

ASH - A SCHEDULER FOR HOAs

A Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

by

Qian Li

September 2005

© 2005

Qian Li

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

Dr. Chris Pollett

Dr. Sami Khuri

Dr. David Taylor

ABSTRACT

ASH - A SCHEDULER FOR HOAs

By Qian Li

The resource constrained scheduling problem is a basic problem in computer science research, supply chain management research, and operations research. My project focuses on developing job scheduler, Super-ASH, capable of aiding homeowner associations in achieving an efficient schedule for repair jobs. The goal is not only to have customers' requests fulfilled in a timely manner, but also to give immediate feedback of a timeframe in which activities jobs will be completed. Moreover, job scheduler enables homeowner associations to set up their own configurations, allowing them to set the job type table, define the scheduling queue model, and change the interval and amounts of the incoming budget based on their own real conditions. The two basic scheduling objectives we seek to optimize are to minimize the (actual) average flow time, and to minimize the (actual) average stretch. To achieve these goals, we considered the advantages and drawbacks of a multi-level-queue scheduling algorithm, the simplicity of First In First Out (FIFO) and Shortest Processing Time First (SPTF), and the semi-clairvoyant R algorithm. From considering these, we developed our Super-ASH scheduler. In this report, the detailed design and implementation of a Super-ASH scheduler is described. Also the different test cases and results will be investigated in various ways to yield good scheduling performance.

TABLE OF CONTENTS

1. INTRODUCTION	9
1. 1 Scheduling Problems	11
1. 2 Define HOAs Scheduling Problem.....	13
2. BACKGROUND and RELATED WORK	15
2. 1 Performance Measures on Scheduling Algorithms.....	15
2. 2 Original Scheduling Algorithms.....	16
2. 2. 1 Greedy UnitTask Scheduling Algorithm	16
2. 2. 2 First In First Out (FIFO)	19
2. 2. 3 Shortest Processing Time First (SPTF).....	20
2. 2. 4 Semi-clairvoyant R algorithm (Sc-R)	20
2. 2. 5 Comparisons on Sc-R, FIFO and SPTF	23
2. 3 Inspirations from Our Initial Research	24
3. DESIGN and IMPLEMENTATION	27
3. 1 Supplemental Constraints	27
3. 2 Job Type Table and Contractors	29
3. 3 Configuration Objectives	30
3. 4 Configuration Parameters	31
3. 4. 1 Budget Intervals and Amounts.....	31
3. 4. 2 Queue Models	33
3. 4. 3 Statistical Results and Other Display Options	36
3. 5 Super-ASH Algorithm	42
3. 5. 1 Super-ASH Overview	42
3. 5. 2 Class Organization	42
4. EXPERIMENTS and ANALYSIS	45
4. 1 Discoveries in Experiments	45
4. 1. 1 Number of Jobs	46
4. 1. 2 Cost of Jobs.....	46
4. 1. 3 Intervals and Budget Amounts.....	50
4. 1. 4 Different Scheduling Queue Models	58
4. 1. 5 Yearly Performance Comparison.....	58
4. 2 Test Results Analysis.....	61
5. CONCLUSION and FUTURE WORK	63
6. ASH User's Guide	65
6.1 Creating Job Type Table.....	65

6.2	Creating Job List.....	67
6.3	Choosing Scheduling Method.....	69
REFERENCES		72

LIST OF TABLES

Table 2-1. Schedule result for no job late for their deadlines	17
Table 2-2. Schedule results for some jobs late for deadlines and got penalties.....	18
Table 2-3. One example of the combinational scheduler scheduling results.....	23
Table 2-4. Enumeration of some results on different number of jobs	23
Table 4-1. Low cost jobs with time 5 scheduling statistical results.....	47
Table 4-2. High cost jobs with Time 5 scheduling statistical results.....	48
Table 4-3. Mix cost jobs with Time 5 scheduling statistical results	49
Table 4-4. Same amounts and different intervals for high cost jobs	51
Table 4-5. Same amounts and different intervals for low cost jobs.....	52
Table 4-6. Same intervals and different amounts for high cost jobs	54
Table 4-7. Same intervals and different amounts for low cost jobs.....	56
Table 4-8 Yearly performance test on high competition jobs.....	60
Table 4-9 Yearly performance test on low competition jobs.....	60

LIST OF FIGURES

Figure 2-1. Visualized results comparison of table 4	24
Figure 3-1. Two supplement constraints – priority and budget.....	28
Figure 3-2. Job type table and contractors.....	30
Figure 3-3. Three scheduling queue models.....	34
Figure 3-4. Step one: display the current HOAs’ job type table.....	37
Figure 3-5. Step two: show the partial HOAs’ release table based on budget (15, 2000)	38
Figure 3-6. Step three: display the categorized scheduling results.....	39
Figure 3-7. Step four: display the categorized scheduling time flow	40
Figure 3-8. Step five: display the categorized scheduling statistics results.....	41
Figure 3-9. Super-ASH class organization	44
Figure 4-1. Visualization on low cost jobs with time 5 scheduling results	47
Figure 4-2. Visualization on high cost jobs with time 5 scheduling results	48
Figure 4-3. Visualization on mix cost jobs with time 5 scheduling results	49
Figure 4-4-1. Same amounts \$10,000 different intervals for high cost jobs.....	51
Figure 4-4-2. Same amounts different intervals for high cost jobs - AFT, AAFT, AS and AAS.....	52
Figure 4-5-1. Same amounts \$10,000 different intervals for low cost jobs.....	53
Figure 4-6-1. Same interval 6 different amounts for high cost jobs	54
Figure 4-6-2. Same interval 6 different amounts for high cost jobs with AAFT, AFT, AS and AAS.....	55
Figure 4-7-1. Same interval 6 different amounts for low cost jobs	56
Figure 4-7-2. Same interval 6 different amounts for low cost jobs with AAFT, AFT, AS and AAS.....	57
Figure 4-8 Yearly performance test on high competition Jobs.....	60
Figure 4-9 Yearly Performance test on low competition jobs	61
Figure 6-1-2. Add job type one by one once click the “Add” button in “Type” category	66
Figure 6-1-3. The created Job Type Table after one by one “Add” and default generator	66
Figure 6-2-1. Adding jobs one by one	67
Figure 6-2-2. One job added into the Job List	68
Figure 6-2-3. Using job generator to generate jobs	68
Figure 6-2-4. Job List table shows all the added and generated jobs	69
Figure 6-3-1. Make scheduling window for budgets and scheduling method setting	70
Figure 6-3-2 Scheduling results based on the above configurations	70

1. INTRODUCTION

Homeowner Associations are in charge of scheduling many repair jobs every day. Good job scheduling plays an important role in homeowner satisfaction. Job scheduling is the task of efficiently allocating limited resources towards jobs to be performed in a given timeframe. An effective allocation will result in an efficient scheduling of a set of activities on a set of resources. This is a basic problem in both computer science research and real application research. Due to the difficulty of this problem, most of the research in this field has incorporated some potentially inapplicable assumptions, such as all the jobs arrive at the same time, each job's processing time is one unit, and so on. The goal of this project is to develop an applicable and configurable job scheduler capable of aiding the homeowner associations (HOAs) in achieving efficient scheduling results by providing immediate feedback on when the customers' requests will be completed and by fulfilling jobs in a timely manner.

Depending on the jobs, scheduling performance is optimized by using different scheduling algorithms. Some algorithms aim to minimize the average response time, while others are concerned with variables such as the average actual processing time or average actual waiting time. For those specific goals to be achieved, the scheduling algorithm plays a significant role during the process. Algorithms that assign the start and end times schedule the jobs waiting in a queue. Jobs' processing time is optimized by constraints based on certain objective functions. Those constraints are typically either time constraints (completion deadline) or resource constraints (competition for the same resource or limited resources). To develop an efficient scheduling algorithm for HOAs,

we must have a well-defined scheduling problem that specifies all the particular objectives, considerations, constraints and requirements.

Based on the HOAs scheduling problem, we began by researching the classical and original scheduling algorithms. The first scheduling algorithm we considered was the Greedy task-scheduling algorithm [CLR90]. This algorithm will efficiently schedule unit-time jobs on a single processor. The other scheduling algorithms considered in my project are First In First Out (FIFO), Shortest Processing Time First (SPTF), and the Semi-clairvoyant R algorithm [BMLP04]. These algorithms are differentiated in their methods and objectives. The new scheduling algorithm, the Super-ASH algorithm, was designed based on HOAs' scheduling problem. The new algorithm can effectively ameliorate the advantages and disadvantages of the other algorithms. This algorithm is suitable for alternate requirements and configurable for different needs under different conditions.

