

The scheduling algorithm for the additive manufacturing domain will be used by the printing job's scheduler within a printing service system. The development of the scheduling algorithm starts with analyzing printing service systems. The definition of a printing service system contains such information as what components in a system, the way they interact with each other and where the scheduler will be placed. As a result, the analysis of different implementations of printing service systems will provide information about the scope of a scheduler's implementation, define contracts that are used for interacting with other components, a scheduler component architecture and processes that must be supported by the scheduler.

Printing service systems don't include only printers and jobs, they are also responsible for the handling of users printing requests, matching printers with provided requirements, planning and scheduling workloads. As the printing service is the manufacturing process, the system must handle a massive amount of workload, provide fault-tolerance, support SLA (service level agreement) for user requests, and have the possibility to prioritize them [Zip01].

Currently, the additive manufacturing domain is actively researching and there are researchers that trying to unify the structure and processes of printing service systems and cover such aspects as design, model types, used software, printers characteristics, logistics, assembly, etc. But, there is still a lack of researchers that contain a detailed description of inner components of a system. The research paper with the most detailed description of the service system was chosen for continued analysis [MZT16].

1.1. Scheduler architecture

An important role in the development of the scheduling algorithm is the architecture of the scheduler where the algorithm will be applied. The architecture contains information about main actors, rules and restrictions in the communication and execution flows.

The definition of the printing service system architecture [MZT16] consists of 5 layers that are responsible for their own domain and communicate with them neighbor layers according to the defined interface. Each layer is replaceable and can be changed by other implementation, which correspond to the defined interface. In addition, every layer placed in order based on responsible domain from generic to more specific.

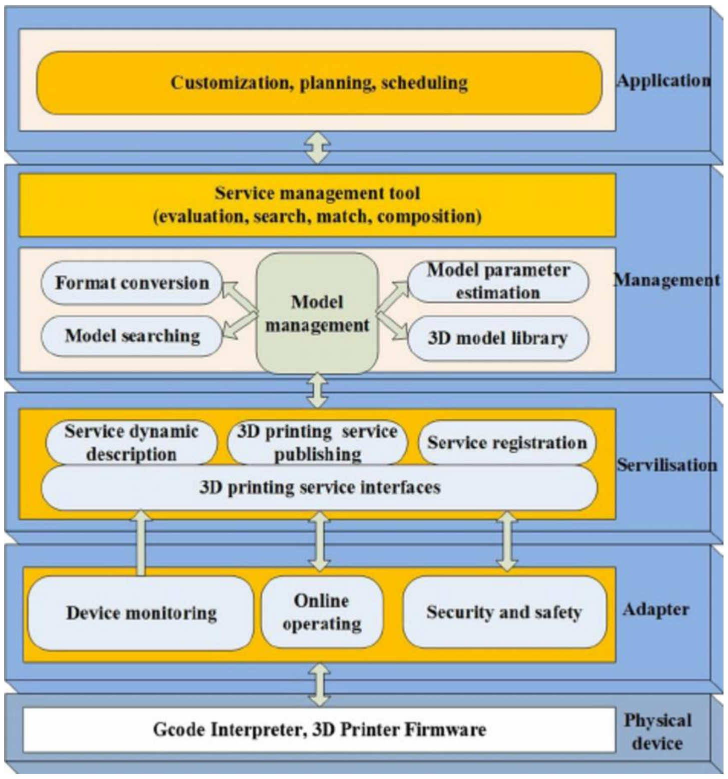
Customization, planning, scheduling
Application
Service management tool
(evaluation, search, match, composition)
Format conversion
Model searching
Model management
Model parameter estimation
3D model library
Management
Service dynamic description
3D printing service publishing
Service registration
3D printing service interfaces
Servilisation
Device monitoring
Online operating
Security and safety
Adapter
Gcode Interpreter, 3D Printer Firmware
Physical device

FIGURE 1 [MZT16]: Layers of the printing service system

The scheduler for incoming printing jobs is used in the application layer and management layers that present an interface for printing service on-demand use and include main business processes. Layers interact with servilisation and adapter layers to gather required information for support processes. The servilisation layer is responsible for service publishing and discovering in the system that could execute a user request. The adapter layer is used to gather online 3D printing services dynamic information and to support online printer operations.

In the proposed architecture the scheduler component will be placed with components from other layers on the same actors but will be decoupled by defining abstraction between them. The scheduler architecture contains such components:

1. AM (Access manager) – the interface-dependent component used for handling new incoming jobs and providing current statuses;

2. SM (Scheduling manager) – the hardware-independent component responsible for scheduling jobs;

3. MM (Machine manager) – the hardware-dependent component responsible for communication with a machine;

4. p_i – a printer that performs jobs.

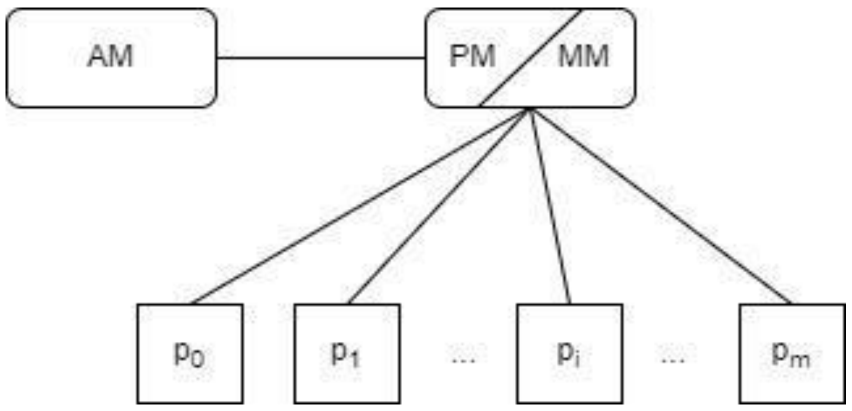

FIGURE 2: Architecture of a scheduler in a printing service system

The main responsibility of the MM is to prepare special commands/models for execution and communicate with the end printer to send commands to execute. The main goal of the AM is to interact with clients of the system to handle incoming requests and provide the required information.

In the context of this research work, MM and AM actors will not be covered, because provided algorithms will be hardware and user interface independent and could be used with any implementation of MM and AM.

1.2. Mathematical models

The SM component presented in the previous section interacts with neighbor components according to the defined interfaces. The communication is performing with 2 different sides of the system: demand and service providing – exchanging information about incoming printing request, current status of printing and printers characteristics in system.

The considered printing service system contains already defined models for print-on-demand requests and providing information about the print service [MZT16]:

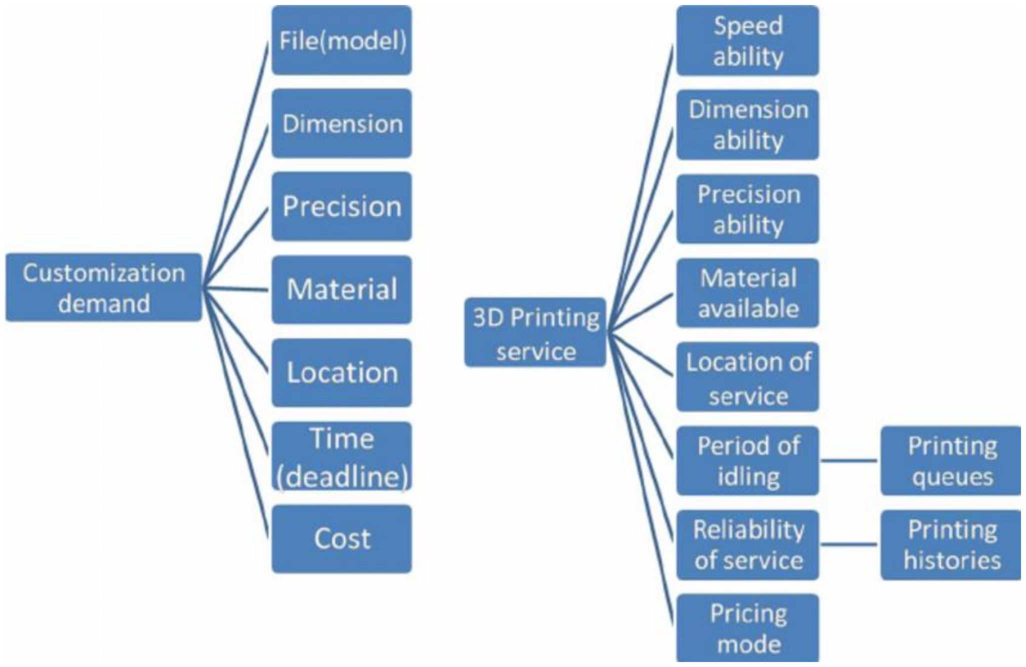

FIGURE 3 [MZT16]: Printing service system models

Provided scheduling algorithms will not require all fields from defined models. From the providing printing service side are required such fields as speed ability, precision ability and building size ability. Some of these parameters are calculated by aggregating the parameters of printers within the provider. From an incoming demand request model are required such fileds: model dimension, precision and time slot (incoming and deadline).

Based on this contracts, the mathematical models were introduced that will be used later. Main entities in system are printing job, printer and job's schedule.

Dimensions sizes of printing model and an area of printing field will be denoted as $d=<x, y, z>$, where x is size of x axis, y is size of y axis, z is size of z axis.

A printing job that comes to system for scheduling and after execution will be denoted as $j=<r, t_0, t_d, d, c>$, where r – requested resolution of incoming job, t_0 – incoming time of a job, t_d – expected deadline of a job, d – dimension sizes of printing model, c – coefficient of resolution priority against time priority of schedule.

A printer will be denoted as $p=<s, r, d>$, where s – printer's speed, r – printing resolution, d – dimension sizes of printing field.

A schedule of job i to printer j will be denoted as $s_{ij}=<j_i, p_j, t_s, t_f>$, where j_i – printing job, p_j – printer, t_s – scheduled start time, t_f – scheduled finish time.

2. Assumptions and constraints of the additive manufacturing domain

The development of the solution for scheduling jobs should starts with defining assumption and constraints in additive manufacturing domain for the future scheduling algorithm. Constraints are “hard” and cannot be defined based on the “nature” of the domain. Assumptions are “soft” and have only recommendations for the future system.

2.1. Assumptions

Below will be presented general assumptions for solutions that are not strictly characterized by the processes during additive manufacturing. These assumptions are not mandatory for additive manufacturing but were introduced to have more concrete statements which should be taken into account during the introduction of the algorithm. Assumptions:

1. Fixed count of printers.
2. Job not failed.
3. Duration of jobs execution.
4. Count of printers in system.
5. Workload volume.

2.1.1. Fixed count of printers

The first assumption for scheduling is related to the count of resources that could be used for jobs scheduling. In the scheduling algorithm, we will consider the total count of printers as a fixed value.

$$|P|_t = |P|_{t+1} \tag{1}$$

This assumption will simplify the distribution of new incoming jobs between available printers. Also, this removes the requirement for rescheduling if one of the printers will be excluded from the printing cluster.

2.1.2. Job not failed

In this research work, we will operate jobs that cannot fail due to different reasons. Such assumptions will help in development due to simplifying processes that will be performed within the scheduling algorithm.

In a real-world situation, however, the job may fail and requiring some action on the part of the operator and, in the worst case, restarting the job from scratch. This does not happen often, but still some action needs to be taken. In the future, in the next iterations, the job failing should be handled.

2.1.3. Duration of jobs execution

The scheduling algorithm will consider the estimated execution time t_{ij} of s_{ij} as the value based on volume of printing model v_i of j_i and speed of the printer $p_j.s$. Printing volume of model will be depend of dimensions of model:

$$v_i = j_i.d.x * j_i.d.y * j_i.d.z \tag{2}$$

The total estimated execution time t_{ij}:

$$t_{ij} = f(v_i, p_j.s) \tag{3}$$

In a real-world situation, execution time is estimated using not only volume and speed, but also additional settings and characteristics of printing. But in general, execution time still mostly depends on volume and printer speed.

Such assumptions will help decrease the computation, which is not directly related to scheduling. In the future, some algorithms of calculation estimated execution time could be integrated into the scheduling algorithm.

2.1.4. Count of printers in system

Partition of printers - sets of printers P, where $p_i \in P$ that have specific types and are presented in different locations. Printing service concept was reviewed previously in architecture of the literature overview [MZT16].

Choosing the printing partition will satisfy the location demand requirement and provide some not precise understanding of the time bound. This is due to the fact that the partition will have some average wait time status (parameter) and some material and cost assumptions.

We will use the considered count of printers based on some statistics of the already implemented cluster/system – usually, it is in range of 10-30 printers:

$$10 \leq |P| \leq 30 \tag{4}$$

2.1.5. Workload volume

Generally, the count of jobs in system J depends on the count of printers in the partition P. We will consider the number of current waiting jobs for one printer in the system from no more than 2 to 5:

$$|J| = c * |P|, \quad 2 \leq c \leq 5 \tag{5}$$

As printing jobs are long running tasks, such assumptions decline options when job's waiting time in pending state for long time. Additionally, the assumption correlated with real world situation, cause it rare case when job spend in pending state weeks.

2.2. Constraints

In compare with assumptions, constraints are statements that the scheduling algorithm must support. They are based on goals and analysis of additive manufacturing domain. Constraints:

1. Preference of quality against execution time.
2. Printing area constraint.
3. Deadline of printing job.
4. Single job executor.

2.2.1. Preference of quality against execution time

The difference between the usual scheduling algorithm and algorithm for additive manufacturing is that the algorithm for additive manufacturing is caring about not only the fastest execution t_{ij}, but also tries to consider the suitability between a job j_i and a printer p_j. This suitability is represented as expected resolution $j_i.r$ and actual printer resolution $p_j.r$ and affects the final quality.

The simplest way to configure this aspect is to introduce a coefficient that will be used to present impact of resolution against execution time:

$$0 \leq j_i.c \leq 1 \tag{6}$$

The scheduling algorithm for additive manufacturing has to accept coefficients to handle dynamic settings of preferring between speed execution and succeed specified expected resolution.

2.2.2. Printing area

Schedule s_{ij} cant be introduce for job j_i with dimensions of a printing that more than dimensions of printing field of a printer p_j. For all schedules S the dimensions of jobs J should be less or equal then dimensions of scheduled printers P:

$$j_i.d \leq p_j.d,\ j_i, p_j \in s_{ij} \tag{7}$$

Where comparing is of dimensions performed for 3 axes:

$$\begin{cases} j_i.d.x \leq p_j.d.x \\ j_i.d.y \leq p_j.d.y \\ j_i.d.z \leq p_j.d.z \end{cases} \tag{8}$$

That constraint has to be taken into account during scheduling because it guarantees that the model will not be out of the printing area.

2.2.3. Deadline of printing job

Additive manufacturing is oriented toward providing service for end customers. It is good practice to provide user a possibility to set a deadline $j_i.t_d$ for incoming job j_i. So after scheduling a new incoming job j_i, the algorithm should provide an response to user that shows if job could be scheduled with provided deadline or not. It could be done by comparing scheduled finish time $s_{ij}.t_f$ and job deadline $j_i.t_d$.