Based on the analysis of these classical algorithms, plus considerations of the real applications in HOAs, we developed Super-ASH. Super-ASH is capable of handling jobs with random arrival times and scheduling them without association relationship constraints according to their particular types under budget and resource constraints. Although those jobs are not completely predictable, there is some pre-knowledge about the upcoming jobs, such as the estimated process time, approximate cost, and the approximate frequency. We call a scheduler that operates under such conditions of limited pre-knowledge a semi-clairvoyant scheduler. The Super-ASH algorithm is such a semi-clairvoyant scheduler. Our main contribution is to compare how different semi-

clairvoyant scheduling algorithms perform on HOA scheduling problem. This involves analyzing these algorithms on the HOA scheduling problem under various different conditions. Our results shed light on the relatively complicated relationship between the initial conditions and the scheduling algorithms.

This report illustrates the development process and implementation of Super-ASH. The report is divided into four sections. The first section describes the background and related work. The second section focuses on the design and implementation of the Super-ASH. Section 3 describes the experiments we conducted and also gives our analysis of them. The last section discusses the conclusions and significance for future work based on the results of this project.

1.1 Scheduling Problems

In general, scheduling problems are denoted as $\alpha|\beta|\gamma$ [GKLL79], where α stands for the machine environment, β stands for the various side constraints and characteristics, and γ stands for a desirable criterion. In this project, the HOAs scheduling problem is defined according to the standard notation $\alpha|\beta|\gamma$. At the beginning of each scheduling problem, we have a set of jobs J , numbered from 1 to n . Each job, J_i , has a release time r_i , a processing time p_i , a deadline d_i and weight w_i . A job J_i can then be denoted as tuple (r_i, p_i, d_i, w_i) . If a job must be finished without interruption, it called non-preemptive scheduling environment. In a preemptive environment, a job in process can be interrupted and continued at a later point of time.

Next we focus on β - side constraints and characteristics, which will specify the machine environment and many possible side limitations and characteristics, such as logical dependence or independence among jobs, and the jobs arrival internals. Of course, the job release time should always be before its actual processing time and finish time. No matter which side constraints or characteristics are taken into consideration, the final goal of scheduling is to produce a good schedule for certain objectives. The better understanding on the scheduling environment characteristics and special constraints will surely result in more efficient scheduling performance.

However, there are no consistent rules to define what kind of scheduling algorithm is good because it is application dependent. Sometimes the goal is to minimize the completion time, and sometimes the goal is to maximize the total weights gained from matching the deadline of each job. Formally, the goal is the desirable criterion γ , and the purpose of the scheduling algorithm is to construct the algorithm to optimize this criterion.

1. 2 Define HOAs Scheduling Problem

In this section, we define the HOA scheduling problem. First, we consider the processing environment. Only one association is in charge of scheduling all the requested repair jobs and assigning those jobs to different types of contractors. Also, the HOAs' environment is non-preemptive. This means there is no interruption during the processing of jobs. That is, once a job starts, the job will keep running until it is completed.

Given randomly arriving jobs, the HOA scheduler's task is to continually allocate and reallocate the constrained resources as they arrive and are completed. This kind of problem is classified as a classical resource-constrained scheduling problem. This class of problems can be summarized as the following: the incoming project set which consists of J independent jobs labeled as $1, \dots, n$. The set of jobs referred to as $J = \{1, \dots, n\}$ respectively, and categorized according to the association pre-defined job type table. All the jobs in any category are undividable as well as independent.

Each job requires certain resources to be performed. The set R , stands for the budgets and contractors used by the HOA. The processing time of job j is denoted as P_j and its request for resource R is denoted by R_j . Once a job starts, it cannot be preempted. Each job also has its own time descriptor. This descriptor consists of a time of receipt, a release time, a start time, and a completion time. Without loss of generality, we assume that certain operations such as depositing money into an account, releasing jobs to contractor, and similar operations, take no time to perform.

All the parameters in the scheduling are assumed to be non-negative and integer valued.

The objective is to determine a schedule with minimal (actual) average flow time, and an (actual) average stretch, so that both goals are fulfilled: timely feedback and efficient resource use.

2. BACKGROUND and RELATED WORK

The design of scheduling algorithms has a significant history as scheduling problems arise in a variety of settings. In this section, we will discuss some of this history and some of the common scheduling algorithms.

2.1 Performance Measures on Scheduling Algorithms

As we discussed before, each algorithm has its own objectives. Some algorithms focus on scheduling jobs within their deadlines in order to incur the lowest penalties for going over the due dates. In addition to the deadlines and missed deadline penalties, both average flow time and average stretch are important measurements for job scheduling algorithms.

Average flow time is the average response time for each job. And the average stretch is the average proportion between the actual processing time and predicted processing time.

The following formula will give the detail description on these two objectives. For a set of jobs J_i where $i = 1, 2, \dots, n$, each job has release time r_i (non-negative integer), and processing time p_i (non-negative integer). After the scheduling of jobs, each job has its

own completion time c_i . The average flow time is $\frac{1}{n} \sum_{i=1}^n \{(c_i - r_i + 1)\}$ and the average

stretch is equal to $\frac{1}{n} \sum_{i=1}^n \left\{ \frac{(c_i - r_i + 1)}{p_i} \right\}$. During the scheduling of jobs, active jobs can be

divided into two categories: *partial jobs* and *total jobs* [BLMP04]. Partial jobs have already been began in the past by the scheduler; whereas, the total jobs have never been executed by the scheduler. All of the above parameters are calculated and compared in this project.

2. 2 Original Scheduling Algorithms

2. 2. 1 Greedy UnitTask Scheduling Algorithm

The Greedy algorithm, as its name implies, always makes the choice that looks the best at that time. That is, it tries to make a locally optimal choice that could lead to the final global optimal solution. However, Greedy algorithms rarely find the globally optimal solution. This is because they usually do not operate exhaustively on all the possible data. The Greedy task-scheduling algorithm is effective because in real applications it is easy to implement and often come-up with good approximations to the optimum. Of course, the most important benefit of Greedy algorithms lies in their conceptual simplicity and their computational efficiency. If a Greedy algorithm can be proven to yield the global optimum for given problems class, it typically becomes the actual method of choice.

The Greedy Unit Task Scheduling algorithm takes as input a job set J that includes a set of unit-time tasks with deadlines. Each J_i ($i \in 1, 2, \dots, n$) is independent from the other jobs. When $M = (J, J_i)$ is a weighted set with weight function w , the Greedy unit task scheduling (M, w) returns an optimal subset [CLR90]. By this theorem, the algorithm can be used to find a maximum weight independent job set. This method is an efficient algorithm for scheduling unit-time tasks with deadline and penalties for a single processor $\alpha=1$. Its running time is $O(n^2)$ no matter what kind of sorting algorithm is applied in the process of scheduling.

The Greedy unit task scheduling algorithm is as follows:

Initiate n time slots empty

Sort the n waiting for scheduling jobs by their weights

For ($time=1$; $time \leq n$; $time++$)

Schedule the highest weighted job J_i ($i \in 1, 2, \dots, n$) in the currently waiting queue

If (J_i deadline time slot is empty)

then assign J_i at that time slot

else if (before J_i deadline time slots available)

then assign the latest available time slot in deadline to J_i

Otherwise pick the latest open time slot before J_i 's deadline.

To test an algorithm such as the above, one typically uses a program to generate jobs randomly. Each job has its own description parameters, such as the job number, weights, processing time and deadline. At the beginning, jobs are generated in the waiting queue and the sorter sorts all the active jobs based on their weights. The sorted jobs will then be scheduled in a decreasing weight sequence.

The following tables display two typical results:


Jobno	Weight	Deadline	Greedy Unit Task scheduling 	Time	Job no	The schedule results: <i>The number of late tasks = 0</i> The schedule results: <i>The total penalties = 0</i>
1	59	18		1	3	
2	43	6		2	18	
3	23	1		3	9	
4	29	15		4	8	
5	71	16		5	19	
6	51	16		6	2	
7	99	16		7	15	
8	14	6		8	12	
9	11	16		9	11	
10	30	13		10	4	
11	21	15		11	10	
12	75	8		12	6	
13	97	17		13	14	
14	55	16		14	16	
15	38	7		15	5	
16	56	16		16	7	
17	29	19		17	13	
18	75	2		18	1	
19	39	6		19	17	

Table 2-1. Schedule result for no job late for their deadlines


Jobno	Weight	Deadline	<p style="text-align: center;">Greedy UnitTask scheduling</p> 	Time	Job no	<p>The schedule results:</p> <p><i>The number of late tasks = 4</i></p> <p><i>The total penalties = 94 (8+12+37+37)</i></p>
1	46	7		1	3	
2	50	23		2	23	
3	44	1		3	22	
4	84	9		4	14	
5	29	15		5	16	
6	25	12		6	13	
7	94	13		7	1	
8	8	9		8	9	
9	65	9		9	4	
10	80	10		10	10	
11	37	4		11	20	
12	69	19		12	6	
13	43	11		13	7	
14	40	4		14	8	
15	46	23		15	5	
16	15	12		16	17	
17	81	16		17	18	
18	12	9		18	11	
19	37	22		19	12	
20	49	11		20	21	
21	37	1		21	19	
22	66	3		22	15	
23	97	2		23	2	

Table 2-2. Schedule results for some jobs late for deadlines and got penalties

The scheduling results in Table 2-1 and Table 2-2 show us the efficient scheduling algorithm with respect to minimizing penalties. Table 2-1 and Table 2-2 were calculated without taking into consideration a fixed budget. Its other limitations include an assumption of unit processing time and non-release time. That is, assumes all the jobs can be done in unit-time; in actuality, each job should have its particular processing time. Additionally, the coming jobs could not all arrive at the same moment as each of them has its specific receiving time and release time. However, based on our research, the HOAs do not have a detailed deadline schema and corresponding penalizations on those overdue jobs. Therefore, in this project the deadline is not included as a scheduling

parameter. These results will thus serve as a guide when we incorporate budget constraints.