2.2.4. Single job executor

The algorithm will handle each incoming job j_i as a job that could be scheduled only to one printer at the same time. In other word job j_i could have only one schedule s_{ij} simultaneously.This will simplify the process of scheduling jobs and introduce a clearer definition of the work performed.

3. Schedulers types

The scheduling type is one of the characteristics of scheduling algorithms and plays the major role in the developing of the algorithm – different algorithms types have different results of the scheduling and different inner structures. Before introduction of the scheduling algorithm, must be chosen the type of scheduler, where the algorithm will be used. The main types are: Queuing and Planning [SKK03].

Researches from the HPC domain could be used for analyzing descriptions and metrics of presented algorithm’s types. Schedulers plays a major role in HPC (High performance computing) domain as distributing jobs among executor actors is the main theme of researches in this domain. Scheduling algorithms in the additive manufacturing domain have the same aspects - they distribute jobs between shared resources.

Previously, most HPC systems operated by RMS (Resource Management Systems) were based on the queuing type, as reservation and future planning were not required. Absent of advanced reservation feature block implementation of such features as diffuse requests, automatic duration extension, service level agreements, etc. In addition, supercomputers become more and more heterogeneous in their architecture and configuration, and previous resource management systems are often not flexible enough to reflect these changes. As a result, a lot of researches have been done over the last decade to improve scheduling strategies [FRu96, FRu98] and the list of RMS’s scheduling algorithms types has been extended by planning type.

3.1. Queue scheduler type

Queuing schedulers try to utilize currently free resources with waiting resource requests from queues. Future resource planning is not done for all waiting requests. Therefore, waiting resource requests have no proposed start time [SKK03].

Today almost all resource management systems fall into the class of systems that use queue schedulers. The structure of queue schedulers can be represented as several queues with different restrictions on the number of requested resources and duration that exist for submitting requests for resources. Jobs within a queue are ordered according to a scheduling policy, e.g. FCFS (first come, first serve). In addition, queues might be activated only for specific times [WLM96, FJe97].

The task of a queuing system is to assign free resources to wait for requests. The highest prioritized request is always the queue head. If it’s possible to start more than one queue head, further criteria like queue priority or best fit (e. g. leaving least resources idle) are used to choose a request. If not enough resources are available to start any of the queue heads, the system waits until enough resources become available. These idle resources may be utilized with less prioritized requests by backfilling mechanisms. Queuing system does not necessarily need information about the duration of requests unless backfilling is applied.

3.2. Planning scheduler type

Planning schedulers are planning resources for the present and future, which results in an assignment of start times to all requests. Obviously, duration estimates are mandatory for this planning [SKK03]. With this knowledge, advanced reservations are easily possible.

There is no scheduling queue as it is in queue scheduler type - every incoming request is planned immediately. Also, planning systems are not restricted to the mentioned scheduling policies FCFS, SJF (shortest job first), and LJF (longest job first).

Each time a new request is submitted or a running request ends before it was estimated to end, a new schedule has to be computed - replanning. However, with FCFS, the replanning process is not necessary, as new requests are simply placed as soon as possible in the schedule without discarding the current schedule. Obviously, some sort of backfilling is implicitly done during the replanning process. The same approach with backfilling could be used in this type of algorithms. But planning schedulers also have drawbacks in compare with queue typed schedulers - a cost of scheduling is higher than in queuing systems.

The scheduling result of planning scheduler could be graphically represented as a rectangle in a executor/time space and current scheduling profile of a system will be presented as a grid on scheduling jobs – walltime diagram.

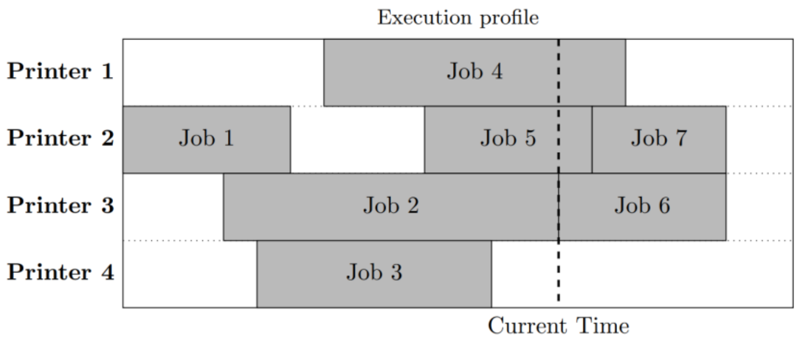

FIGURE 4: Representation of planning system scheduling result

At Figure 4 you can see that requests are placed as soon as possible in the current schedule. But they might be placed in front of already planned requests. However, these previously placed requests are not delayed (i. e. planned at a later time), as they already have a proposed start time assigned.

3.3. Characteristic of scheduling

In order to choose the most suitable type of scheduler for additive manufacturing domain, the list of general characteristics was prepared and domain for this characteristics was analyzed. Characteristics:

1. Variable or fix time reservations.
2. Resource reclaiming.
3. Automatic duration extension.
4. Partition and job flexibility.
5. Workload knowledge.

This list of characteristics is helping to choose the structure and type of algorithm that are more suitable from different already described options and introduces additional constraints, which should be followed during the development of the algorithm.

3.3.1. Variable Time Reservation

Time reservation characteristics [SKK03] of jobs show that can a job be shifted on the time axis or not. The term variable-time request stands for a resource request which can move on the time axis to earlier or later time (depending on the used scheduling policy). On the other hand, a fix-time request denotes a reservation - it cannot be shifted on the time axis.

Fix and variable time requests have the same structure of data. They come with information about the number of resources, the duration, and the start time. Variable-time reservation requests are more flexible than fix-time requests but have some drawbacks. Assume the following scenario: if a user wants to make a resource reservation as soon as possible, the request is planned according to the situation when the reservation is submitted. In good case, the job will be planned earlier than estimated, the reservation will not be delayed. In the worst case, the job will be executed after the job was planned and as a result, the user will receive the result after initial scheduled time.

This situation could be described using model presentation. This characteristic means that the chosen printer for execution of j_i could be changes from p_m to p_n and schedules (s_{in} and s_{im}) could have different scheduled times t_s and t_f:

$$\begin{cases} s_{in}.t_s \neq s_{im}.t_s \\ s_{in}.t_f \neq s_{im}.t_f \end{cases} \quad (9)$$

Variable Time Reservation will help to achieve a better quality of printing because some jobs can be shifted and rescheduled to use more suitable printer. Thus, more jobs will have more options for the printer executor.

On the other hand, some metrics should be introduced to measure the duration of allowed postponed deadline time of jobs. This can be handled with the introduction of priorities of each job, which should be used for making decisions during scheduling jobs.

3.3.2. Resource Reclaiming

Space-sharing is commonly applied to schedule HPC (High Performance Computing) applications because the resources are assigned exclusively. Parallel applications (especially from the domain of engineering technology) often traverse several phases (e.g. computation, communication, or check pointing) requiring different resources [SKK03]. In some cases, the application (temporarily) switches to another resource in order to speed up execution and improve fault- tolerance. This would increase the overall utilization of the system. It is also helps to manage jobs with a deadline. Such applications are called malleable or evolving [FRS03] and should be supported by a management system.

Resource reclaiming is not supported in the management scheduling system, due to the “nature” of additive manufacturing, switching context is a difficult process and does not add a huge impact on execution. Reclaiming in additive manufacturing is a manual operation that requires some actions from an operator. Additionally, it is not possible to resume jobs that were started on other printers. Preemption also not be considered in the scheduling algorithm because it depends on resource reclaiming which is not supported.

Once a job is initiated it will run to completion while holding all assigned resources throughout its execution.

3.3.3. Automatic Duration Extension

Estimating runtime of the job is a well-known problem [MFE01, SFT99]. It is annoying if a job is aborted shortly before termination because the results are lost and the resources were wasted. Hence users tend to overestimate their jobs by a factor of at least two to three [Str02] to ensure that their jobs will not be aborted. A simple approach to help the users is to allow extending the runtime of jobs while they are running [SKK03]. This might solve the problem, but only if the schedule allows the elongation (i. e. subsequent jobs are not delayed).

The execution time of printing orders in an additive manufacturing system can be delayed due to different reasons. Most of them are related to the physical world because jobs are executed in the physical environment.

The main reason for changing the execution time of jobs is the inaccuracy in the estimation of their duration because jobs are executed on different printers with different characteristics. The other reason related to different freezes in the printer system caused a delay of the finish execution time. This has a little impact on the total execution in comparison with planned execution time but anyway should be taken into account.

3.3.4. Partition and job flexibility

Each parallel job is executed in a partition that consists of a number of shared resources [FRS03]. The size of such a partition may depend on the type of system, the application, and the workload of the system [FRu96]. Moreover, the size of the partition of a specific job may change during the lifetime of this job. Types of partitions: fixed, variable, additive and dynamic.

Many other researchers use the variable partitioning paradigm, in which each job requires a specific number of processors but can be scheduled on any subset of processors of the system.

However, once a partition for a job has been selected its size cannot change anymore. Finally, in dynamic partitioning, the size of a partition may change at runtime.

When a printing request comes to the system, it must have characteristics of workload for printers. A job description contains the count models to print. As one printer can print only one model due to the process of printing and exact count of models specified in the job model, the job partition is characterized as a variable partition.

For the same reason, the number of consumed resources is defined value and in this case, jobs can be characterized as rigid jobs (the number of resources assigned to a job is specified external to the scheduler, and precisely that number of resources are made available to the job throughout its execution [FRS03]). However, execution characteristics can be changed based on the execution printer.

3.3.5. Job and Workload Knowledge

Incoming jobs differ in the type, quantity, and accuracy of the information available to and used by the scheduler. Characteristics of individual jobs are useful in scheduling. Workload information is also useful in choosing a scheduling policy [FRS03]. The knowledge available to the scheduler can be at one of the following levels:

1. None - no prior knowledge is available or used in scheduling, so all jobs are treated the same upon submission;

2. Workload - knowledge of the overall distribution of service times in the workload is available, but no specific knowledge about individual jobs. Again, jobs are treated the same, but policy attributes and parameters can be tuned to the workload;

3. Class - each submitted job is associated with a class, and some key characteristics of jobs in the class are known;

4. Job - the execution time of the job on any given number of resources is known exactly.

Job knowledge, which is defined to be exact, is unrealistic in practice. However, assuming omniscience in modeling studies makes it possible to obtain an optimistic bound on performance that is not achievable in practice. Assuming job knowledge in modeling studies sets the performance standard against which practically realizable scheduling algorithms, which use class knowledge at most, can be compared.

Incoming printing jobs have the same type, accuracy of information, structure, and descriptions, but each job has its own volume of required resources. The required precisions, speed, printing volume, and other characteristics are different for each incoming job.

Based on this input information, the knowledge of incoming jobs for scheduling algorithms is class type knowledge. This type is usually for scheduling algorithms because more precise knowledge of jobs is unrealistic and used only in theoretical works.

3.4. Comparing queue and planning types

Queuing systems try to utilize currently free resources with waiting for resource requests. Future resource planning is not done for all waiting requests. Therefore, waiting resource requests have no proposed start $s.t_s$ and finish $s.t_f$ time. The task of a queuing system is to assign free resources to wait for requests. The highest prioritized request is always the queue head. If not enough resources are available to start any of the queue heads, the system waits until enough resources become available. Queuing system does not necessarily need information about the duration of requests unless backfilling is applied.

Planning systems have similar flows and criteria as a queuing system, but they have some differences in other aspects. First of all, they plan resources for the present and the future, which results in the assignment of a start time for all requests. In addition, planning systems have a list of requirements that should be achieved to be implemented for some systems. List of requirements for a planning scheduler:

1. Possibility to estimate duration time for the incoming task.
2. Possibility to claim required resource without human intervention.
3. Jobs should contain a set of attributes required to scheduling and calculation estimated duration.

Both types contain such features which useful for additive manufacturing:

1. Replanning (backfilling).
2. Support custom priority policy.

To compare both types and choose which one is more suitable for this task, below presented at Table 1 with the assumptions and constraints from the previous part.

Table 1: Comparison Planning and Queuing systems

Characteristic name	**Planning systems**	**Queuing systems**
Characteristics		
variable or fix time reservations	+	N/A
resource reclaiming	+	+
automatic duration extension	+	+
parition and job flexibility		
workload knowledge	+	+
Assumptions		
fixed count of printers	+	+
job not failed	+	+
duration of jobs execution	+	N/A
count of printers in system	+	+
workload volume	+	+
Constraints		
preference of quality against execution time	+	-
printing area constraint	+	+
deadline of printing job	+	-
single job executor	+	+

Based on the results of the comparison, the Planning systems type was selected for the scheduling system for additive manufacturing.

4. Scheduling priorities

The main requirement for automated scheduling is the possibility to make a decision for a better place of schedule without human intervention, that why algorithm should have proper related to the domain priority mechanism.

Each printer p_j and each printing job j_i have their own printing parameters which could be considered as parameters for making a decision. One printer is more suitable for execution for a task, another not. The most suitable decision-making mechanism is a scheduling policy based on custom priority policy specific to the additive manufacturing domain.

The priority of a schedule $s_{ij}.pr$ will be calculated based on a two component schedule parameters. The first will be the resolution (precision) parameter pr_r of printers and the second is the time of waiting and execution - the duration time priority pr_t. As a result the definition of a schedule in priority scheduling algorithm will be extend with pr_r , pr_t and pr: $s_{ij}=<j_i, p_j, t_s, t_f, pr, pr_r, pr_t>$

The total priority of schedule $s_{ij}.pr$ will be calculated using $s_{ij}.pr_r$ and $s_{ij}.pr_t$. The impact $s_{ij}.pr_r$ against of $s_{ij}.pr_t$ will be prioritized by coefficient parameter $j_i.c$:

$$s_{ij}.pr = (j_i.c * s_{ij}.pr_r) + ((1 - j_i.c) * s_{ij}.pr_t), \text{where } s_{ij}.pr_r \in [0, 1]\ , s_{ij}.pr_t \in [0, 1] \qquad (10)$$

The lower value of priority $s_{ij}.pr$, the worst schedule of job j_i to printer p_j we have. These components are not presented as a set of constant values, the scheduler will compute them dynamically based on the current context.

4.1. Resolution priority

The set of linear functions with configuration parameters specific to the current context will be used for calculating the resolution (precision) priority $s_{ij}.pr_r$. Calculation of $s_{ij}.pr_r$ is performed in scope of all suitable printers P_i that could execute job j_i. The P_i introduced based on hard constraints of jobs. Parameters of the set of linear functions present the current context of the printing system for printing job j_i:

1. T_{exp} - the threshold between lower/higher resolution and excepted, constant value and introduced at system initialization.
1. R_{MIN} - the minimum value of printer resolution $p_j.r$, where $p_j \in P_i$.
2. R_{MAX} - the maximum value of printer resolution $p_j.r$, where $p_j \in P_i$.