2. 2. 2 First In First Out (FIFO)

The FIFO scheduling strategy assigns priority to the jobs in the order in which they request processing. The priority of each job is computed by the enqueueer by time stamping all incoming jobs; this project used the automatically generated increased Job ID as a substitute. And this is also the measure to decide each job's priority in the queue. Every time the new jobs come, the enqueueer will add the job to the tail of the queue with a unique Job ID, and the dispatcher will remove the jobs from the head of the queue.

This is a non-preemptive scheduling environment. The greatest benefit of this algorithm is its ease to implement, without fear of starvation at all. Starvation is a resource competition phenomenon in many resource allocation scenarios in which certain sets of processes are perpetually ignored because their priority is not as high as that of other processes [NG02]. Although FIFO is not completely capable of handling the HOA's complicated scheduling problem, it can be improved by combining with the Sc-R algorithm. That is, a scheduling strategy to achieve better scheduling performance by applying the predictable knowledge of the coming jobs during the scheduling. Also, we applied the priority strategy to the highest number queue that holds a number of jobs with long processing time requests. This idea is inspired by a multiple-level queue scheduling algorithm in the operating system. On the other hand, this is also a good reference for comparing the scheduling algorithms.

2. 2. 3 Shortest Processing Time First (SPTF)

The SPTF scheduling algorithm always chooses the job that requires minimum processing time to run first. It surely guarantees that the scheduler can obtain the best average flow time. However, it might ignore those jobs with long service time requests. If the ready queue is saturated, the jobs with long service time requests tend to be left in the ready queue while those jobs with short process time requests receive service. In some extreme cases where the schedule system has little idle time, jobs with large processing time requests will never be served. As a result, the total starvation of those jobs that have large processing time requests may be a serious liability of the scheduling algorithm.

Before the design of the HOA scheduling algorithm, we built up a combinational scheduler that can schedule the same set of randomly generated jobs by applying different scheduling algorithms. They are Sc-R algorithm, FIFO and SPTF respectively.

2. 2. 4 Semi-clairvoyant R algorithm (Sc-R)

Semi-clairvoyant algorithm is an algorithm that will make serial decisions on each input with some knowledge under uncertainty. In our project, we use the predictable job processing time to define the scheduling queue model and size in order to get better scheduling performance. In a study of an online problem similar to ours involving web servers [SRRJ97], it was found that currently most web servers apply First In First Out (FIFO) scheduling instead of the Shortest Processing Time First (SPTF) scheduling even though SPTF can give smaller average flow time. The most important issues in choosing

between these two algorithms are: a) SPTF is 2-competitive, which means the SPTF performance is conceptionally never more than twice as the optimal performance that could be achieved, with respect to average stretch and b) web servers use FIFO because of the fear of long process time job trapped into starvation. Some developers and researchers insist that the web servers should take SPTF as another scheduling choice.

It has been shown that the semi-clairvoyant R (Sc-R) algorithm is $O(1)$ -competitive with respect to average flow time on a single machine. Under certain condition, the semi-clairvoyant R algorithm is $O(1)$ -competitive with respect to average stretch on a single machine. However, the algorithm cannot simultaneously be $O(1)$ -competitive with respect to both average flow time and average stretch [BLMP04]. Because of the scheduling efficiency of Semi-clairvoyance R algorithm, we implement the algorithm as a reference for developing Super-ASH although Sc-R is a preemptive scheduling algorithm.

The Sc-R algorithm can guarantee that at all times each queue of jobs has at most one partial job. This fact is also useful for algorithm analysis. The overall scheduler will put the released jobs into different queues. The queues are numbered as k from 1 to n . A job belongs to queue $k \in (1, 2, \dots, n)$ means that $p_i \in [2^k, 2^{k+1})$. Scheduling from the lowest numbered queue:

If at any time, all the jobs in the queue k , then pick the partial job to execute if one exists, otherwise, pick any one of the total jobs to run;

- If at any certain time, there are two lowest numbered queue m , n ($m < n$) and both of them have jobs in them:
 - If m has exactly one total job J_i , and n has exactly one partial job J_j , then run the partial job J_j ;
 - Under any other situations, run the partial jobs in m if there is one, otherwise run a total job in m .

And this method implemented in my code as following:

```

Class Rscheduler extends Scheduler{
    public Rscheduler(){
        int i;
        jobsClass = new Queue [4];
        for(i=0; i<4; i++)
        {
            jobsClass[i] = new Queue();
        }
    }

    public Job run(int time)
    {
        ...
        for(i=0; i<jobsClass. length; i++)
        {
            count = jobsClass[i]. getItemNum();
            if( count == 0)
            {
                continue;
            }
            else
            ... return (firstJob);
        } ...
    }

```

As the algorithm shows these jobs are executed in a preemptive environment. In Super-ASH, all the jobs will be processed under a non-preemptive condition because of the real application considerations. However, the implementation will give us some light on

developing Super-ASH. Consequently, the HOAs scheduling problems will not be necessary to worry the distinction between the partial and total jobs.

2. 2. 5 Comparisons on Sc-R, FIFO and SPTF

The following tables show one of our test cases and statistical result:

ID	r_i	p_i	Sc-R algorithm			SPTF algorithm			FIFO algorithm		
			s_i	c_i	f_i	s_i	c_i	f_i	s_i	c_i	f_i
1	1	4	1	7	7	1	4	4	1	4	4
2	2	14	16	29	28	69	82	81	5	18	17
3	4	1	4	4	1	5	5	2	19	19	16
4	4	2	5	6	3	6	7	4	20	21	18
5	5	3	8	10	6	8	10	6	22	24	20
6	5	13	30	42	38	43	55	51	25	37	33
7	6	9	43	51	46	16	24	19	38	46	41
8	7	5	11	15	9	11	15	9	47	51	45
9	8	13	52	64	57	56	68	61	52	64	57
10	8	9	65	73	66	25	33	26	65	73	66
11	9	9	74	82	74	34	42	34	74	82	74
Average Flow time			30.45			27.0			35.55		
Average Stretch			3.46			2.9			6.36		

Table 2-3. One example of the combinational scheduler scheduling results

(Note: r_i stands for release time, p_i stands for process time, s_i stands for start time, c_i stands for complete time, f_i stands for finish time, and $f_i = c_i - s_i + 1$)

No. Alg.	5 jobs		6 jobs		7 jobs		8 jobs		9 jobs		10 jobs	
	A. F	A. S	A. F	A. S	A. F	A. S	A. F	A. S	A. F	A. S	A. F	A. S
Sc-R	25.0	4.65	20.0	2.38	27.43	3.15	22.25	2.82	31.4	3.74	47.7	3.877
SPTF	21.2	1.92	18.67	2.11	24.43	2.33	20.75	2.33	27.78	2.89	45.5	3.36
FIFO	28.4	8.41	28.16	9.49	38.42	12.25	25.63	4.31	38.0	6.83	59.9	9.13

(Note: A.F stands for average flow time and A.S stands for average stretch)

Table 2-4. Enumeration of some results on different number of jobs

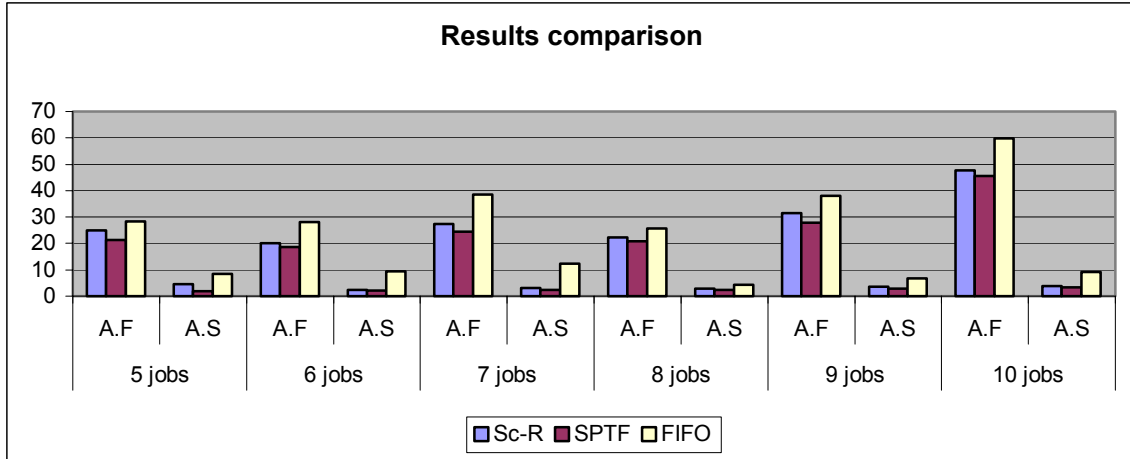


Figure 2-1. Visualized results comparison of table 4
 (Note: A. F stands for Average Flow Time, A. S stands for Average Stretch)

The above table shows the random results of a different number of jobs' schedules. The calculation on average flow time and average stretch. This is not a formal statistics analysis, but it is given anecdotal evidence. Because the scheduling performance mostly depends on randomly generated jobs, it is not possible to make real cover-all statistical comparisons. However, the data can present the overview performance of different algorithms. Through the data and the chart, the SPTF scheduler shows significant difference from the other two with respect to average flow time and average stretch. But this algorithm still has some fatal flaws, such as a long process time job that leads to starvation.

2. 3 Inspirations from Our Initial Research

These researched and tested scheduling algorithms partly satisfy the requirements of HOAs scheduling problems. For example, a Greedy unit task-scheduling focuses on

meeting deadlines in order to minimize the penalties, a Semi-clairvoyant R algorithm focuses on achieving better average flow time, a FIFO focuses on obtaining the easy implementation and fairness among jobs, and a SPTF just focuses on reaching the shortest waiting time in the queue and average turnaround time. Obviously, none of them can independently become a satisfactory solution to the HOAs scheduling problem. This is because these algorithms did not take the budget as a pre-condition in scheduling the jobs. Although the Greedy unit task scheduling algorithm did consider the penalties on budget aspect, it assumed the budget is always enough to afford all the costs of those jobs. In our situation, it may be the case that jobs have to be put on hold because of insufficient budget. That is, jobs can be scheduled only as long as there is enough money to pay for them. To find a feasible way out, we can try to apply those algorithm methods and combine them with a kind of trade off to come up with the Super-ASH scheduling algorithm.

During the development of Super-ASH, we designed an ASH scheduling algorithm that uses the Semi-clairvoyant R algorithm. The algorithm assures that the partial knowledge about the coming jobs, such as the processing time of the coming repair and maintenance jobs, can be taken into account in the performance and be used to apply the priority methodology embraced in the Greedy unit task scheduling algorithm.

Two important issues not taken into consideration throughout the original scheduling algorithms are budget and cost per job. Another limitation of the tested algorithms are the assumptions: Greedy unit task scheduling algorithm assumes every job can be done in

one unit time, FIFO assumes each job's ID number is its real timestamp, Semi-clairvoyant R algorithm assumes that there is always enough budget to get the job done, and SPTF assumes that each job has no deadline limitation. So, in the design and implementation of Super-ASH, we took away all these assumptions and developed an algorithm that works under the real constraints of HOA job scheduling.

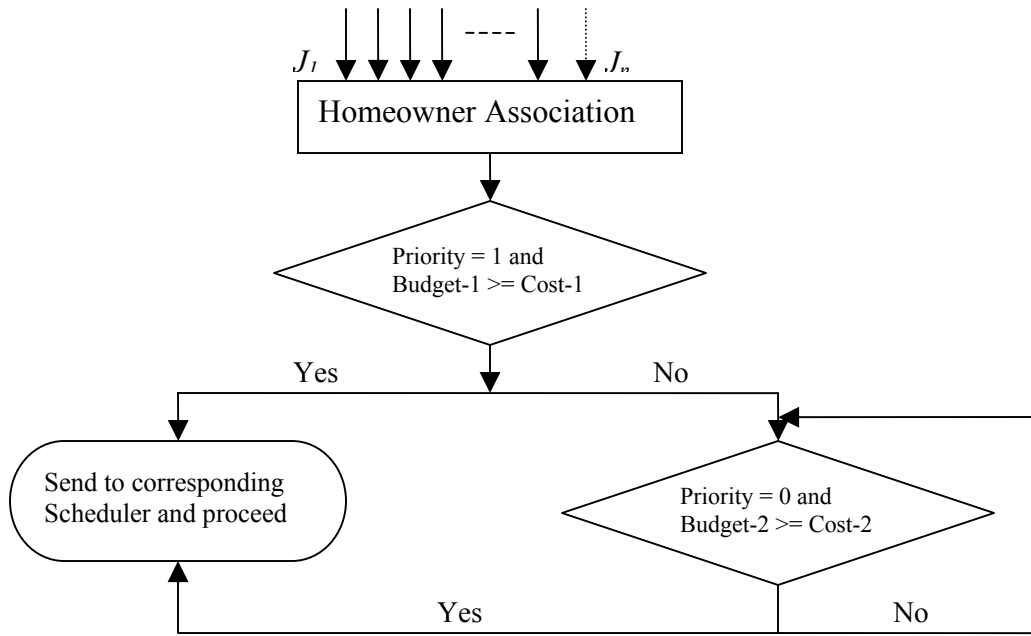
The Super-ASH algorithm receives and processes the coming jobs in partial amounts. In serving each request the Super-ASH algorithm has a choice of several alternatives, each with a particular cost. The alternative chosen at one step may influence the costs of alternatives on future requests. This characteristic requires a semi-clairvoyant algorithm to make appropriate decisions for the scheduler.

3. DESIGN and IMPLEMENTATION

As we discover, there are still many constraints that have not been taken into consideration during the original scheduling process, such as the cost, the budget, and emergency and non-emergency jobs. It is necessary to add those factors into Super-ASH, so that it will be as relevant to the HOA scheduling problem as possible.

3. 1 Supplemental Constraints

We begin with two additional factors – Priority and Budget. The requested jobs from the customers can be divided into two categories: emergency and normal. Each category is presented as Priority = 1 and Priority = 0 respectively. Naturally, the jobs with priority value “1” will be considered ahead of jobs with priority value “0”. However, prioritizing the jobs is not enough to assure whether these jobs can be scheduled. We also need to consider another critical real life issue – budget. The cost of jobs will be compared with the current available budget; if the budget is sufficient to process the jobs, then jobs will be sent to a different scheduler to be scheduled. Meanwhile, the current budget will subtract the costs of the released emergency jobs. Then the rest of the budget can be used to pay the normal jobs. The first step on scheduling the jobs will be as the follows:



Note: Budget-1 represents the amounts before scheduling all the requested jobs; Budget-2 represents the amounts left after releasing the emergency jobs. Cost-1 equals the costs of to-be-released emergency jobs; Cost-2 equals the costs of to-released normal jobs.

Figure 3-1. Two supplement constraints – priority and budget

After adding these two parameters, two more variables are decided upon the jobs' time of receipt and release time. The time of receipt represents when the HOAs receive the requested the jobs from the homeowners. The release time represents when the HOAs officially release the job to the contractors when the budget is adequate to afford those jobs.

As we discussed before, the disadvantage to SPTF as a scheduler is that it can have starvation issue. To avoid starvation, our algorithm applies the three different queues models to assign priorities to different queue levels. Moreover, all the jobs in the same-leveled queue will be processed based on First In First out scheduling method. So, within

a queue starvation cannot happen. Starvation might happen between different queue levels, but this should be rare as it would receive a larger amount of job requests than would naturally occur in the HOA setting.

3. 2 Job Type Table and Contractors

The job type table and the contractors are the two schemas are used in homeowner associations. The association is the only entity that holds the rights to read, write, and modify the job type table. In other words, the associations are in charge of maintaining the job type table according to their own contracts and experience.

This job type table behaves as a reference to the schedulers. It directs a scheduler on how to work under the specific constraints and conditions. Schedulers can look at reference data from HOAs' job type table, such as the job's corresponding contractor, the job's approximate cost, and the job's approximate process time. If the job type table accurately reflects the properties of the repair jobs, then the scheduling results will be a more efficient arrangement of those jobs. Also, more experienced HOAs can obtain more accurate data through long period time of experience and knowledge.