The incoming job j_i for scheduling itself contains the desired value of the print's resolution $j_i.r$ which will be more suitable for this job. But if there will be an available printer with a better resolution $p_j.r$ it doesn't mean we should skip it. The specified desired value of precision mostly works as a lower bound for a schedule.

The set of linear functions for calculation priority $s_{ij}.pr_r$ consists of two functions - the first is responsible for calculation the priority for printer p_j with higher resolution $p_j.r$ than $j_i.r$ and the second function for calculation priority for schedule s_{ij} where printer has lower resolution $p_j.r$ than $j_i.r$. Calculation of resolution priority is presented as operation $rp(j_i, p_j, R_{MIN}, R_{MAX})$:

$$rp(j_i, p_j, R_{MIN}, R_{MAX}) = \begin{cases} \dfrac{j_i.r - p_j.r}{j_i.r - R_{MIN}} * T_{EXP} + (1 - T_{EXP}), & if\ p_j.r \in [R_{MIN}, j_i.r) \\ 1 - T_{EXP}, & if\ p_j.r = j_i.r \\ \dfrac{R_{MAX} - p_j.r}{R_{MAX} - j_i.r} * T_{EXP}, & if\ p_j.r \in (j_i.r,\ R_{MAX}] \end{cases} \tag{11}$$

To calculate resolution priority $s_{ij}.pr_r$ for new schedule s_{ij} with job j_i and printer p_j that has resolution $p_j.r$ we should provide $p_j.r$ to the presented functions.

The parameter T_{EXP} is a global constant that does not change during the scheduling and execution processes. R_{MAX} and R_{MIN} parameters have the scope of printers P_i that means these parameters could be calculated after scheduler filter all printers in partition P for availability for job j_i.

As all parameters have a scope of job, we can compose function only once for job j_i and reuse this composed functions for all schedule s_i for this job.

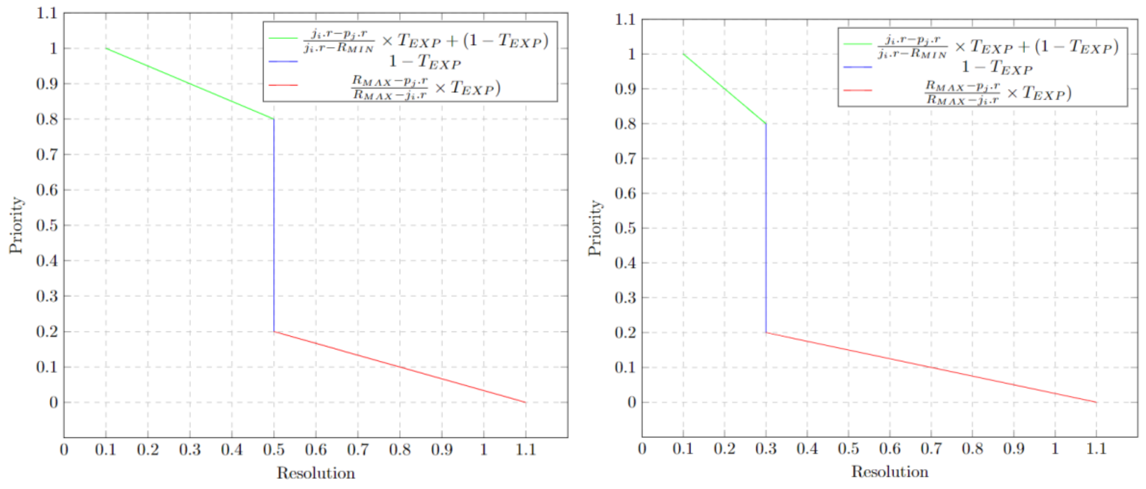

FIGURE 5: Resolution priority functions

At Figure 5 presented how linear functions change for different jobs but for some P_i. The parameters for provided printers P_i have values $T_{EXP} = 0.2$, $R_{MIN} = 0,1$, R_{MAX}=1.1 of linear functions. On the left plot presented functions for the job with parameter $j_1.r = 0.5$. On the right plot presented a job with parameter $j_2.r = 0.3$. Using this figure you can see how priority changes for printer p_j with resolutions $p_j.r = 0.4$ for jobs j_1 and j_2.

4.2. Finish time priority

The similar approach as presented for resolution priority can be used for calculating based on a finish time priority pr_t of job j_i - calculate priority by parameterized functions. Nevertheless, there are some differences in functions. In the resolution priority case pr_r, we had a desired resolution value $j_i.r$, which was the reason for introducing 3 different functions: 2 handled edge cases and one handled the expected value.

Additionally, in the case of finish time priority pr_t calculation, we have such an assumption - the calculation isn't based on the expected value of duration. Parameters that are required for priority calculation operation $tp(j_i, p_j, T_{MIN}, T_{MAX})$:

1. T_{MIN} - the minimum finish time $s_{im}.t_f$ for the job j_i.
2. T_{MAX} - the maximum finish time $s_{im}.t_f$ for the job j_i.

In the previous chapter, we introduce requirements for each priority - they should be in a range from 0 to 1: $pr_t \in [0, 1]$.

Many function types are suitable for the calculation. The first type is the linear function, which distributes priority evenly from minimum time to maximum finish time $s_{ij}.t_f$ of schedule s_{ij} for job j_i and pinter p_j:

$$tp(s_{ij}, T_{MIN}, T_{MAX}) = 1 - \frac{s_{ij}.t_f - T_{MIN}}{T_{MAX} - T_{MIN}}, \text{where } s_{ij}.t_f \in [T_{MIN}, T_{MAX}] \tag{12}$$

The second type of function suitable for calculation of time priority is the cosine type of functions. The central value in such type of function is shifted to a lower priority value that means priority in the first part of the finish time $s_{ij}.t_f$ (from minimum to middle value duration) will decrease faster than in the previous type. In the second part (from middle to maximum value) the priority $s_{ij}.pr_t$ will grow slower:

$$tp(s_{ij}, T_{MIN}, T_{MAX}) = cos(\frac{s_{ij}.t_f - T_{MIN}}{T_{MAX} - T_{MIN}} * {}^{\pi}/_2 + {}^{\pi}/_2) + 1, \text{where } s_{ij}.t_f \in [T_{MIN}, T_{MAX}] \quad (13)$$

The rate of change (acceleration of the function) can be increased if the result will be squared:

$$tp(s_{ij}, T_{MIN}, T_{MAX}) = (cos(\frac{s_{ij}.t_f - T_{MIN}}{T_{MAX} - T_{MIN}} * {}^{\pi}/_2 + {}^{\pi}/_2) + 1)^2, \text{where } s_{ij}.t_f \in [T_{MIN}, T_{MAX}] \quad (14)$$

The plots of 3 functions are presented at Figure 6. The parameters for function are T_{MIN}=10, T_{MAX}= 100.

The next type of calculator priority function is a sinusoidal type of function. In comparing with the cosine type of functions, this type is characterized by shifting priority in the middle to the highest value of priority:

$$tp(s_{ij}, T_{MIN}, T_{MAX}) = sin(\frac{s_{ij}.t_f - T_{MIN}}{T_{MAX} - T_{MIN}} * {}^{\pi}/_2 + {}^{\pi}/_2), \text{where } s_{ij}.t_f \in [T_{MIN}, T_{MAX}] \quad (15)$$

However, when the result of the *sin* function is squared we can notice first part has shifted to highest priority, but the second part is shifted to lowest priority.

$$tp(s_{ij}, T_{MIN}, T_{MAX}) = sin(\frac{s_{ij}.t_f - T_{MIN}}{T_{MAX} - T_{MIN}} * {}^{\pi}/_2 + {}^{\pi}/_2)^2, \text{where } s_{ij}.t_f \in [T_{MIN}, T_{MAX}] \quad (16)$$

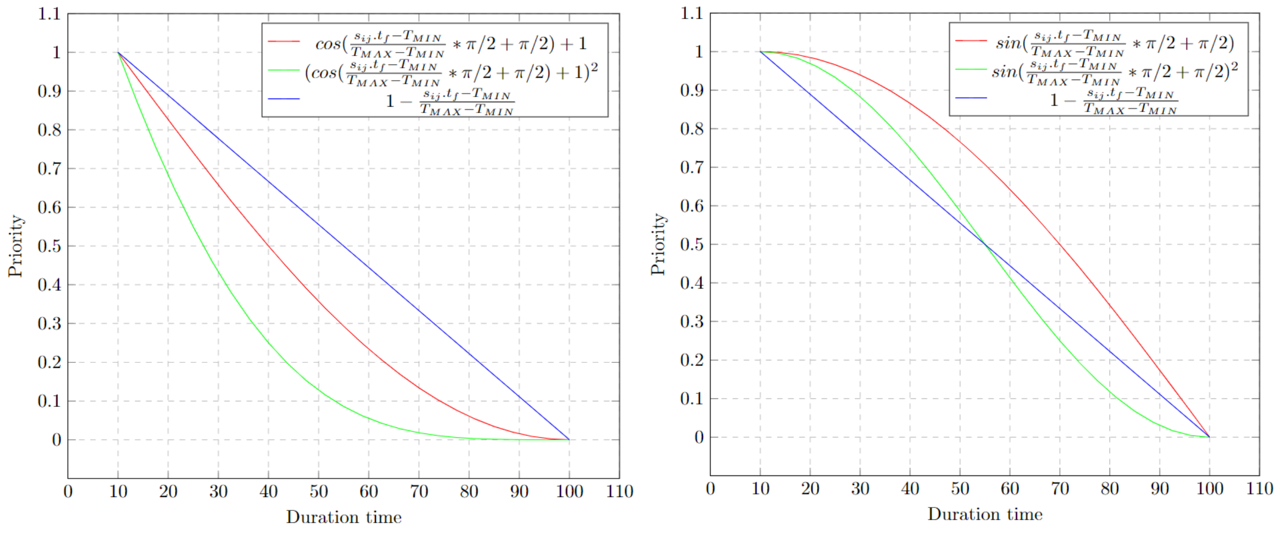

FIGURE 6: Finish time priority functions

5. Scheduling algorithms

5.1. Unprioritized least finish time stacking algorithm

Unprioritized least finish time stacking or ULFS scheduling algorithm is a first scheduling algorithm for the additive domain introduced in this research paper. Unprioritized keyword in the name of algorithm means that algorithm uses only finish time during scheduling and doesn't take into account resolution factor. A resolution compatibility could be covered by introduction a strict filter rule for printers that are will be considered as an executor for a scheduling job. But such strict rule not acceptable for scheduling algorithm due to previously described Contstraint 1 – that why this algorithm is not under consideration as a final algorithm of this research work and will be used only for comparing with the next algorithms.

ULFS is based on the general planning scheduling algorithm with some adaptation for the domain. As a result the structure of components that used in the algorithm will be similar:

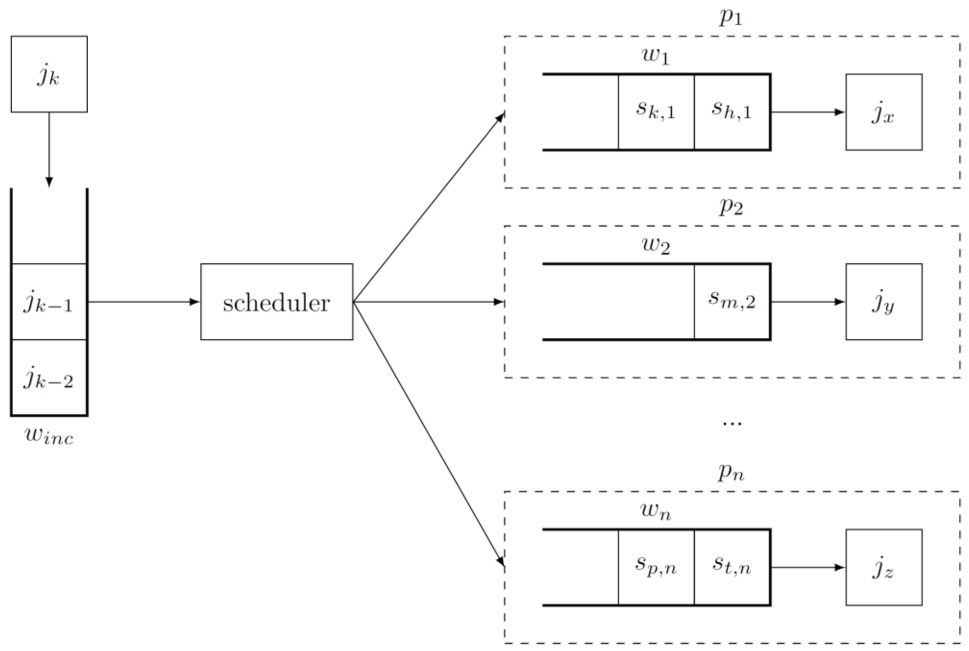

FIGURE 7: Structure of ULFS scheduling algorithm

The w_{inc} queue contains incoming jobs that scheduler has to schedule. The job at head of w_i queue schedules by calling *schedule(j_i)* operation that is exposed for external calling by a scheduler. As a result of scheduling operation scheduler enqueue a schedule to one of a printer's queue *w*. A printer queues are used for storing schedules in order by $s_{ij}.t_s$ and when a printer finished a current job, it took next from queue.

As mentioned previously all incoming job j_k come to incoming queue w_{inc} where order by incoming time – as a result all schedules stacks after previous schedules in a printer queue w_j. Due to this behavior we can characterized algorithm as continues stacking scheduling algorithm.

After the calling *schedule(j_i)* operation, the scheduler filter all printer in system *P* by applying scheduling compatibility rules that are parameterized by j_i to get a list of printers P_i that are compatible to execute j_i:

$$P_i = \{p: P | compatible(p, j_i)\} \tag{17}$$

For this algorithm the compatible rules are:

1. Fitting of model into printer's printing area rule (Contraint 2).
2. Printer resolution $p_j.r$ not less than job expected resolution $j_i.r$ rule.

The second compatible printer rule is a strict analog of Constraint 1. It's required as preventing mechanism for always succeed user request.

For example purpose the P_i will consist such printers: P_i = { p_k, p_j, p_x, p_y }.

The next steps is generate all possible "stacked" schedules used previously filtered list of printers P_i. The set of all possible "stacked" schedules using printers P_i will be notated as S_i. To introduction S_i that we need to calculate execution period for all free next slots of printers (operation *nextslot(p)* will be used for getting next free slot start time) – the start time $s_{ji}.t_s$ and finish time $s_{ji}.t_f$. The duration (p, ji) operation will be used for calculating estimated job duration:

$$S_i = \{p_j: P_i \bullet s_{ij} = \{t_s = nextslot(p_j), t_f = nextslot(p_j) + duration(p_j, j_i)\}\} \tag{18}$$

At Figure 8 presented example how schedules placed for list of printers . As you can notice, a schedule could be after current execution job or previous last printer's schedule - S_i = { s_{ik}, s_{ij}, s_{iy}, s_{iy} }.

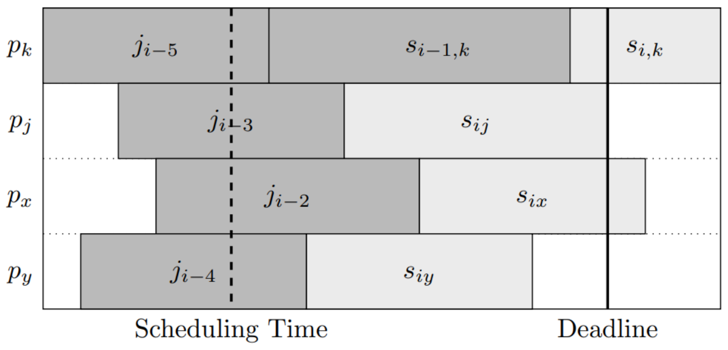

FIGURE 8: Walltime with schedules S_i

In addition, not all schedules for jobs j_i fitted into requested deadline $j_i.t_d$ - such schedules will be removed from a list S_i due to Constraints 3. If all schedules will not fit into requested deadline $j_i.t_d$ – the scheduling result will be considered as unsuccessful and user will be notified about it. As a result, after removing all schedules from S_i that not succeed our requested deadline time $j_i.t_d$, the list will be: $S_i = \{ s_{ij}, s_{iy} \}$.

The next step is to choice a final schedule s_i from a list S_i. It is chosen based on the finish time $s_{ji}.t_f$, in considered example s_i=s_{iy} as it has least finish time $s_{iy}.t_f$. The algorithm choose a schedule with the least finish time:

$$s_i = \min_{s_{ij} \in S_i} s_{ij}.t_f \tag{19}$$

The last step is to add chosen schedule s_i to scheduled printer's queue w_j.
The description of the ULFS scheduling algorithm:

Algorithm 1 ULFS scheduling algorithm

```
Input: job $j_i$
Algorithm:
1: Filter printers in system $P$ by applying $compatible(p, j_i)$;
2: Introduce all possible "stacked" schedules $S_i$ for printers $P_i$;
3: Filters schedules $S_i$ by fitting into requested deadline;
4: Chose schedule $s_i$ by minimum least scheduled finish time;
5: Add a final schedule $s_i$ to the queue $w_j$ of a chosen printer.
```

5.2. Prioritized stacking algorithm

Prioritized stacking or PS scheduling algorithm is the first version of the algorithm presented in this research paper that succeed all previously introduced constraints and assumptions. The main difference with ULFS is that PS chooses a final schedule based on a schedule's priority $s_{ij}.pr$ instead of choosing a schedule with the least finish time $s_{ij}.t_f$.

As it was in ULFS, PS filtering printers P for incoming jobs j_i using compatible rules and introducing the list of possible schedules S_i. But PS algorithm has a different set of rules – only fitting of model into printer's printing area rule (based on Constraint 2) is left. This is due to the presence of priority calculation in this algorithm - the requested resolution will be a part of the priority component. As a result the printer resolution $p_j.r$ not less than the job expected resolution $j_i.r$ rule will not be used in PS algorithm.

Let suppose that the list of printers P_i after applying compatible rules will be the same as it was in the ULFS algorithm example: $P_i = \{ p_k, p_j, p_x, p_y \}$. Also, lets the current execution context (current printing jobs and schedules in queues) will be the same.

PS algorithm uses the same mechanism for introducing all possible schedules S_i for incoming job j_i – iterating over printers P_i and using operation *nextslot(p)* and *duration(p, j_i)* for the introducing schedules for all printers p_j from P_i. As parameters and approach for introducing schedules S_i are the same as they were in ULFS, the walltime of schedules for PS algorithm will be the same as it was in PS algorithm:

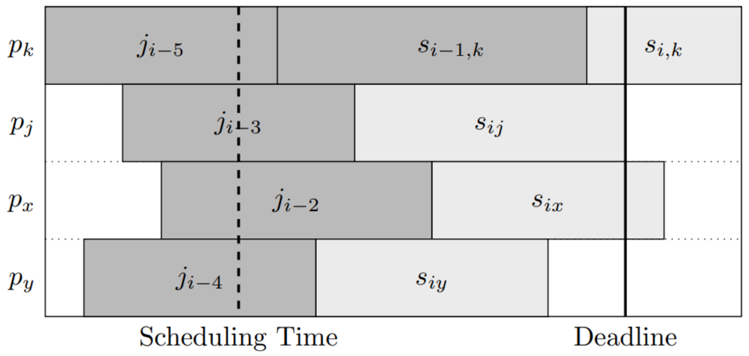

FIGURE 9: Walltime with schedules S_i

PS algorithm also filters all schedules S_i for provided deadline $j_i.t_d$ - as a result the S_i after filtering will be - $S_i = \{ s_{ij}, s_{iy} \}$.

In comparison with ULFS, the next step in PS will be different – before choosing a final schedule, the algorithm will calculate time priority and resolution priority to all schedules from Si. Before calculating priority the algorithm has to define parameters $T_{MIN}, T_{MAX}, R_{MIN}, R_{MAX}$. The requested deadline of job $j_i.t_d$ will be used as T_{MAX} and the scheduling time will be used as T_{MIN}. R_{MIN}

and R_{MAX} will be taken from list of printers P_i as maximum and minimum values of $p_j.r$. The formulas for calculating and assigning of priorities will be:

$$\begin{gathered} T_{MIN} = schedulingtime; \\ T_{MAX} = j_i.t_d; \\ R_{MIN} = \min_{r_j \in \{p_j : P_i \bullet p_j.r\}} r_j ; \\ R_{MAX} = \max_{r_j \in \{p_j : P_i \bullet p_j.r\}} r_j; \\ S_i = \{s_{ij} : S_i \bullet s_{ij}\{pr_t = tp(s_{ij}, T_{MIN}, T_{MAX}), pr_r = rp(s_{ij}.j, s_{ij}.p, R_{MIN}, R_{MAX})\}\}. \end{gathered} \tag{20}$$

The choosing formula of a final "stacked" schedule s_i in PS will be different in compare with the ULFS – a schedule will be chosen by maximum priority value $s_{ij}.pr$:

$$s_i = \max_{s_{ij} \in S_i} s_{ij}.pr \tag{21}$$

Let assume we have such parameters during the calculating priorities for s_{ij}, s_{iy} schedules: $R_{MIN} = 0.1$; $R_{MAX} = 0.5$; $j_i.r = 0.2$; scheduling time $= 5$; $j_i.t_d = 15$; $j_i.c = 0.7$. Printers parameters will be $p_j.r = 0.1$ and $p_y.r = 0.5$. Scheduled finish time of schedules are $s_{ij}.t_f = 15$ and $s_{iy}.t_f = 13$. As a result time and resolution priorities of schedules S_i will be: $s_{ij}.pr_r = 1$; $s_{iy}.pr_r = 0$; $s_{ij}.pr_t = 0$; $s_{iy}.pr_t = 0.2$. Total priority will be: $s_{ij}.pr = 0.7$; $s_{iy}.pr = 0.06$. As a result s_{ij} will be chosen as a final schedule: $s_i = s_{ij}$. The different final decision in case of PS and ULFS represents the difference in approaches of algorithms.

The last step is the same as in UFLS algorithm – FS is the adding a final "stacked" schedule s_i to the printer's queue.

The text description of the PS algorithm:

Algorithm 2 FS scheduling algorithm

Input: job j_i

Algorithm:

```
1: Filter printers in system $P$ by applying $compatible(p, j_i)$;
2: Introduce all possible "stacked" schedules $S_i$ for printers $P_i$;
3: Filters schedules $S_i$ by fitting into requested deadline;
4: Found required for priority calculation parameters $T_{MIN}$, $T_{MAX}$, $R_{MIN}$, $R_{MAX}$;
5: Calculate priorities $pr_r$ and $pr_t$ for $s_{ij} \in S_i$ using parameters $T_{MIN}$, $T_{MAX}$, $R_{MIN}$, $R_{MAX}$;
6: Chose schedule $s_i$ by maximum priority $s_{ij}.pr$;
7: Add a final schedule $s_i$ to the queue $w_j$ of a chosen printer.
```

5.3. Backfilling queue with previous jobs algorithm

Backfilling queue with previous jobs or BQPJ scheduling algorithm is based on the same schedule's priority principle as used in PS algorithm – decisions about final schedules based on the schedule's priority that represents suitability of job's scheduled placements with user's requested preference (Constraint 1).

But the difference with PS algorithm is that BQPJ algorithm is not a stacking type algorithm. That means there is no guarantee that a new job will be scheduled only after previous scheduled jobs (stacked) – a reshuffling of previously scheduled jobs J_p could be done. Under the reshuffling means changing scheduled place (start time $s_{ij}.t_s$, finish time $s_{ij}.t_f$ and printer $s_{ij}.p$) of previously scheduled jobs J_p.

At Figure X presented results of the scheduling with the reshuffling of previous jobs schedules J_p. In this example were scheduled previous jobs $J_p = \{ j_{i-4}, j_{i-3}, j_{i-2}, j_{i-1} \}$ and incoming job j_i. In the stacking type algorithm, a schedule s_i could be located only after in time axis. But in backfilling based algorithm, a schedule s_i could be placed early than previously scheduled job (only if such schedules succeed deadlines of both jobs). In this result, the schedule s_{i-2} was rescheduled with another start time $s_{i-2}.t_s$, finish time $s_{i-2}.t_f$ that giving a possibility to place schedule s_i in front of s_{i-2} (backfill).

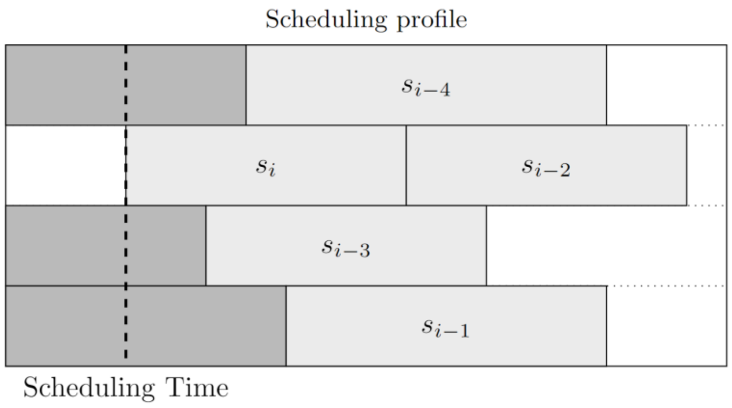

FIGURE 10: Scheduled incoming job infront of previous scheduled job.

The main goal of such backfilling is to check more schedule's places in comparison with previously described algorithms that could be considered as a final schedule. Stacked algorithms considered only schedule's places after previous schedules, but it's not always an optimal solution. Sometimes better places will be when previously scheduled job from J_p will be moved and a new schedule will take this place. But to support it, BQPJ algorithm has to move from making decision only based on one schedule to considering the priority of the whole schedules scope (previously scheduled jobs and incoming job j_i) – schedules profile PR_i. In other words, schedules are considered together, not separately.

One of the rule for backfilling is that jobs that already started execution will not be affected by the backfilling due to Characteristic 2, only jobs J_p in printer's queue w_j could be affected. In general, the backfilling could be done during the scheduling due to Characteristic 1 (variable time reservation) of the domain that was described previously.

The structure of the scheduling algorithm will be the same as it was presented in ULFS algorithm. There still will be incoming queue w_{inc}, printer's queues w_j and the scheduler. The flow of handling incoming job j_i will be similar as it was done before, except one thing – the algorithm will be assigning not only new schedule s_i to printer queue (stacked), but will rewrite all schedules from printers queues w_j if it will be required. To support that the inner structure of the scheduler has to be changed. But in general the flow will be the same – the scheduler will start scheduling after receive call *of schedule(j_i)* operation that provides a new job for schedule from the head of the incoming queue w_{inc}.

Before the defining of the algorithm, let introduce new operations and models which will be used in the description. Operation *scheduled(W)*, where W – set of printers queue: $W = \{w_1, \dots, w_n\}$. This operation is used to get all jobs that were scheduled previously, but haven't started execution yet. This operation is used every time when new job j_i comes to schedule to get J_p set:

$$J_p = scheduled(W), where\ scheduled(W) = \cup_{w_j \in W} w_j \qquad (22)$$

When jobs J_p are found, the algorithm starts generating different backfilling combinations. To describe the generating process, lets firstly define what is backfilling queue $w_{back,i}$. This queue is responsible for define the order in which job will come to *schedule(j_i)* operation of previous algorithm FS. All combinations of ordering ($w_{back,i}$ queues) are stored in a set of queues W_{back}. As a result there is such mathematical notations:

$$\begin{gathered} |w_{back,i}| = |J_p| + 1; \\ \exists!\, j \in w_{back,i} : j \in J_p, j_i; \\ W = \{w_{back,1}, \dots, w_{back,k}\}. \end{gathered} \qquad (23)$$

To define set of queues W, the BQPJ will call special operation *backfill(J_b, j_i)* that is responsible for generating different backfilled combinations W. The generating will be performed in such way: firstly the operation will prepare a base queue that will be used as a template and after that will iterate over base queue and insert j_i in each position and save it as a combination $w_{back,i}$. A base queue defines as each job j from J_p order by incoming time $j.t_i$. The inserting performs using a pointer that initially points at a head of a base queue and after each defined combination will move to the head direction. A pointer will move until it will reach out of queue scope. During the inserting of j_i into position, previous job placed here will be shifted to the tail direction of queue. As all jobs J_p must be scheduled and point will be in each position of base queue, the count of combination will be $|W| = |J_p| + 1$.

The algorithm of generating backfilled combination could presented using such example – let define the list of previous jobs as $J_b = \{ j_{i-2}, j_{i-1}\}$ and incoming job as j_i. The *backfill*(J_b, j_i) operation will return: $W = \{\{ j_{i-2}, j_{i-1}, j_i \}, \{ j_{i-2}, j_i, j_{i-1}\}, \{ j_i, j_{i-2}, j_{i-1}\}\}$.