Each HOA has various contractors in charge of different types of jobs. There might be a contractor for plumbing, a contractor for painting, a contractor for roofing and some contractors who have other specific skills. Because the contractors are mutually independent from each other, they can do their jobs concurrently without any conflicts. Consequently, the Super-ASH scheduler will categorize the jobs which correspond to the

job type table first, and then schedule them independently. As illustrated in the below diagram, jobs come in and then are categorized into different independent schedulers according to their types:

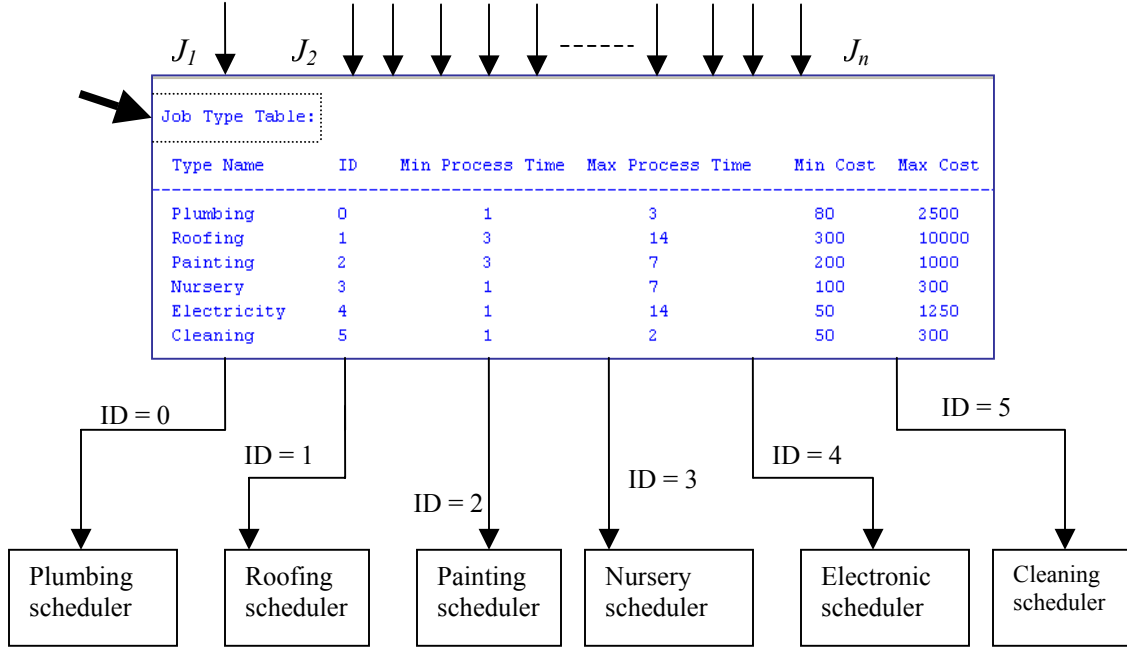


Figure 3-2. Job type table and contractors

After the homeowner association releases the jobs and then sends these jobs to corresponding contractors, the contractors perform the further scheduling.

3.3 Configuration Objectives

Different types of schedulers are needed handle different requirements and distinctive objectives. Because the main purpose of developing a scheduling algorithm for HOAs is to satisfy the homeowners' requests in a timely manner and efficient way, the developing Super-ASH algorithm will assure minimizing the two most basic objective functions. One

is the *actual average flow time* (AAFT), $\frac{1}{n} \sum_{i=1}^n \{(c_i - r_i' + 1)\}$ where r_i' stands for the

HOAs release time. The second is the *actual average stretch* (AAS), which equals

$\frac{1}{n} \sum_{i=1}^n \left\{ \frac{(c_i - r_i' + 1)}{p_i} \right\}$. The goal of trying minimizing average flow time and minimal

average stretch will focus on finishing all the jobs by the earliest time possible. The goals of minimizing the actual average flow time and actual average stretch will concentrate on releasing the jobs as soon as possible. Actually, we cannot always achieve the best performance by reaching the lowest AAFT and AAS. Sometime we need to do some trade off between AAFT and AAS. For example, there are two jobs, one is roof job and its process time is around 30 time units, another is a cleaning job with process time around 1 time units. If we schedule the roof job in 2 months and the cleaning in one day, there will be around thirty time units wasted in the total time flow but the stretch do not have any affects. On the other hand, for two jobs with the same process time, the actual average flow time becomes more important than the actual average stretch. In our algorithm, we will concentrate on obtaining the best actual average flow time during our scheduling if it is necessary to make a trade-off between these two measures. However, we cannot set the configurations only based on our goals, we need to consider the application reality as well.

3. 4 Configuration Parameters

3. 4. 1 Budget Intervals and Amounts

The frequency with which the HOAs' budget will be recharged and how much the

HOAs' budget will be refilled directly affects a job release time r_i' . If the budget can be

filled as much as possible and as often as necessary, then jobs need not to be delayed, waiting until there is sufficient money to send them to the related contractor.

Consequently, the customers' repair job requests can be released immediately by the HOAs. As a result, the average flow time equals to the actual average flow time, and the average stretch is the same as the actual average stretch. However, this is an ideal condition during scheduling. It will not happen all the time, so we need to configure Super-ASH by setting our own budget recharging intervals and amounts, as follows:

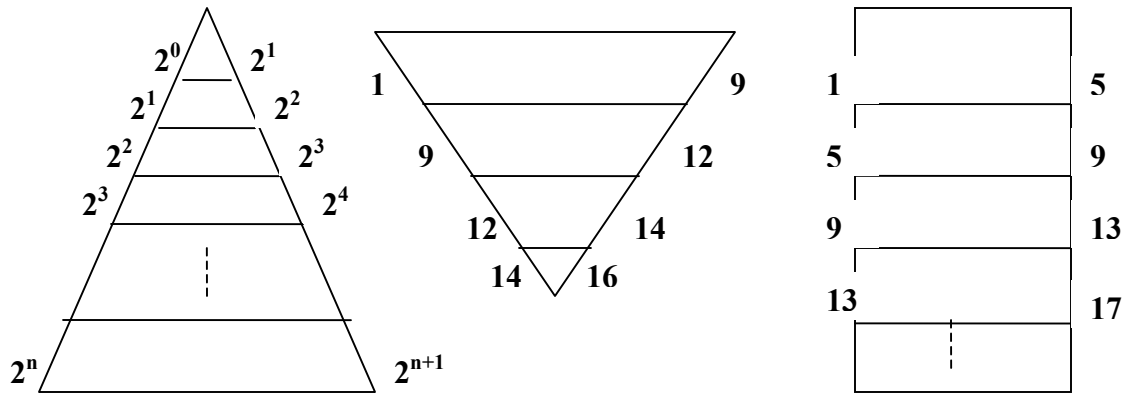
```
public void setBudget(int interval, int delta) {  
    budgetInterval = interval;  
    budgetDelta = delta;  
}  
...  
admin. setBudget(30, 10000);    ...
```

Also, our algorithm will assume the HOAs have some amount money as the start-up condition of the entire system. That is, at the very beginning, the HOAs account will start with certain amount of money, and get money at the end of each specified time interval. Actually, the purpose of adding the budget recharging intervals and amounts is not only to satisfy the real application request, but also to offer opportunities to improve the whole algorithm's performance. Some smart ways exist to configure your data to get better AAF and AAS even if the amounts are the same. For example, the AAA homeowner association has a \$10,000. 00 deposit every 30 days. They have many ways to recharge their accounts, such as *admin. setBudget (30, 10000)* or *admin. setBudget (15, 5000)*. Both of them get \$10,000.00 per thirty days with these two different depositing ways. However, the scheduling results are totally different. More detailed results will be revealed in Section 4.

3. 4. 2 Queue Models

A variety of queues are used during scheduling. For example, the emergency job queue, the normal job queue, and the scheduling queue. Each queue has its own specific width and length. Obviously, the length of the queue is the number of items in this queue. The width of the queue is actually the range of values stored in it. For instance, there is a Q_1 [2, 8] that its width is $8 - 2 + 1 = 7$.

Both width and length affect the scheduling performance to some extent. The queue width of the contractors' scheduling queues will especially affect the performance according to our research. In our program, the length of the queues will be assumed to be unbounded, and the queue width will be used to specify the queue models. Different queue models can benefit different groups of jobs. Most of a job's scheduling performance is highly dependent on the jobs themselves. Thus, different choices on the queue models bring different results and different performance. In our project, we use the semi-clairvoyance strategy to design the width of different queue models and build up three scheduling queue models based on the pre-setup job type table: Power2 scheduling queue model, UpdownTriangle scheduling queue model, and Even scheduling queue model, illustrated as below:



Note: From the left to the right, they are Power2 Queue, UpdownTriangle Queue and Even Queue respectively. And the numbers listed on their edges means its width on that layer, for example [2^n , 2^{n+1}).

Figure 3-3. Three scheduling queue models

The Power2 scheduling queue and the Even scheduling queue are expandable queues, therefore both of them beat the UpdownTriangle scheduling queue in terms of flexibility. The middle queue model width is really constrained by the all the possible jobs' processing times. But this is just one aspect of comparing these models; they also show their different efforts during the scheduling. Super-ASH allows the HOAs to select more than one model to schedule their jobs if they are not sure which one they should choose. If you do not specify a choice, Super-ASH will pick the Power2 scheduling model as its default.

The following code implements the Power2, Even and UpdownTriangle queues in the Super-ASH algorithm:

```
public static void setQueueModel(int model)
{
```

```

        if (model >= POWER2 && model <= EVEN)
            queueModel = model;
    }

    public int getJobLevel(int pocessingTime)
    {
        int jobLevel;

        if (queueModel == POWER2)
            jobLevel = getPower2JobLevel(pocessingTime);
        else if (queueModel == UPDOWNTRIANGLE)
            jobLevel = getTriangleJobLevel(pocessingTime);
        else
            jobLevel = getEvenJobLevel(pocessingTime);
        return (jobLevel);
    }

    private int getPower2JobLevel(int pocessingTime)
    {
        int jobLevel = 1;
        while (pocessingTime >= (int)Math. pow(2. 0, (double)jobLevel))
        {
            jobLevel++;
        }
        Return (jobLevel);
    }

    private int getTriangleJobLevel(int pocessingTime)
    {
        int jobLevel;

        if(pocessingTime<9)
        {
            jobLevel = 1;
        }
        else if(pocessingTime<12)
        {
            jobLevel = 2;
        }
    }

```

```

        else if(pocessingTime<14)
        {
            jobLevel = 3;
        }
        else
        {
            jobLevel = 4;
        }
        Return (jobLevel);
    }

private int getEvenJobLevel(int pocessingTime)
{
    int jobLevel;

    if(pocessingTime < 1)
    {
        jobLevel = 1;
    }
    else
    {
        jobLevel = (pocessingTime - 1) / 4 + 1;
    }
    Return (jobLevel);
}

```

Because job-scheduling performance is highly job dependent, none of these three scheduling queue models stand out from the others under arbitrary conditions. Nevertheless, there are still strategies for making a good choice among these three. This will be demonstrated in Section 4.

3. 4. 3 Statistical Results and Other Display Options

As mentioned before, there are many choices to be made by the HOAs before scheduling. If the HOAs do not have a concept of Super-ASH is doing, the whole scheduling process is an absolute black box. Therefore, there is no way to help them determine what options

to set for will cause better scheduling performance. In order to guide the user, Super-ASH offers a series of statistical results and mid-procedure results display options, such as the average flow time, actual average stretch, release job table, every contractor's scheduling time flow, etc. The users can choose to define interval variables used by the scheduling for scheduler, such as the budget interval and amount. The algorithm may appear as a black box, but the complete scheduling process, from the beginning to the end, is actually transparent and traceable. To understand better, the snapshots below will give the overall picture of the process flow of Super-ASH scheduling:

Job Type Table:					
Type Name	ID	Min Process Time	Max Process Time	Min Cost	Max Cost
Plumbing	0	1	3	80	2500
Roofing	1	3	14	300	10000
Painting	2	3	7	200	1000
Nursery	3	1	7	100	300
Electricity	4	1	14	50	1250
Cleaning	5	1	2	50	500



Figure 3-4. Step one: display the current HOAs' job type table

This *Job Type Table* is modified and maintained by the homeowner associations. It includes the job type name, job type ID, minimum and maximum process times, minimum and maximum costs. This table is an important reference for HOAs in configuring their schedulers.

Release jobs under the current budget setting, \$2000 per 15 days

```

*****
Budget is set $2000 per 15 day.
*****
Job Release Table:

```

job ID	Type Name	priority	cost	receive time	release time	expected complete time
1	Electricity	0	1011	2	2	13
2	Electricity	0	472	15	15	23
3	Electricity	0	180	26	26	30
4	Electricity	0	458	40	40	54
5	Electricity	0	549	51	51	61
25	Roofing	0	7215	1	61	72
6	Electricity	0	942	68	76	87
7	Electricity	0	1105	88	88	102
8	Electricity	1	526	101	101	114
9	Electricity	0	1169	122	122	131
10	Electricity	0	261	133	133	143
26	Roofing	0	6080	73	136	139
11	Electricity	0	1084	142	151	154
12	Electricity	0	647	161	161	162
13	Electricity	0	123	176	176	184
27	Roofing	0	2947	139	181	192
14	Electricity	0	1215	196	196	207
15	Electricity	1	722	216	216	229
16	Electricity	0	542	235	235	248
28	Roofing	0	6653	173	241	247
17	Electricity	0	1243	248	256	264
18	Electricity	0	1232	263	271	272
31	Roofing	0	390	273	273	283
19	Electricity	0	641	275	275	279

Figure 3-5. Step two: show the partial HOAs' release table based on budget (15, 2000)

At the top of the HOA *Job Release Table*, the current budget interval and amount are shown. There are eleven columns in the table. They are unique job ID, job type name, job priority number (“1” for emergency, “0” for normal), job cost, receive time, process time, start time, complete time, finish time, and actual finish time respectively. Since these jobs are waiting in the queue to be released to corresponding contractors, they do not have a start time, complete time, and (actual) finish time. Thus they are all initialized as “0”. Each job’s release time is given by the releasing procedure Figure 3-1.

Scheduling algorithm is set to ASH.										
Job classification algorithm is set to POWER2.										

Job Table:										
Plumbing jobs:										
job ID	Type Name	priority	cost	receive time	process time	release time	start time	complete time	finish time	Actual Finish time

Roofing jobs:										
job ID	Type Name	priority	cost	receive time	process time	release time	start time	complete time	finish time	Actual Finish time

25	Roofing	0	7215	1	11	61	61	71	11	71
26	Roofing	0	6080	73	3	136	136	138	3	66
27	Roofing	0	2947	139	11	181	181	191	11	53
28	Roofing	0	6653	173	6	241	241	246	6	74
31	Roofing	0	390	273	10	273	273	282	10	10
29	Roofing	0	5384	216	14	316	316	329	14	114
30	Roofing	0	9063	243	14	406	406	419	14	177
32	Roofing	0	8380	325	4	466	466	469	4	145

Painting jobs:										



Figure 3-6. Step three: display the categorized scheduling results

The released jobs are then sent to different contractors. The contractors will start the schedulers and schedule the classified jobs. The scheduling results will be shown as above. Because of the page limitation, we truncate one of contractors'-Roofing scheduling results. The table shows the scheduling method and the applied queue model, followed by categorized scheduling results. The results list each job's start time, complete time, and (actual) finish time. This information demonstrates the final results of each contractor, yet it does not show how each job processes. In step 4, it gives the full picture on each job process.

Since Super-ASH acts as a job scheduler simulator, it will schedule every released job until all of them to be scheduled. So, if you would like to check the number of jobs is left

in the system at the end of each simulation at certain time slot, you can simply check the scheduling results table and their complete times. Based on their complete times, it is really easy to figure out how many jobs still waiting in the queue.

Roofing job schedule:		Electricity job schedule:	
Time	Job ID	Time	Job ID
61	25	2	1
62	25	3	1
63	25	4	1
64	25	5	1
65	25	6	1
66	25	7	1
67	25	8	1
68	25	9	1
69	25	10	1
70	25	11	1
71	25	12	1
136	26	15	2
137	26	16	2
138	26	17	2
181	27	18	2
182	27	19	2
183	27	20	2
184	27	21	2
185	27	22	2
186	27	26	3
187	27	27	3
188	27	28	3
189	27	29	3
190	27	40	4
191	27	41	4

more...

Figure 3-7. Step four: display the categorized scheduling time flow

The categorized scheduling time flow gives detailed information on each job. The left column stands for the time slots, and the right column stands for the job ID.

Roofing jobs: There are 6 jobs completed in the 1 full year: Year Average Flow Time = 9.166666666666666 Year Actual Average Flow Time = 64.66666666666667 Year Average Stretch = 1.0 Year Actual Average Stretch = 9.124819624819624 There are 2 jobs completed in part of the 2 year: Part Year Average Flow Time = 9.0 Part Year Actual Average Flow Time = 161.0 Part Year Average Stretch = 1.0 Part Year Actual Average Stretch = 24.44642857142857 There are total 8 jobs completed within 2 years: Average Flow Time = 9.125 Actual Average Flow Time = 88.75 Average Stretch = 1.0 Actual Average Stretch = 12.95522186147186	Electricity jobs: There are 23 jobs completed in the 1 full year: Year Average Flow Time = 8.956521739130435 Year Actual Average Flow Time = 10.73913043478261 Year Average Stretch = 1.0163879598662207 Year Actual Average Stretch = 1.6132259045302524 There are 1 jobs completed in part of the 2 year: Part Year Average Flow Time = 1.0 Part Year Actual Average Flow Time = 1.0 Part Year Average Stretch = 1.0 Part Year Actual Average Stretch = 1.0 There are total 24 jobs completed within 2 years: Average Flow Time = 8.625 Actual Average Flow Time = 10.333333333333334 Average Stretch = 1.0157051282051281 Actual Average Stretch = 1.5876748251748252
--	---

more...

Figure 3-8. Step five: display the categorized scheduling statistics results

The last step, step 5, will give categorized scheduling results which shows the AFT, AAFT, AS and AAS based on the all the scheduled jobs in a certain time period, in this example, it shows the scheduling results yearly. In each category, there are the number of jobs completed in each year, and the (actual) average flow time, and the (actual) average stretch of each year. Additionally, the overall (actual) average flow time and (actual) average stretch are demonstrated at the end.

The above steps are the regular default steps. If the HOAs would like to compare the different results under different configurations, they also can make multiple configurations as needed, such as setting the budget as (15, 5000) or (30, 10000) at the same time. The flexibility in Super-ASH will be explored in the following sections.