As was mentioned previously, the decision about final schedules in BQPJ based on a scope of schedules, instead of a single schedule. Such scope will be represented as a profile of schedules PR_i – contains a set of schedules for backfilling combination $w_{back,i}$. PR_i contains field pr – priority of profile. This priority $PR_i.pr$ is calculated as a sum of schedules within the profile:

$$PR_i.pr = \sum_{s_{ij} \in P_i} s_{ij}.pr \tag{24}$$

The *schedule*$(w_{back,i})$ operation is using for the scheduling a queue with backfilled combination $w_{back,i}$ and get schedules profile PR_i for such combination as a result. This operation schedules jobs from aqueue following the order which they were introduced. A single job form queue is scheduling by calling *schedule(j)* operation for single job that was presented in FS algorithm – this operation schedules job with priorities and return schedule with maximum priority (the same FS decision mechanism). But with the same exception, at the start for each calling of *schedule*$(w_{back,i})$ operation printers queues w_j will be set to empty: $w_j = \emptyset$ - this is required to not generate schedule duplicates. And first call *schedule(j)* will operate empty queues. As a result, calculating of schedules profiles based on a backfilling combination will be:

$$PR = \sum_{w_{back,i} \in W} schedule(w_{back,i}), \text{ where } |PR| = |W|;$$
$$schedule(w_{back,i}) = \sum_{j_k \in w_{back,i}} schedule(j_k). \tag{25}$$

When a set of profiles *P* is ready, the next step of BQPJ algorithm is to choose the final profile PR_f as a source for the update of printers queues. The algorithm is choosing PR_f the using priority field $PR_f.pr$ in the same way as it was in the stacking scheduling algorithm:

$$PR = \max_{PR_i \in PR} PR_i.pr \qquad (26)$$

The final step is update update printers queues with schedules from final chosen profile PR_f.

The text description of the BQPJ algorithm:

Algorithm 3 BQPJ scheduling algorithm

Input: job j_i

Algorithm:

```
1: Define previous scheduled jobs $J_p$;
2: Generate queues $W$ with backfilled combination of $J_p$ and $j_i$;
3: Get schedules profiles $PR$ using PS algorithm;
4: Chose final schedules profile $PR_f$ by maximum priority $PR_i.pr$;
5: Update printers queues with schedule from $PR_f$.
```

5.4. Full places enumeration algorithm

Full places enumeration or FPE algorithm is another scheduling algorithm in this research work that is based on schedules profiles PR_i. The same as in BQPJ algorithm, FLE algorithm makes decisions about scheduling based on a set of schedules PRi instead of a single schedule sij - consider the whole scope of schedules, instead of considering them individually. But in comparison with BQPJ, FLE considers more possible options of PR_i due to FLE enumerates all places for each jobs for schedule (previous jobs J_p and incoming job j_i) and introducing schedules profiles PR_i for all unique combinations of places s_{ij}.

The same as previously, the algorithm is scheduling a job that is provided by calling operation *schedule(j_i)* of the scheduler – schedule new incoming job j_i that come from incoming queue w_{inc}. When j_i comes to scheduling, the algorithm first step is defining previously scheduled jobs J_p as it was in BQPJ – using *scheduled(W)* operation. After that FLE combined J_p and j_i into J_s set. J_s is a set of jobs that are going to full places enumeration scheduling.

When a set with jobs for schedule is defined J_s, the algorithm call the inner operation *schedule(J_s)*. This operation is the main part of this scheduling algorithm. It iterates over each job's possible places introduces profile PR_i for each. After such cycle the operation switch to next job from

J_s and again extends *PR* with new combination. As a result in PR we have all possible combinations of schedules that are allowed (checked previously introduce *compatible(p, j_i)* operation).

Lets consider example of the *schedule(J_s)* operation scheduling result. Lets define $J_p = \{ j_1 \}$ and j_2 is incoming job. Js in this case will be: $J_s = \{ j_1, j_2 \}$. Set of printers in system will be: $P = \{ p_1, p_2 \}$ and every printers are compatible with each job at J_s. As a result combinations of all places will be: $PR = \{\{ p_1: \{s_1, s_2\}, p_2: \{\}\}, \{ p_1: \{ s_2, s_1 \}, p_2: \{\}\}, \{ p_1: \{ s_1 \}, p_2: \{ s_2 \}\}, \{ p_1: \{ s_2 \}, p_2: \{ s_1 \}\}, \{ p_1: \{\}, p_2: \{ s_1, s_2\}\}, \{ p_1: \{\}, p_2: \{ s_2, s_1\}\}\}$. The result time complexity of combinations generating is will depend on count of jobs from J_s and count of printers *P*:

$$O(|Js|! * C(|P|, |Js|)), where\ C(|P|, |Js|) = \frac{|Js|!}{|P|!*(|Js|-|P|)!} \quad (26)$$

As a result you can noticed, the generating of all possible combinations for J_s in FLE algorithm is NP problem task.

When all possible schedule's place introduce, the next step is to check each schedule for the deadline – when any of the schedules in the profile do not succeed deadline, a combination will not be considered later. When all profiles were filtered, resolution and time priorities will be calculated using the same approach as before. A final profile PR_f is chosen by maximum priority value $PR_f.pr$.

The text description of FLE algorithm:

Algorithm 4 FLE scheduling algorithm

Input: job j_i

Algorithm:

```
1: Define previous scheduled jobs and incoming $j_i$ as $J_s$;
2: Generate all possible schedules profile combinations $PR$ for $J_s$;
3: Calculate priority for all define profiles $PR$;
4: Chose final schedules profile $PR_f$ by maximum priority $PR_i.pr$;
5: Update printers queues with schedule from $PR_f$.
```

6. Simulation

The workflow of scheduling algorithm will be simulated using special tools for simulation - the algorithm will be simulated using virtual environment that will replace actual physical printers in systems. The simulation environment provides all required contract to provide client of environments implement specific scheduling algorithms based on simulated environment. In other case there is no reason to setup environment for simulation if not all scheduling functionality will be covered.

As a result was implemented custom console simulation tool for simulate printing systems [TSC22]. The simulation environment contains list of actors with different types. There are 2 actors: printer actor $pa_i \in PA$ and management system *MS*. Printer actor characterizes by assigned printer characteristics p_j and executes scheduled jobs. Management system is *PM (planning manager)* that contains collection of schedules S and S_e and scheduler.

The actors execute actions in cycles. So cycle is a metric for duration of actions, and as a result all time variable specified in scheduling algorithm will be measure in cycles. The simulation flow for one cycle:

1. Execute actions of *MS*:
    a. Check source for any incoming job j_i.
    b. [If there is ji] Call scheduler for scheduling j_i.
    c. Update S with new schedule s_{ij}.
2. Iterate over printer actor $pa_j \in PA$ and execute their cycle actions:
    a. [If schedule s_{ij} completed] register execution result of s_{ij}.
    b. [If no schedule in execution] try to new schedule s_{ij} from *MS*.
    c. [If schedule s_{ij} in execuition state] execute printing cycle for s_{ij}.

As a result simulated will be such set of algorithms:

1. ULFS algorithm;
2. PS algorithm with linear time priority calculation:
    a. With linear time priority calculation function.
    b. With sin-based time priority calculation function.
    c. With cos-based time priority calculation function.
3. BQPJ algorithm with linear time priority calculation:
    a. With linear time priority calculation function.
    b. With sin-based time priority calculation function.

c. With cos-based time priority calculation function.

4. FPE algorithm with linear time priority calculation.

The data for simulation was prepared manually based on all assumption and constraints presemted at research work. The printers specification and workload is closed as possible to real incoming data - this will help to achieve the closest simulation result to real system.

6.1. Printers simulation set

For simulation of algorithms was prepared list of printers P with such recommendation that will help to define set the same as real printing system:

1. Resolution of printer *p.r*: the most popular resolution of printers is 0.4 and usually placed in range: $0.2 \leq p.r \leq 1$.
2. Printing speed p.s: p.s is high correlated with printing resolution *p.r*. Usually range of speed for 3D printer: $20 \leq p.s \leq 50$.
3. Printing field dimensions sizes *p.d*: for different models of 3D printers *p.d* correlated is with resolution *p.r* – the smaller value *p.r*, the smaller *p.d.*

Based on recommentations, was introduced *P*. The table with full charactics of each printer is presented at Appendix 1. Summary charactirictics of printers:

1. Count of printers: $|P| = 12$ (Assumption 4).
2. Correlation between *P.r* and *P.d*: 0.96, where *P.r* – collection of printers resolutions in *P*, P.d – collection of printing fields dimensions of printers in *P*.
3. Correlation between *P.s* and *P.r*: 1, where *P.r* – collection of printers resolutions in *P*, *P.s* – collection of printers speeds in *P*.

6.2. Jobs simulation sets

The list of recommendation to define jobs as close as possible to real system worklod:

1. Incoming time $j.t_0$: this value has high impact for decisions during scheduling. The most correctness way define this value closest to real situation is distribute values evenly in range of considered simulation time window.
2. Resolution *j.r*: values for *j.r* should be introduced based on real world examples of printing jobs. According to popular 3d models storage Thingeverse [Thi22] the most popular resolution *j.r* is 0.4 and it placed in range: $0.2 \leq j.r \leq 1$.

3. Dimensions size of model *j.d*: *j.d* correlated with resolution *j.r* – the smaller value *j.r*, the smaller *j.d*.
4. Correlation between *j.r* and *j.c*: at this stage we will consider negative correlation between these two fields – the higher expected resolution job will have, the smaller coefficient of impact resolution priority will be.

Based on recommentations, was introduced partially defined jobs data set with all specififcations except deadline $j.t_d$ and incoming time $j.t_0$. Summary charactirictics of partial specifications:

1. Count of jobs in data set – 35.
2. Correlation between *J.r* and *J.d*: 0.91, where *J.r* – collection of expected resolutions of jobs in *J*, *J.d* – collection of models dimensions of jobs in *J*.
3. Correlation between *J.r* and *J.c*: -1, where *J.r* – collection of expected resolutions of jobs in *J*, *J.c* – collection of coefficients of jobs in *J*.

Based on paritall specification were generated 3 workload simualation sets of jobs:

1. With small range deadlines $j.t_d$ and medium distribuited jobs incoming times $j.t_0$ window (Appendix 2).
2. With medium range deadlines $j.t_d$ and medium distribuited jobs incoming times $j.t_0$ window (Appendix 3).
3. With medium range deadlines $j.t_d$ and wide distribuited jobs incoming times $j.t_0$ window (Appendix 4).

Each data sets are presented at Appendicies as a table with specification of each jobs.

To define different scale of jobs deadlines ranges and distribution of jobs incoming time window were used special relatives coefficients: deadlines scale coefficient *Cd* and incoming time window offset $T_{inc\ end}$. In case of deadlines $j.t_d$, they were define based on 3 parameters: *j.d* (volume of model), *j.c* and *Cd* (the bigger *Cd* – bigger value of $j.t_d$ will be). In case of incoming time $j.t_0$, they were randomnly generated in range of window from 0 to $T_{inc\ end}$.

Coefficients of jobs data sets are: jobs set 1 – $Cd = 0.15$, $T_{inc\ end} = 5000$; jobs set 2 - $Cd = 0.6$, $T_{inc\ end} = 5000$; jobs set 3 - $Cd = 0.6$, $T_{inc\ end} = 12500$.

6.3. Results of simulations

The description of result’s summary characterisitcs that will be used to evaluated simulations for each algorithm:

1. Total run time of workload– T_{total}.
2. Average wait time of jobs – T_{wait}.
3. Total difference between requested resolution *j.r* and actual printer’s resolution *p.r* (lower better) – R_{dif}.
4. Average printer utilization – *U*.

At Table 2 presented calculated simulations characteristics for of each algorithm and jobs sets. Tables with results of each simaulation are presented at Appendixes secction – from Appendix 5 to Appendix 19.

Table 2: calculated simulations characteristics

Algorithm	T_{total}	T_{wait}	R_{dif}	*U*
Jobs set 1				
ULFS, PS (linear), PS (sin), PS (cos)	N/A	N/A	N/A	N/A
BQPJ (linear)	489420	103876	0.1	57,1%
BQPJ (sin)	851354	133929	-0.1	38,4%
BGPJ (cos)	546704	100869	0.1	51,2%
FPE	-	-	-	-
Jobs set 2				
ULFS	568113	96038	-0.3	47.6%
PS (linear)	682938	113752	-0.3	41.2%
PS (sin)	682938	118619	-0.4	41.1%
PS (cos)	649630	87233	0.1	39.9%
BQPJ (linear)	857699	156761	-1.5	39.4%
BQPJ (sin)	2198305	318748	-2.2	18.1%
BGPJ (cos)	882991	139350	-1.3	36.4%

FPE	-	-	-	-
Jobs set 3				
ULFS	582839	109749	-0.3	46.4%
PS (linear)	520323	105040	-0.5	53.9%
PS (sin)	582839	109498	-0.6	48.4%
PS (cos)	496739	103175	-0.1	52.9%
BQPJ (linear)	1024818	150442	-0.9	33.2%
BQPJ (sin)	1876697	253215	-1.3	20.6%
BGPJ (cos)	800937	141608	-0.8	40.4%
FPE	-	-	-	-

N/As in results of first data set's simulation for algorithms ULFS, PS (linear), PS (sin), PS (cos) mean that a algorithm was not able schedule incoming job with provided value of deadline $j.t_d$. The negative values of R_{dif} mean that a algorithm assign job for printer with better quality as was requested by $j.r$ value.

In case of FPE algorithm – it didn't finish the calculating of schedules places when count of jobs for schedule became 5 and more. As a result, scheduling attempts for all 3 simulation were cancelled.

RESULTS AND CONCLUSIONS

Results

1. Researches related to printing service systems were analyzed for nowadays standards that are mostly used in service systems. Based on the analysis were introduce the scope of the scheduling algorithm where it will be applied, the general structure of the scheduler and neighbor components that could be affected by a new algorithm. The main flow of communication between a system and printers was analyzed. Mathematical models of main entities in system were introduced based on these standards.
2. The additive manufacturing domain was analyzed for possible assumptions and constraints that the scheduling algorithm must succeed before the applying in the domain. The result of this analysis is described as sets of assumptions and constraints. Totally were defined 5 assumptions and 4 constraints. Assumptions helped in making decisions during implementation and choosing more suitable approaches and structures for algorithms, and constraints helped in defining boundaries and specific aspects of the scheduling of printing jobs that the algorithm has to succeed.
3. Researches of scheduling workload and specifically scheduling in High-Performance Computing were analyzed to choose the type of scheduling system. Based on researches, was introduced a set of characteristics of the scheduling and the adaptive manufacturing domain was characterized using them.
4. The priorities mechanism based on resolution and time parameters of jobs were introduced to achieve one of the tasks of the research work – provide users possibilities to choose a preference between quality and printing time.
    a. In case of a resolution priority calculation, were presented required parameters for it and developed the method for calculating based on linear functions.
    b. For the time priority calculation were developed 5 calculation methods that based on different types of mathematical functions and have their own characteristics. Every 5 methods have the same interface and could be used interchangeably.
5. Different scheduling algorithms were introduced and described in this research. They were described in order of their complexity – a calculation complexity and count of different operations required for scheduling a job. Additionally, algorithms that were presented

early were used as basement for following described algorithms – they reused inner operations and introduced models:

a. Unprioritized least finish time stacking (ULFS) is the first algorithm that was described in this research work. It is not succeeding all constraints and goals of research work due to making decision of schedule based only on time priority. ULFS was introduced as a basic planning scheduling algorithm that was used for comparing with next presented algorithms.
b. PS algorithm is the most straightforward of algorithm that could be used. It is based on the ULFS, but with changing into introduction schedules options and making decision policy mechanism. As a result, it is easiest to implement and not required so much calculation complexity to schedule a job.
c. BQPJ is based on the same priority calculation mechanism as PS, but it's not a stacking type of algorithm. Instead, it is based on rescheduling previously scheduled jobs using the backfilling approach.
d. FPE is the most complex algorithm from a calculation and implementation perspective in comparison with previously introduced algorithms. It is based on principles described in BQPJ and PS, but to calculate schedules it enumerates all possible combinations that are not always could be done. The reason why it can't be done is the enumeration of all places is an NP problem that could not be always solved.

6. The simulation environment tool was introduced in the scope of this research. 3 workload simulation data sets of jobs and one data set of printers were introduced to perform simulation. Each introduced algorithm was simulated and summary metrics were calculated to compare them.

Conclusions

1. The planning scheduling system type was chosen as a basement for introduced scheduling algorithms. In comparison with the queue type, based on the introduced characteristics, assumptions and constraints, the planning type is more suitable for achieving research's goal.

2. BQPJ algorithm is recommended as the algorithm for scheduling workload based on performed simulations. In the first run of simulation, only BQPJ (FPE was canceled due to timeout) was able to schedule all incoming jobs from the data set. ULFS and PS in the middle of the simulation return responses with the message that they are not able to schedule one of the jobs. And this is not unexpected, because the data set for the first simulation was especially prepared to check how algorithms deal with strict deadlines commitments and a high load workload. In second and third simulations BQPJ also showed that it was able to introduce schedules with better resolution and fit into deadlines at the same time. In the case of PS, schedules have better resolution than ULFS algorithm, but utilization is worse and PS is not able to handle the first simulation's workload.
3. All introduced types of time priority calculation methods could be used in scheduling algorithms. Choosing the exact type of calculation could be done based on the characteristics of the systems – the type of calculation affects the result time priority for jobs schedules.

REFERENCES

[BSh19] P. Basiliere, M. Shanler, 2019. (PDF) Hype Cycle for 3D Printing – Gartner. [online] Available at: <https://www.gartner.com/en/documents/3947508/hype-cycle-for-3d-printing-2019> [Accessed 21 December 2020].

[FJe97] D. G. Feitelson, M. A. Jette, 1997. (PDF) Improved Utilization and Responsiveness with Gang Scheduling. [online] Available at: < Improved Utilization and Responsiveness with Gang Scheduling.> [Accessed 02 September 2021].

[FRS03] D.G. Feitelson, L. Rudolph, K.C. Sevcik, U. Schwiegelshohn, P. Wong, 2003. (PDF) Theory and Practice in Parallel Job Scheduling. [online] Available at: <https://www.cs.huji.ac.il/~feit/parsched/jsspp97/p-97-1.pdf> [Accessed 02 September 2021].

[FRu96] D.G. Feitelson, L. Rudolph, 1996. (PDF) Toward convergence in job schedulers for parallelsupercomputers. [online] Available at: <https://www.cs.huji.ac.il/~feit/parsched/jsspp96/p-96-1.pdf> [Accessed 02 September 2021].

[FRu98] D. G. Feitelson, L. Rudolph, 1998. (PDF) Metrics and Benchmarking for Parallel Job Scheduling. [online] Available at: <https://www.cs.huji.ac.il/~feit/parsched/jsspp98/p-98-1.pdf> [Accessed 02 September 2021].

[MFe01] A. Mu'alem, D. G. Feitelson, 2001. (PDF) Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. [online] Available at: <https://www.semanticscholar.org/paper/Utilization%2C-Predictability%2C-Workloads%2C-and-User-in-Mu'alem-Feitelson/b94d6bb4506dbb02244467f989b8aa1f06389988> [Accessed 02 September 2021].

[Min07] S. Mingsheng, 2007. (PDF) Optimal algorithm for scheduling large divisible workload on heterogeneous system. [online] Available at: <https://www.sciencedirect.com/science/article/pii/S0307904X07001308>

[Accessed 10 January 2021].

[MZT16] J.Mai, L. Zhang, F. Tao, L. Ren, 2016. (PDF) Customized production based on distributed 3D printing services in cloud manufacturing. [online] Available at: <https://www.researchgate.net/publication/284001571_Customized_production_based_on_distributed_3D_printing_services_in_cloud_manufacturing> [Accessed 02 September 2021].

[SFT99] W. Smith, I. Foster, V. Taylor, 1999. (PDF) Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance. [online] Available at: <https://www.globus.org/sites/default/files/p.pdf> [Accessed 02 September 2021].

[SKK03] A.Streit, O.Kao, A. Keller, 2003. (PDF) Scheduling in HPC Resource Management Systems: Queuing vs. Planning. [online] Available at: <https://www.academia.edu/18703311/Scheduling_in_HPC_Resource_Management_Systems_Queuing_vs_Planning> [Accessed 02 September 2021].

[Str02] A. Streit. 2016. (PDF) A Self-Tuning Job Scheduler Family with Dynamic Policy Switching. [online] Available at: <https://d-nb.info/971579393/34> [Accessed 02 September 2021].

[TSC22] Simulation tool for 3D printing systems (source code). [online] Available at: < https://github.com/artsiomtserashkovich/taas.printingscheduling.simulation.consoletool> [Accessed 12 February 2022].

[Thi22] Thingeverse web site. [online] Available at: <https://www.thingiverse.com/> [Accessed 12 February 2022].

[WLM96] K. Windisch, V. Lo, R. Moore, D. Feitelson, B. Nitzberg, 1996. (PDF)A Comparison of Workload Traces from Two Production Parallel Machines. [online] Available at: <https://www.researchgate.net/publication/232658182_A_Comparison_of_Workload_Traces_from_Two_Production_Parallel_Machines > [Accessed 02 September 2021].

[Zip01] P. Zipkin, 2001. (PDF) The Limits of Mass Customization. [online] Available at: <https://www.researchgate.net/publication/243770787_The_Limits_of_Mass_Customization> [Accessed 02 September 2021].

Appendix 1. Printers simulation data set

N	***p.s***	***p.d.x***	***p.d.y***	***p.d.z***	***p.r***
1	25	100	100	125	0.2
2	25	125	125	125	0.3
3	30	150	150	150	0.3
4	30	200	200	200	0.4
5	30	250	250	200	0.4
6	35	250	250	250	0.4
7	35	300	300	300	0.4
8	35	300	300	300	0.5
9	40	325	325	325	0.6
10	45	300	300	350	0.6
11	40	350	350	400	0.7
12	40	400	400	400	0.8

Appendix 2. Jobs simulation data set

N	*j.r*	*j.d.x*	*j.d.y*	*j.d.z*	*j.c*
1	0.2	47	23	40	0.8
2	0.2	62	26	57	0.8
3	0.2	69	39	60	0.8
4	0.3	78	54	82	0.7
5	0.3	79	55	91	0.7
6	0.3	85	62	98	0.7
7	0.3	84	63	97	0.7
8	0.3	93	67	117	0.7
9	0.3	108	75	132	0.7
10	0.3	122	78	132	0.7
11	0.4	129	84	150	0.6
12	0.4	141	96	152	0.6
13	0.4	158	116	156	0.6
14	0.4	141	96	152	0.6
15	0.4	158	116	156	0.6
16	0.4	159	117	159	0.6
17	0.4	191	129	159	0.6
18	0.4	192	129	162	0.6
19	0.4	193	169	168	0.6
20	0.4	210	180	172	0.6
21	0.4	224	192	173	0.6
22	0.4	226	199	184	0.6
23	0.5	230	200	192	0.5
24	0.5	243	204	206	0.5
25	0.5	249	214	215	0.5
26	0.5	255	227	227	0.5
27	0.5	261	239	227	0.5
28	0.6	266	256	236	0.4
29	0.6	269	263	247	0.4
30	0.6	271	275	272	0.4
31	0.6	272	295	258	0.4
32	0.7	279	296	221	0.3
33	0.7	293	301	328	0.3
34	0.7	298	312	332	0.3
35	0.8	346	383	388	0.2

Appendix 3. Simulation 1 BQPJ (linear) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	6578	6578	10901	4323	0.2	0.2
2	3	17437	17437	19351	1914	0.2	0.4
3	8	26072	27356	28179	2107	0.2	0.7
4	2	27408	28722	41514	14106	0.3	0.3
5	1	9143	10902	50441	41298	0.3	0.2
6	1	47171	101775	153420	106249	0.3	0.2
7	1	21256	50442	101774	80518	0.3	0.2
8	2	1720	1720	28721	27001	0.3	0.3
9	2	9446	88038	127638	118192	0.3	0.3
10	2	2815	41515	88037	85222	0.3	0.3
11	2	25687	127639	187839	162152	0.4	0.3
12	3	45663	337806	380669	335006	0.4	0.4
13	3	40302	278240	337805	297503	0.4	0.4
14	4	34733	271127	307867	273134	0.4	0.4
15	4	29000	220070	271126	242126	0.4	0.4
16	3	34181	216617	278239	244058	0.4	0.4
17	3	20840	20840	102456	81616	0.4	0.4
18	4	647	647	72297	71650	0.4	0.4
19	3	24919	102457	216616	191697	0.4	0.4
20	5	37725	373321	489420	451695	0.4	0.4
21	5	5816	5816	138679	132863	0.4	0.4
22	4	11610	72298	220069	208459	0.4	0.4
23	4	36559	307868	465582	429023	0.5	0.4
24	6	7846	7846	124552	116706	0.5	0.5
25	6	40328	286382	417313	376985	0.5	0.5
26	5	23050	138680	373320	350270	0.5	0.4
27	6	30363	124553	286381	256018	0.5	0.5
28	7	14424	14424	113625	99201	0.6	0.6
29	7	19859	113626	221493	201634	0.6	0.6
30	7	47733	221494	346622	298889	0.6	0.6
31	7	48276	346623	474412	426136	0.6	0.6
32	8	27356	28180	121297	93941	0.7	0.7
33	8	38074	278788	426376	388302	0.7	0.7
34	8	29338	121298	278787	249449	0.7	0.7
35	9	5186	5186	206033	200847	0.8	0.8
36	9	14228	206034	446152	431924	0.8	0.8

Appendix 4. Simulation 1 BQPJ (sin) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	3	6578	6578	7478	900	0.2	0.4
2	3	17437	17437	19351	1914	0.2	0.4
3	7	26072	26072	27068	996	0.2	0.6
4	8	27408	29000	30762	3354	0.3	0.7
5	2	9143	49338	63982	54839	0.3	0.3
6	1	47171	125956	177601	130430	0.3	0.2
7	1	21256	74623	125955	104699	0.3	0.2
8	1	1720	1720	74622	72902	0.3	0.2
9	2	9446	63983	103583	94137	0.3	0.3
10	2	2815	2815	49337	46522	0.3	0.3
11	2	25687	103584	163784	138097	0.4	0.3
12	3	45663	378613	421476	375813	0.4	0.4
13	3	40302	319047	378612	338310	0.4	0.4
14	3	34733	276183	319046	284313	0.4	0.4
15	3	29000	216617	276182	247182	0.4	0.4
16	4	34181	254653	307472	273291	0.4	0.4
17	3	20840	20840	102456	81616	0.4	0.4
18	4	647	647	72297	71650	0.4	0.4
19	3	24919	102457	216616	191697	0.4	0.4
20	4	37725	465188	581287	543562	0.4	0.4
21	5	5816	5816	138679	132863	0.4	0.4
22	5	11610	138680	286451	274841	0.4	0.4
23	4	36559	307473	465187	428628	0.5	0.4
24	4	7846	72298	254652	246806	0.5	0.4
25	4	40328	581288	785868	745540	0.5	0.4
26	6	23050	198089	348259	325209	0.5	0.5
27	5	30363	598497	851354	820991	0.5	0.4
28	6	14424	14424	198088	183664	0.6	0.5
29	5	19859	286452	598496	578637	0.6	0.4
30	7	47733	140017	265145	217412	0.6	0.6
31	6	48276	348260	584853	536577	0.6	0.5
32	7	27356	27356	140016	112660	0.7	0.6
33	8	38074	188253	335841	297767	0.7	0.7
34	8	29338	30763	188252	158914	0.7	0.7
35	9	5186	5186	206033	200847	0.8	0.8
36	9	14228	206034	446152	431924	0.8	0.8

Appendix 5. Simulation 1 BQPJ (cos) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	6578	6578	10901	4323	0.2	0.2
2	3	17437	17437	19351	1914	0.2	0.4
3	8	26072	27356	28179	2107	0.2	0.7
4	2	27408	28722	41514	14106	0.3	0.3
5	1	9143	10902	50441	41298	0.3	0.2
6	1	47171	101775	153420	106249	0.3	0.2
7	1	21256	50442	101774	80518	0.3	0.2
8	2	1720	1720	28721	27001	0.3	0.3
9	2	9446	88038	127638	118192	0.3	0.3
10	2	2815	41515	88037	85222	0.3	0.3
11	2	25687	127639	187839	162152	0.4	0.3
12	3	45663	319047	361910	316247	0.4	0.4
13	3	40302	259481	319046	278744	0.4	0.4
14	3	34733	216617	259480	224747	0.4	0.4
15	5	29000	138680	189736	160736	0.4	0.4
16	4	34181	220070	272889	238708	0.4	0.4
17	3	20840	20840	102456	81616	0.4	0.4
18	4	647	647	72297	71650	0.4	0.4
19	3	24919	102457	216616	191697	0.4	0.4
20	4	37725	430605	546704	508979	0.4	0.4
21	5	5816	5816	138679	132863	0.4	0.4
22	4	11610	72298	220069	208459	0.4	0.4
23	4	36559	272890	430604	394045	0.5	0.4
24	6	7846	7846	124552	116706	0.5	0.5
25	6	40328	274724	405655	365327	0.5	0.5
26	6	23050	124553	274723	251673	0.5	0.5
27	5	30363	189737	442594	412231	0.5	0.4
28	7	14424	14424	113625	99201	0.6	0.6
29	7	19859	113626	221493	201634	0.6	0.6
30	7	47733	221494	346622	298889	0.6	0.6
31	7	48276	346623	474412	426136	0.6	0.6
32	8	27356	28180	121297	93941	0.7	0.7
33	8	38074	278788	426376	388302	0.7	0.7
34	8	29338	121298	278787	249449	0.7	0.7
35	9	5186	5186	206033	200847	0.8	0.8
36	9	14228	206034	446152	431924	0.8	0.8