3. 5 Super-ASH Algorithm

3. 5. 1 Super-ASH Overview

Super-ASH combines various mathematical models for achieving efficient scheduling results with respect to the Average Flow Time, Actual Average Flow Time, Average Stretch and Actual Average Stretch. Also, it got rid of some potentially inapplicable assumptions, such as the jobs' same arrival times, the job unit process time, and so on. Super-ASH considers the jobs exactly as they appear in the real application, such as arbitrary arrival time, various process time, different expenses of each job, several contractors to make the job process parallelizable etc. Therefore, this is a scheduling algorithm that is more appropriate for dealing with HOAs' realistic data compared with original FIFO, SPTF and Greedy unit task scheduling algorithms.

Besides its applicability, another outstanding characteristic is its flexibility. It allows the HOAs to configure Super-ASH in different ways based on their own conditions and regulations. They can modify the job type table based on their association's own experience, change the budget interval and amounts depending on the facts, and select the appropriate scheduling queue model for different types of jobs in order to get better scheduling performance.

3. 5. 2 Class Organization

There are nine major classes in Super-ASH implementation. They are *JobAdministrator*, *ScheduleSimulator*, *Job*, *Jobtype*, *JobQueue*, *JobGroup*, *FIFO_Scheduler*, *SFP_Scheduler* and *ASH-Scheduler*. *JobAdministrator* manages all job groups in a job

type table and releases those jobs according to the budget; then it classifies the released jobs into each job group. *ScheduleSimulator* simulates the homeowners' requests and generates repair jobs, then selects the scheduler type and display options. The *Job* class maintains all attributes of a job, including job basic attributes such as job type, start time, and finish time. *Job* class has methods for running at a specific time, for sorting jobs in a collection and for displaying job attributes. The *JobType* class maintains all attributes of a job type, including process time and cost range for a specific job type. The *JobQueue* class holds a sequence of Jobs. A *JobGroup* is associated with a job type. The *JobGroup* class holds all jobs of a type. Each job group makes its schedule independently with various algorithms. *FIFO_Scheduler*, *SFP_Scheduler* and *ASH-Scheduler* are the three schedulers used for job scheduling.

The relationship among them is shown as follows:

4. EXPERIMENTS and ANALYSIS

This project aims to develop a configurable scheduling algorithm by getting rid of some potential assumptions and taking the real application parameters into consideration.

Therefore, the test cases are designed to verify these two aspects of the algorithm as well as the performance. For the purposes of comparison, we added two more original scheduling algorithms – FIFO and SPTF into the test cases.

4.1 Discoveries in Experiments

The complete test case plan is divided into two parts: a white box test and a black box test. The white box test is a code-based testing. That is, all the tests are based on the understanding of the code. In the white box test, we focus on the precision and accuracy of the scheduling results. Most of the white box tests focused on code debugging and brainstorming, and then the details skipped in this paper. We will reveal some interesting discoveries in black box test cases and results.

The black box tests we re-subdivided it into five categories: job number tests, job cost tests, budget interval and amounts tests, scheduling model tests, and yearly performance tests. To make the results clear on what issues are compared, we only chose to test two types of jobs, roofing and electricity. Because these present two extreme conditions and cover almost all the possibilities, we only listed and calculated the roofing and electricity jobs in our tests. The roofing jobs are the long period time jobs with high costs, and the electricity jobs are the short period time jobs with low costs.

4. 1. 1 Number of Jobs

In Super-ASH, jobs can arrive at arbitrary times. So it is impossible to directly set the number of incoming jobs. In order to figure out the performance under a different number of jobs, we changed the time period to change the number of jobs. And the time parameter is treated as the number of days. In increasing order, we set the time as 5 days, 10 days, one month, a half year and one year, and the number of the jobs ranged from 10 to 1,000. The test sample data and diagram are shown in Table and Figures 4-1, 4-2 and 4-3.

4. 1. 2 Cost of Jobs

The budget is a critical issue in the HOA problem. If the budget is not sufficient to afford a job, then the job must not to be sent to the contractor until there is enough in the budget to cover the payments. If the job's cost is very high, the time period to wait for the deposit into the budget account will be relatively longer than low cost jobs. The actual

average flow time (AAFT) is equal to $\frac{1}{n} \sum_{i=1}^n \{(c_i - r_i' + 1)\}$ and the actual average stretch

(AAS) is $\frac{1}{n} \sum_{i=1}^n \left\{ \frac{(c_i - r_i' + 1)}{p_i} \right\}$ where r_i' stands for the HOAs release time, the r_i' really

affects Super-ASH's actual performance. If it is postponed for a long time, the value of Actual Average Flow Time and Actual Average Stretch will rise. Therefore, different costs of jobs affect our objectives – minimal Actual Average Flow Time and Actual Average Stretch.

Since roofing is the most expensive job class and electricity is the lowest cost job class, they are used to distinguish the performance differences. As shown in the job type table, the roofing costs range from \$300 to \$10,000 and the electricity costs range from \$50 to \$1,250. For the low-cost jobs table, it shows the scheduling results on electricity jobs-low cost job. And the high-cost jobs table demonstrates the results of only scheduling the high cost jobs-roofing. The detailed results are shown in three parts, low-cost jobs, high-cost jobs, and combination of low-cost jobs and high-cost jobs.

I. Low-Cost Jobs:

Average : Low-Cost-Time-5										
	Budget [15, 5000]					Budget [30, 10,000]				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	35. 63	34. 33	33. 57	45. 91	37. 91	39. 23	38. 41	37. 17	50. 00	36. 43
A.A.F.T	39. 72	38. 41	37. 66	50. 00	42. 00	39. 23	38. 41	37. 17	50. 02	36. 43
A.S	4. 11	4. 28	3. 61	9. 07	4. 00	4. 60	5. 056	4. 13	9. 84	3. 88
A.A.S	4. 88	5. 06	4. 38	9. 84	4. 78	4. 60	5. 056	4. 11	9. 84	3. 88

Table 4-1. Low cost jobs with time 5 scheduling statistical results

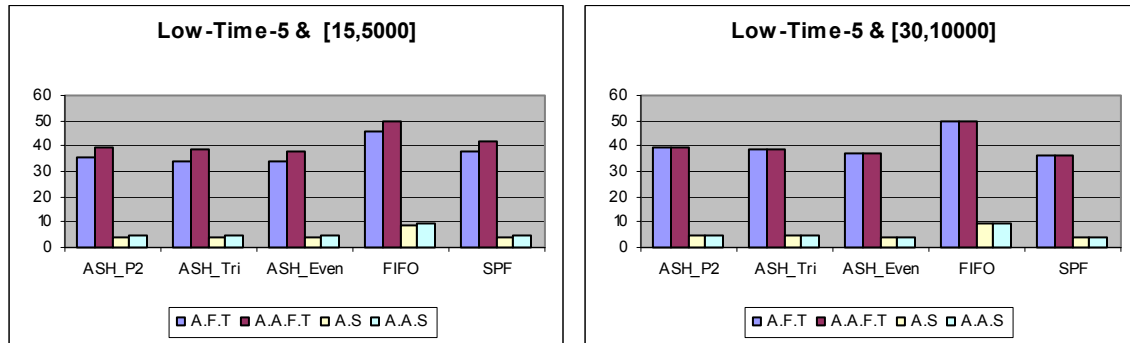


Figure 4-1. Visualization on low cost jobs with time 5 scheduling results

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

It is clear that Super-ASH is better than the FIFO and SPTF in terms of the Average Flow Time, Actual Average Flow Time, Average Stretch and Actual Average Stretch. The

Even queue model beats the Power2 and UpdownTriangle models with respect to the performance. FIFO is the worst example among the schedulers.

II. High-Cost Jobs:

Average : High-Cost-Time-5										
	Budget [15, 5000]					Budget [30, 10,000]				
	ASH_P2	ASH_Tri	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	8.84	8.72	8.72	9.65	82.21	10.31	10.35	10.25	11.77	74.47
A.A.F.T	86.53	86.41	86.41	87.34	159.90	86.58	86.61	86.51	88.04	150.73
A.S	1.26	1.24	1.24	1.85	12.63	1.60	1.66	1.60	2.39	11.40
A.A.S	20.79	20.77	20.77	21.38	32.15	19.75	19.80	19.74	20.54	29.54

Table 4-2. High cost jobs with Time 5 scheduling statistical results

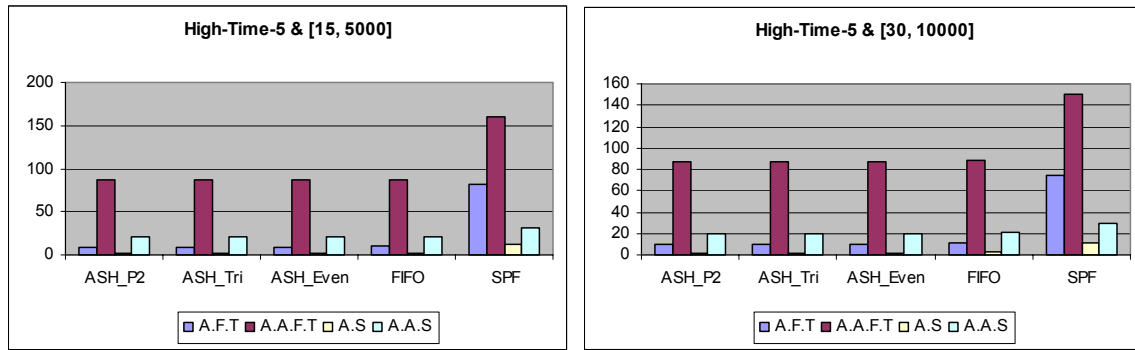


Figure 4-2. Visualization on high cost jobs with time 5 scheduling results

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

As with the low-cost jobs test result, Super-ASH is the best one in relation to scheduling performance. It always has the lowest value in Average Flow Time, Actual Average Flow Time, Average Stretch and Actual Average Stretch compared with FIFO and SPTF. One difference from the low-cost job scheduling performance result is that SPTF becomes the worst scheduler instead of FIFO. Moreover, for high cost jobs there is a big gap between AFT and AAFT, AS and AAS. Because of the effects on the budget, the difference

between release time and receive time grows with respect to low cost jobs. Additionally, the Super-ASH even queue model performed less well in low-cost job scheduling.