Appendix 6. Simulation 2 ULFS result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	2862	2862	7185	4323	0.2	0.2
2	1	26924	26924	36112	9188	0.2	0.2
3	1	44614	81340	97485	52871	0.2	0.2
4	2	29605	32477	45269	15664	0.3	0.3
5	1	41800	41800	81339	39539	0.3	0.2
6	2	35130	45270	64398	29268	0.3	0.3
7	2	13464	13464	32476	19012	0.3	0.3
8	1	46131	97486	170388	124257	0.3	0.2
9	2	38748	64399	103999	65251	0.3	0.3
10	2	40914	104000	150522	109608	0.3	0.3
11	3	7042	7042	40904	33862	0.4	0.4
12	5	40708	318089	354829	314121	0.4	0.4
13	3	23644	40905	100470	76826	0.4	0.4
14	3	31965	100471	143334	111369	0.4	0.4
15	5	3160	3160	54216	51056	0.4	0.4
16	3	46027	339112	400734	354707	0.4	0.4
17	3	32445	143335	224951	192506	0.4	0.4
18	4	2377	2377	74027	71650	0.4	0.4
19	3	34992	224952	339111	304119	0.4	0.4
20	5	30898	201989	318088	287190	0.4	0.4
21	4	24920	74028	206891	181971	0.4	0.4
22	5	17804	54217	201988	184184	0.4	0.4
23	4	33531	206892	364606	331075	0.5	0.4
24	6	8475	8475	125181	116706	0.5	0.5
25	6	9730	125182	256113	246383	0.5	0.5
26	6	21606	256114	406284	384678	0.5	0.5
27	6	49618	406285	568113	518495	0.5	0.5
28	7	22242	241479	340680	318438	0.6	0.6
29	7	5821	5821	113688	107867	0.6	0.6
30	7	41045	340681	465809	424764	0.6	0.6
31	7	7218	113689	241478	234260	0.6	0.6
32	8	4417	4417	97534	93117	0.7	0.7
33	8	14555	255025	402613	388058	0.7	0.7
34	8	8395	97535	255024	246629	0.7	0.7
35	9	38223	257124	457971	419748	0.8	0.8
36	9	17005	17005	257123	240118	0.8	0.8

Appendix 7. Simulation 2 PS (linear) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	2862	2862	7185	4323	0.2	0.2
2	1	26924	64797	73985	47061	0.2	0.2
3	1	44614	165172	181317	136703	0.2	0.2
4	2	29605	29605	42397	12792	0.3	0.3
5	1	41800	125632	165171	123371	0.3	0.2
6	1	35130	73986	125631	90501	0.3	0.2
7	1	13464	13464	64796	51332	0.3	0.2
8	1	46131	181318	254220	208089	0.3	0.2
9	2	38748	42398	81998	43250	0.3	0.3
10	2	40914	81999	128521	87607	0.3	0.3
11	3	7042	7042	40904	33862	0.4	0.4
12	3	40708	339112	381975	341267	0.4	0.4
13	3	23644	40905	100470	76826	0.4	0.4
14	3	31965	100471	143334	111369	0.4	0.4
15	5	3160	3160	54216	51056	0.4	0.4
16	3	46027	381976	443598	397571	0.4	0.4
17	3	32445	143335	224951	192506	0.4	0.4
18	4	2377	2377	74027	71650	0.4	0.4
19	3	34992	224952	339111	304119	0.4	0.4
20	5	30898	384344	500443	469545	0.4	0.4
21	4	24920	278609	411472	386552	0.4	0.4
22	5	17804	236572	384343	366539	0.4	0.4
23	8	33531	97535	142596	109065	0.5	0.7
24	5	8475	54217	236571	228096	0.5	0.4
25	4	9730	74028	278608	268878	0.5	0.4
26	6	21606	243812	393982	372376	0.5	0.5
27	6	49618	393983	555811	506193	0.5	0.5
28	7	22242	113689	212890	190648	0.6	0.6
29	7	5821	5821	113688	107867	0.6	0.6
30	7	41045	212891	338019	296974	0.6	0.6
31	6	7218	7218	243811	236593	0.6	0.5
32	8	4417	4417	97534	93117	0.7	0.7
33	9	14555	128974	241971	227416	0.7	0.8
34	9	8395	8395	128973	120578	0.7	0.8
35	9	38223	482091	682938	644715	0.8	0.8
36	9	17005	241972	482090	465085	0.8	0.8

Appendix 8. Simulation 2 PS (sin) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	2862	2862	7185	4323	0.2	0.2
2	1	26924	64797	73985	47061	0.2	0.2
3	1	44614	199711	215856	171242	0.2	0.2
4	1	29605	73986	108524	78919	0.3	0.2
5	1	41800	160171	199710	157910	0.3	0.2
6	1	35130	108525	160170	125040	0.3	0.2
7	1	13464	13464	64796	51332	0.3	0.2
8	1	46131	215857	288759	242628	0.3	0.2
9	2	38748	38748	78348	39600	0.3	0.3
10	2	40914	78349	124871	83957	0.3	0.3
11	3	7042	7042	40904	33862	0.4	0.4
12	3	40708	339112	381975	341267	0.4	0.4
13	3	23644	40905	100470	76826	0.4	0.4
14	3	31965	100471	143334	111369	0.4	0.4
15	5	3160	3160	54216	51056	0.4	0.4
16	3	46027	381976	443598	397571	0.4	0.4
17	3	32445	143335	224951	192506	0.4	0.4
18	4	2377	2377	74027	71650	0.4	0.4
19	3	34992	224952	339111	304119	0.4	0.4
20	5	30898	384344	500443	469545	0.4	0.4
21	4	24920	278609	411472	386552	0.4	0.4
22	5	17804	236572	384343	366539	0.4	0.4
23	6	33531	393983	494920	461389	0.5	0.5
24	5	8475	54217	236571	228096	0.5	0.4
25	4	9730	74028	278608	268878	0.5	0.4
26	6	21606	243812	393982	372376	0.5	0.5
27	8	49618	97535	169780	120162	0.5	0.7
28	7	22242	113689	212890	190648	0.6	0.6
29	7	5821	5821	113688	107867	0.6	0.6
30	7	41045	212891	338019	296974	0.6	0.6
31	6	7218	7218	243811	236593	0.6	0.5
32	8	4417	4417	97534	93117	0.7	0.7
33	9	14555	128974	241971	227416	0.7	0.8
34	9	8395	8395	128973	120578	0.7	0.8
35	9	38223	482091	682938	644715	0.8	0.8
36	9	17005	241972	482090	465085	0.8	0.8

Appendix 9. Simulation 2 PS (cos) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	2862	2862	7185	4323	0.2	0.2
2	1	26924	26924	36112	9188	0.2	0.2
3	1	44614	110192	126337	81723	0.2	0.2
4	1	29605	36113	70651	41046	0.3	0.2
5	1	41800	70652	110191	68391	0.3	0.2
6	2	35130	35130	54258	19128	0.3	0.3
7	2	13464	13464	32476	19012	0.3	0.3
8	1	46131	126338	199240	153109	0.3	0.2
9	2	38748	54259	93859	55111	0.3	0.3
10	2	40914	93860	140382	99468	0.3	0.3
11	3	7042	7042	40904	33862	0.4	0.4
12	5	40708	318089	354829	314121	0.4	0.4
13	3	23644	40905	100470	76826	0.4	0.4
14	3	31965	100471	143334	111369	0.4	0.4
15	5	3160	3160	54216	51056	0.4	0.4
16	3	46027	339112	400734	354707	0.4	0.4
17	3	32445	143335	224951	192506	0.4	0.4
18	4	2377	2377	74027	71650	0.4	0.4
19	3	34992	224952	339111	304119	0.4	0.4
20	5	30898	201989	318088	287190	0.4	0.4
21	4	24920	74028	206891	181971	0.4	0.4
22	5	17804	54217	201988	184184	0.4	0.4
23	4	33531	206892	364606	331075	0.5	0.4
24	6	8475	8475	125181	116706	0.5	0.5
25	8	9730	97535	155986	146256	0.5	0.7
26	6	21606	125182	275352	253746	0.5	0.5
27	6	49618	275353	437181	387563	0.5	0.5
28	7	22242	113689	212890	190648	0.6	0.6
29	7	5821	5821	113688	107867	0.6	0.6
30	7	41045	212891	338019	296974	0.6	0.6
31	9	7218	7218	88084	80866	0.6	0.8
32	8	4417	4417	97534	93117	0.7	0.7
33	8	14555	155987	303575	289020	0.7	0.7
34	9	8395	88085	208663	200268	0.7	0.8
35	9	38223	448783	649630	611407	0.8	0.8
36	9	17005	208664	448782	431777	0.8	0.8

Appendix 10. Simulation 2 BQPJ (linear) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	2862	2862	7185	4323	0.2	0.2
2	1	26924	64797	73985	47061	0.2	0.2
3	1	44614	199711	215856	171242	0.2	0.2
4	1	29605	73986	108524	78919	0.3	0.2
5	1	41800	160171	199710	157910	0.3	0.2
6	1	35130	108525	160170	125040	0.3	0.2
7	1	13464	13464	64796	51332	0.3	0.2
8	1	46131	215857	288759	242628	0.3	0.2
9	2	38748	67243	106843	68095	0.3	0.3
10	2	40914	106844	153366	112452	0.3	0.3
11	2	7042	7042	67242	60200	0.4	0.3
12	3	40708	321851	364714	324006	0.4	0.4
13	3	23644	23644	83209	59565	0.4	0.4
14	3	31965	83210	126073	94108	0.4	0.4
15	5	3160	3160	54216	51056	0.4	0.4
16	3	46027	364715	426337	380310	0.4	0.4
17	3	32445	126074	207690	175245	0.4	0.4
18	4	2377	2377	74027	71650	0.4	0.4
19	3	34992	207691	321850	286858	0.4	0.4
20	4	30898	741600	857699	826801	0.4	0.4
21	4	24920	608736	741599	716679	0.4	0.4
22	4	17804	460964	608735	590931	0.4	0.4
23	6	33531	450999	551936	518405	0.5	0.5
24	4	8475	74028	256382	247907	0.5	0.4
25	4	9730	256383	460963	451233	0.5	0.4
26	5	21606	619120	853760	832154	0.5	0.4
27	5	49618	54217	307074	257456	0.5	0.4
28	7	22242	23644	122845	100603	0.6	0.6
29	5	5821	307075	619119	613298	0.6	0.4
30	7	41045	122846	247974	206929	0.6	0.6
31	6	7218	214405	450998	443780	0.6	0.5
32	6	4417	5821	214404	209987	0.7	0.5
33	8	14555	165885	313473	298918	0.7	0.7
34	8	8395	8395	165884	157489	0.7	0.7
35	9	38223	257124	457971	419748	0.8	0.8
36	9	17005	17005	257123	240118	0.8	0.8

Appendix 11. Simulation 2 BQPJ (sin) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	2862	2862	7185	4323	0.2	0.2
2	1	26924	64797	73985	47061	0.2	0.2
3	1	44614	199711	215856	171242	0.2	0.2
4	1	29605	73986	108524	78919	0.3	0.2
5	1	41800	160171	199710	157910	0.3	0.2
6	1	35130	108525	160170	125040	0.3	0.2
7	1	13464	13464	64796	51332	0.3	0.2
8	1	46131	215857	288759	242628	0.3	0.2
9	2	38748	67243	106843	68095	0.3	0.3
10	2	40914	106844	153366	112452	0.3	0.3
11	2	7042	7042	67242	60200	0.4	0.3
12	3	40708	321851	364714	324006	0.4	0.4
13	3	23644	23644	83209	59565	0.4	0.4
14	3	31965	83210	126073	94108	0.4	0.4
15	5	3160	3160	54216	51056	0.4	0.4
16	3	46027	364715	426337	380310	0.4	0.4
17	3	32445	126074	207690	175245	0.4	0.4
18	4	2377	2377	74027	71650	0.4	0.4
19	3	34992	207691	321850	286858	0.4	0.4
20	4	30898	741600	857699	826801	0.4	0.4
21	4	24920	608736	741599	716679	0.4	0.4
22	4	17804	460964	608735	590931	0.4	0.4
23	4	33531	857700	1015414	981883	0.5	0.4
24	4	8475	74028	256382	247907	0.5	0.4
25	4	9730	256383	460963	451233	0.5	0.4
26	5	21606	1314710	1549350	1527744	0.5	0.4
27	5	49618	692174	945031	895413	0.5	0.4
28	5	22242	1549351	1836326	1814084	0.6	0.4
29	5	5821	380129	692173	686352	0.6	0.4
30	5	41045	1836327	2198305	2157260	0.6	0.4
31	5	7218	945032	1314709	1307491	0.6	0.4
32	5	4417	54217	380128	375711	0.7	0.4
33	8	14555	165885	313473	298918	0.7	0.7
34	8	8395	8395	165884	157489	0.7	0.7
35	9	38223	257124	457971	419748	0.8	0.8
36	9	17005	17005	257123	240118	0.8	0.8

Appendix 12. Simulation 2 BQPJ (cos) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	2862	2862	7185	4323	0.2	0.2
2	1	26924	64797	73985	47061	0.2	0.2
3	1	44614	199711	215856	171242	0.2	0.2
4	1	29605	73986	108524	78919	0.3	0.2
5	1	41800	160171	199710	157910	0.3	0.2
6	1	35130	108525	160170	125040	0.3	0.2
7	1	13464	13464	64796	51332	0.3	0.2
8	1	46131	215857	288759	242628	0.3	0.2
9	2	38748	67243	106843	68095	0.3	0.3
10	2	40914	106844	153366	112452	0.3	0.3
11	2	7042	7042	67242	60200	0.4	0.3
12	3	40708	321851	364714	324006	0.4	0.4
13	3	23644	23644	83209	59565	0.4	0.4
14	3	31965	83210	126073	94108	0.4	0.4
15	5	3160	3160	54216	51056	0.4	0.4
16	3	46027	364715	426337	380310	0.4	0.4
17	3	32445	126074	207690	175245	0.4	0.4
18	4	2377	2377	74027	71650	0.4	0.4
19	3	34992	207691	321850	286858	0.4	0.4
20	5	30898	514034	630133	599235	0.4	0.4
21	4	24920	460964	593827	568907	0.4	0.4
22	5	17804	366262	514033	496229	0.4	0.4
23	6	33531	363172	464109	430578	0.5	0.5
24	4	8475	74028	256382	247907	0.5	0.4
25	4	9730	256383	460963	451233	0.5	0.4
26	6	21606	213001	363171	341565	0.5	0.5
27	5	49618	630134	882991	833373	0.5	0.4
28	7	22242	135008	234209	211967	0.6	0.6
29	5	5821	54217	366261	360440	0.6	0.4
30	7	41045	234210	359338	318293	0.6	0.6
31	7	7218	7218	135007	127789	0.6	0.6
32	6	4417	4417	213000	208583	0.7	0.5
33	8	14555	165885	313473	298918	0.7	0.7
34	8	8395	8395	165884	157489	0.7	0.7
35	9	38223	257124	457971	419748	0.8	0.8
36	9	17005	17005	257123	240118	0.8	0.8

Appendix 13. Simulation 3 ULFS result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	14609	17646	21969	7360	0.2	0.2
2	1	39173	90360	99548	60375	0.2	0.2
3	1	1500	1500	17645	16145	0.2	0.2
4	2	39812	112916	125708	85896	0.3	0.3
5	2	32185	32619	47263	15078	0.3	0.3
6	2	39762	93787	112915	73153	0.3	0.3
7	1	39027	39027	90359	51332	0.3	0.2
8	2	5617	5617	32618	27001	0.3	0.3
9	2	48118	125709	165309	117191	0.3	0.3
10	2	37663	47264	93786	56123	0.3	0.3
11	4	7160	7160	36184	29024	0.4	0.4
12	3	41622	312776	355639	314017	0.4	0.4
13	5	7464	7464	58520	51056	0.4	0.4
14	3	33082	186320	229183	196101	0.4	0.4
15	3	12594	12594	72159	59565	0.4	0.4
16	4	15087	36185	89004	73917	0.4	0.4
17	4	46449	337969	407926	361477	0.4	0.4
18	3	36689	229184	312775	276086	0.4	0.4
19	3	23733	72160	186319	162586	0.4	0.4
20	4	29773	221869	337968	308195	0.4	0.4
21	4	19949	89005	221868	201919	0.4	0.4
22	5	37426	293162	440933	403507	0.4	0.4
23	6	7323	7323	108260	100937	0.5	0.5
24	6	39365	401022	517728	478363	0.5	0.5
25	6	21650	270090	401021	379371	0.5	0.5
26	5	13324	58521	293161	279837	0.5	0.4
27	6	11081	108261	270089	259008	0.5	0.5
28	7	20906	137980	237181	216275	0.6	0.6
29	7	46793	474972	582839	536046	0.6	0.6
30	7	36837	349843	474971	438134	0.6	0.6
31	7	10190	10190	137979	127789	0.6	0.6
32	7	31653	237182	349842	318189	0.7	0.6
33	8	20521	166877	314465	293944	0.7	0.7
34	8	9387	9387	166876	157489	0.7	0.7
35	9	24685	249232	450079	425394	0.8	0.8
36	9	9113	9113	249231	240118	0.8	0.8

Appendix 14. Simulation 3 PS (linear) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	14609	17646	21969	7360	0.2	0.2
2	1	39173	123058	132246	93073	0.2	0.2
3	1	1500	1500	17645	16145	0.2	0.2
4	2	39812	84186	96978	57166	0.3	0.3
5	1	32185	32185	71724	39539	0.3	0.2
6	1	39762	132247	183892	144130	0.3	0.2
7	1	39027	71725	123057	84030	0.3	0.2
8	2	5617	5617	32618	27001	0.3	0.3
9	2	48118	96979	136579	88461	0.3	0.3
10	2	37663	37663	84185	46522	0.3	0.3
11	4	7160	7160	36184	29024	0.4	0.4
12	3	41622	312776	355639	314017	0.4	0.4
13	5	7464	7464	58520	51056	0.4	0.4
14	3	33082	186320	229183	196101	0.4	0.4
15	3	12594	12594	72159	59565	0.4	0.4
16	4	15087	36185	89004	73917	0.4	0.4
17	3	46449	355640	437256	390807	0.4	0.4
18	3	36689	229184	312775	276086	0.4	0.4
19	3	23733	72160	186319	162586	0.4	0.4
20	4	29773	221869	337968	308195	0.4	0.4
21	4	19949	89005	221868	201919	0.4	0.4
22	5	37426	293162	440933	403507	0.4	0.4
23	6	7323	7323	108260	100937	0.5	0.5
24	4	39365	337969	520323	480958	0.5	0.4
25	6	21650	270090	401021	379371	0.5	0.5
26	5	13324	58521	293161	279837	0.5	0.4
27	6	11081	108261	270089	259008	0.5	0.5
28	7	20906	137980	237181	216275	0.6	0.6
29	8	46793	314466	403621	356828	0.6	0.7
30	7	36837	349843	474971	438134	0.6	0.6
31	7	10190	10190	137979	127789	0.6	0.6
32	7	31653	237182	349842	318189	0.7	0.6
33	8	20521	166877	314465	293944	0.7	0.7
34	8	9387	9387	166876	157489	0.7	0.7
35	9	24685	249232	450079	425394	0.8	0.8
36	9	9113	9113	249231	240118	0.8	0.8

Appendix 15. Simulation 3 PS (sin) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	14609	17646	21969	7360	0.2	0.2
2	1	39173	123058	132246	93073	0.2	0.2
3	1	1500	1500	17645	16145	0.2	0.2
4	2	39812	84186	96978	57166	0.3	0.3
5	1	32185	32185	71724	39539	0.3	0.2
6	1	39762	132247	183892	144130	0.3	0.2
7	1	39027	71725	123057	84030	0.3	0.2
8	2	5617	5617	32618	27001	0.3	0.3
9	2	48118	96979	136579	88461	0.3	0.3
10	2	37663	37663	84185	46522	0.3	0.3
11	4	7160	7160	36184	29024	0.4	0.4
12	3	41622	312776	355639	314017	0.4	0.4
13	5	7464	7464	58520	51056	0.4	0.4
14	3	33082	186320	229183	196101	0.4	0.4
15	3	12594	12594	72159	59565	0.4	0.4
16	4	15087	36185	89004	73917	0.4	0.4
17	3	46449	355640	437256	390807	0.4	0.4
18	3	36689	229184	312775	276086	0.4	0.4
19	3	23733	72160	186319	162586	0.4	0.4
20	4	29773	221869	337968	308195	0.4	0.4
21	4	19949	89005	221868	201919	0.4	0.4
22	5	37426	293162	440933	403507	0.4	0.4
23	6	7323	7323	108260	100937	0.5	0.5
24	4	39365	337969	520323	480958	0.5	0.4
25	6	21650	270090	401021	379371	0.5	0.5
26	5	13324	58521	293161	279837	0.5	0.4
27	6	11081	108261	270089	259008	0.5	0.5
28	7	20906	137980	237181	216275	0.6	0.6
29	7	46793	474972	582839	536046	0.6	0.6
30	7	36837	349843	474971	438134	0.6	0.6
31	7	10190	10190	137979	127789	0.6	0.6
32	7	31653	237182	349842	318189	0.7	0.6
33	8	20521	166877	314465	293944	0.7	0.7
34	8	9387	9387	166876	157489	0.7	0.7
35	9	24685	249232	450079	425394	0.8	0.8
36	9	9113	9113	249231	240118	0.8	0.8

Appendix 16. Simulation 3 PS (cos) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	1	14609	17646	21969	7360	0.2	0.2
2	1	39173	123058	132246	93073	0.2	0.2
3	1	1500	1500	17645	16145	0.2	0.2
4	2	39812	103315	116107	76295	0.3	0.3
5	1	32185	32185	71724	39539	0.3	0.2
6	2	39762	84186	103314	63552	0.3	0.3
7	1	39027	71725	123057	84030	0.3	0.2
8	2	5617	5617	32618	27001	0.3	0.3
9	2	48118	116108	155708	107590	0.3	0.3
10	2	37663	37663	84185	46522	0.3	0.3
11	4	7160	7160	36184	29024	0.4	0.4
12	5	41622	307485	344225	302603	0.4	0.4
13	5	7464	7464	58520	51056	0.4	0.4
14	3	33082	186320	229183	196101	0.4	0.4
15	3	12594	12594	72159	59565	0.4	0.4
16	4	15087	36185	89004	73917	0.4	0.4
17	3	46449	312776	394392	347943	0.4	0.4
18	3	36689	229184	312775	276086	0.4	0.4
19	3	23733	72160	186319	162586	0.4	0.4
20	5	29773	191385	307484	277711	0.4	0.4
21	5	19949	58521	191384	171435	0.4	0.4
22	4	37426	293586	441357	403931	0.4	0.4
23	6	7323	7323	108260	100937	0.5	0.5
24	6	39365	258432	375138	335773	0.5	0.5
25	4	21650	89005	293585	271935	0.5	0.4
26	6	13324	108261	258431	245107	0.5	0.5
27	7	11081	137980	225387	214306	0.5	0.6
28	7	20906	225388	324589	303683	0.6	0.6
29	8	46793	407584	496739	449946	0.6	0.7
30	7	36837	324590	449718	412881	0.6	0.6
31	7	10190	10190	137979	127789	0.6	0.6
32	8	31653	314466	407583	375930	0.7	0.7
33	8	20521	166877	314465	293944	0.7	0.7
34	8	9387	9387	166876	157489	0.7	0.7
35	9	24685	249232	450079	425394	0.8	0.8
36	9	9113	9113	249231	240118	0.8	0.8

Appendix 17. Simulation 3 BQPJ (linear) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	7	14609	15087	15353	744	0.2	0.6
2	2	39173	113884	117287	78114	0.2	0.3
3	1	1500	1500	17645	16145	0.2	0.2
4	2	39812	117288	130080	90268	0.3	0.3
5	1	32185	90549	130088	97903	0.3	0.2
6	1	39762	181422	233067	193305	0.3	0.2
7	1	39027	130089	181421	142394	0.3	0.2
8	1	5617	17646	90548	84931	0.3	0.2
9	2	48118	130081	169681	121563	0.3	0.3
10	2	37663	67361	113883	76220	0.3	0.3
11	2	7160	7160	67360	60200	0.4	0.3
12	3	41622	374399	417262	375640	0.4	0.4
13	5	7464	7464	58520	51056	0.4	0.4
14	3	33082	247943	290806	257724	0.4	0.4
15	3	12594	12594	72159	59565	0.4	0.4
16	3	15087	72160	133782	118695	0.4	0.4
17	3	46449	417263	498879	452430	0.4	0.4
18	3	36689	290807	374398	337709	0.4	0.4
19	3	23733	133783	247942	224209	0.4	0.4
20	4	29773	502483	618582	588809	0.4	0.4
21	4	19949	165038	297901	277952	0.4	0.4
22	4	37426	618583	766354	728928	0.4	0.4
23	4	7323	7323	165037	157714	0.5	0.4
24	6	39365	358818	475524	436159	0.5	0.5
25	4	21650	297902	502482	480832	0.5	0.4
26	5	13324	428199	662839	649515	0.5	0.4
27	6	11081	13324	175152	164071	0.5	0.5
28	6	20906	175153	358817	337911	0.6	0.5
29	7	46793	144314	252181	205388	0.6	0.6
30	5	36837	662840	1024818	987981	0.6	0.4
31	5	10190	58521	428198	418008	0.6	0.4
32	7	31653	31653	144313	112660	0.7	0.6
33	8	20521	166877	314465	293944	0.7	0.7
34	8	9387	9387	166876	157489	0.7	0.7
35	9	24685	249232	450079	425394	0.8	0.8
36	9	9113	9113	249231	240118	0.8	0.8

Appendix 18. Simulation 3 BQPJ (sin) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	6	14609	15087	15581	972	0.2	0.5
2	7	39173	39365	39932	759	0.2	0.6
3	1	1500	1500	17645	16145	0.2	0.2
4	2	39812	113884	126676	86864	0.3	0.3
5	1	32185	90549	130088	97903	0.3	0.2
6	1	39762	181422	233067	193305	0.3	0.2
7	1	39027	130089	181421	142394	0.3	0.2
8	1	5617	17646	90548	84931	0.3	0.2
9	2	48118	126677	166277	118159	0.3	0.3
10	2	37663	67361	113883	76220	0.3	0.3
11	2	7160	7160	67360	60200	0.4	0.3
12	3	41622	374399	417262	375640	0.4	0.4
13	5	7464	7464	58520	51056	0.4	0.4
14	3	33082	247943	290806	257724	0.4	0.4
15	3	12594	12594	72159	59565	0.4	0.4
16	3	15087	72160	133782	118695	0.4	0.4
17	3	46449	417263	498879	452430	0.4	0.4
18	3	36689	290807	374398	337709	0.4	0.4
19	3	23733	133783	247942	224209	0.4	0.4
20	4	29773	502483	618582	588809	0.4	0.4
21	4	19949	165038	297901	277952	0.4	0.4
22	4	37426	618583	766354	728928	0.4	0.4
23	4	7323	7323	165037	157714	0.5	0.4
24	4	39365	766355	948709	909344	0.5	0.4
25	4	21650	297902	502482	480832	0.5	0.4
26	5	13324	681057	915697	902373	0.5	0.4
27	5	11081	428199	681056	669975	0.5	0.4
28	5	20906	915698	1202673	1181767	0.6	0.4
29	5	46793	1564653	1876697	1829904	0.6	0.4
30	5	36837	1202674	1564652	1527815	0.6	0.4
31	5	10190	58521	428198	418008	0.6	0.4
32	6	31653	36837	245420	213767	0.7	0.5
33	8	20521	166877	314465	293944	0.7	0.7
34	8	9387	9387	166876	157489	0.7	0.7
35	9	24685	249232	450079	425394	0.8	0.8
36	9	9113	9113	249231	240118	0.8	0.8

Appendix 19. Simulation 3 BQPJ (cos) result

j	*p*	*j.t₀*	*execution start*	*execution finish*	*time in system*	*j.r*	*p.r*
1	7	14609	15087	15353	744	0.2	0.6
2	2	39173	113884	117287	78114	0.2	0.3
3	1	1500	1500	17645	16145	0.2	0.2
4	1	39812	181422	215960	176148	0.3	0.2
5	1	32185	90549	130088	97903	0.3	0.2
6	2	39762	117288	136416	96654	0.3	0.3
7	1	39027	130089	181421	142394	0.3	0.2
8	1	5617	17646	90548	84931	0.3	0.2
9	2	48118	136417	176017	127899	0.3	0.3
10	2	37663	67361	113883	76220	0.3	0.3
11	2	7160	7160	67360	60200	0.4	0.3
12	3	41622	374399	417262	375640	0.4	0.4
13	5	7464	7464	58520	51056	0.4	0.4
14	3	33082	247943	290806	257724	0.4	0.4
15	3	12594	12594	72159	59565	0.4	0.4
16	3	15087	72160	133782	118695	0.4	0.4
17	3	46449	417263	498879	452430	0.4	0.4
18	3	36689	290807	374398	337709	0.4	0.4
19	3	23733	133783	247942	224209	0.4	0.4
20	4	29773	502483	618582	588809	0.4	0.4
21	4	19949	165038	297901	277952	0.4	0.4
22	5	37426	546020	693791	656365	0.4	0.4
23	4	7323	7323	165037	157714	0.5	0.4
24	4	39365	618583	800937	761572	0.5	0.4
25	4	21650	297902	502482	480832	0.5	0.4
26	5	13324	311379	546019	532695	0.5	0.4
27	5	11081	58521	311378	300297	0.5	0.4
28	7	20906	20906	120107	99201	0.6	0.6
29	7	46793	232769	340636	293843	0.6	0.6
30	6	36837	247675	479341	442504	0.6	0.5
31	6	10190	11081	247674	237484	0.6	0.5
32	7	31653	120108	232768	201115	0.7	0.6
33	8	20521	166877	314465	293944	0.7	0.7
34	8	9387	9387	166876	157489	0.7	0.7
35	9	24685	249232	450079	425394	0.8	0.8
36	9	9113	9113	249231	240118	0.8	0.8