III. Mixed Cost Jobs:

Average : Mix-Cost-Time-5										
	Budget (15, 5000)					Budget (30, 10,000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	14. 48	14. 34	14. 11	16. 48	48. 94	16. 66	16. 64	16. 33	19. 39	44. 48
A.A.F.T	48. 18	48. 04	47. 81	50. 19	82. 64	42. 64	42. 63	42. 32	45. 38	70. 47
A.S	2. 05	2. 15	1. 98	3. 39	6. 31	2. 11	2. 34	2. 12	4. 22	5. 44
A.A.S	12. 52	12. 62	12. 46	13. 86	16. 78	9. 82	10. 04	9. 82	11. 93	13. 15

Table 4-3. Mix cost jobs with Time 5 scheduling statistical results

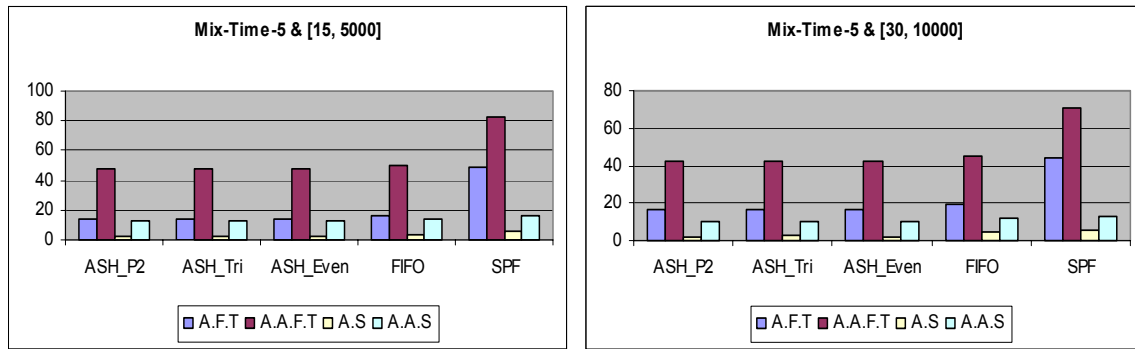


Figure 4-3. Visualization on mix cost jobs with time 5 scheduling results

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

Super-ASH is still the best scheduler with the best scheduling performance in the three test cases. SPTF is the worst scheduler due to its poor performance in scheduling the high cost jobs. The difference between Average Flow Time and Actual Average Flow Time & Average Stretch and Actual Average Stretch is smaller than the high cost jobs and greater than the low cost jobs.

4. 1. 3 Intervals and Budget Amounts

As we mentioned in the previous section, the budget does affect AAF and AAS. It works on the Actual Average Flow Time and Actual Average Stretch in two distinct ways: intervals and amounts. The interval presents how often the budget will be recharged, and the amount stands for how much the budget account was filled. If the budget account were refilled as frequently as possible and as much as possible, there would be no job delayed by the HOA because of insufficient funds. As a result, the r_i' will equal to the r_i and the Actual Average Flow Time will equal to the Average Flow Time and Actual Average Stretch will be the same as Average Stretch.

This part was tested under two conditions: same amount, different intervals; same interval, different amounts. We wrote these conditions as an ordered pair, for example (15, 5000), it means \$5,000 budget amounts added into HOA's budget account every 15 days. The same amount, different intervals cases are (30, 10000), (15, 5000), (6, 2000) and (3, 1000). The same interval, different amounts cases are (6, 2000), (6, 5000), (6, 10000) and (6, 100000) respectively. In order to show the distinct difference, we exaggerated the job generation frequency, that is, we tested the scheduling results based on certain extreme conditions, such as the roofing jobs happens twice a day. Therefore, some of the data is super high that cannot really present the ASH performance.

The four tables of data obtained from the scheduling are from the same one hundred and one jobs. And these jobs are from two types of job: roofing and electricity. As we said before, the two types of jobs covered almost all the possibilities regarding the generation

of jobs. As illustrated in the following table, we set the same amounts as \$10,000 and its intervals are 30, 15, 6 and 3. The same interval is 6 and its amounts are \$2,000, \$5,000, \$10,000 and \$100,000. Because of the space limitation, the table under one condition is split into two parts, and they are marked as “1” and “2”.

High Cost - Same Amounts Different Intervals -1										
	Budget (30, 10000)					Budget (15, 5000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	13. 86	13. 78	13. 78	13. 78	13. 76	11. 86	11. 81	11. 81	11. 81	11. 81
A.A.F.T	423. 07	422. 98	422. 98	422. 98	422. 97	356. 84	356. 79	356. 79	356. 79	356. 79
A.S	1. 57	1. 55	1. 55	1. 55	1. 55	1. 34	1. 33	1. 33	1. 33	1. 33
A.A.S	58. 42	58. 40	58. 40	58. 40	58. 40	49. 19	49. 18	49. 18	49. 18	49. 18
High Cost - Same Amounts Different Intervals – 2										
	Budget (6, 2000)					Budget (3, 1000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	12. 31	12. 05	12. 05	12. 05	12. 05	19. 62	19. 34	19. 34	19. 34	19. 34
A.A.F.T	363. 59	363. 33	363. 33	363. 33	363. 33	377. 67	377. 40	377. 40	377. 40	377. 40
A.S	1. 41	1. 36	1. 36	1. 36	1. 36	2. 17	2. 13	2. 13	2. 13	2. 13
A.A.S	50. 85	50. 80	50. 80	50. 80	50. 80	52. 36	52. 32	52. 32	52. 32	52. 32

Table 4-4. Same amounts and different intervals for high cost jobs

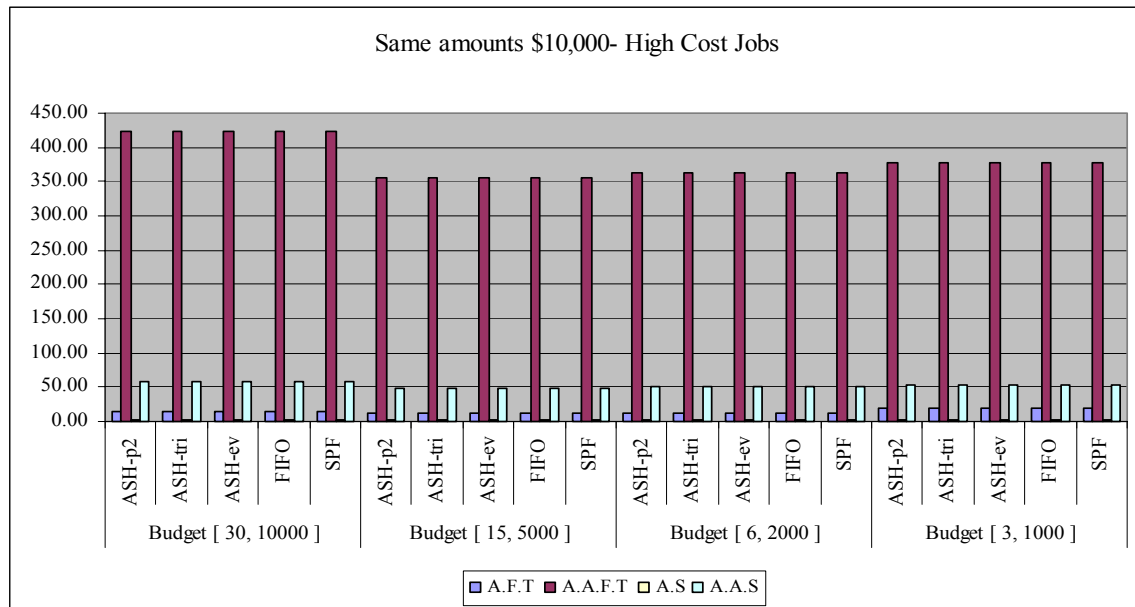


Figure 4-4-1. Same amounts \$10,000 different intervals for high cost jobs

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

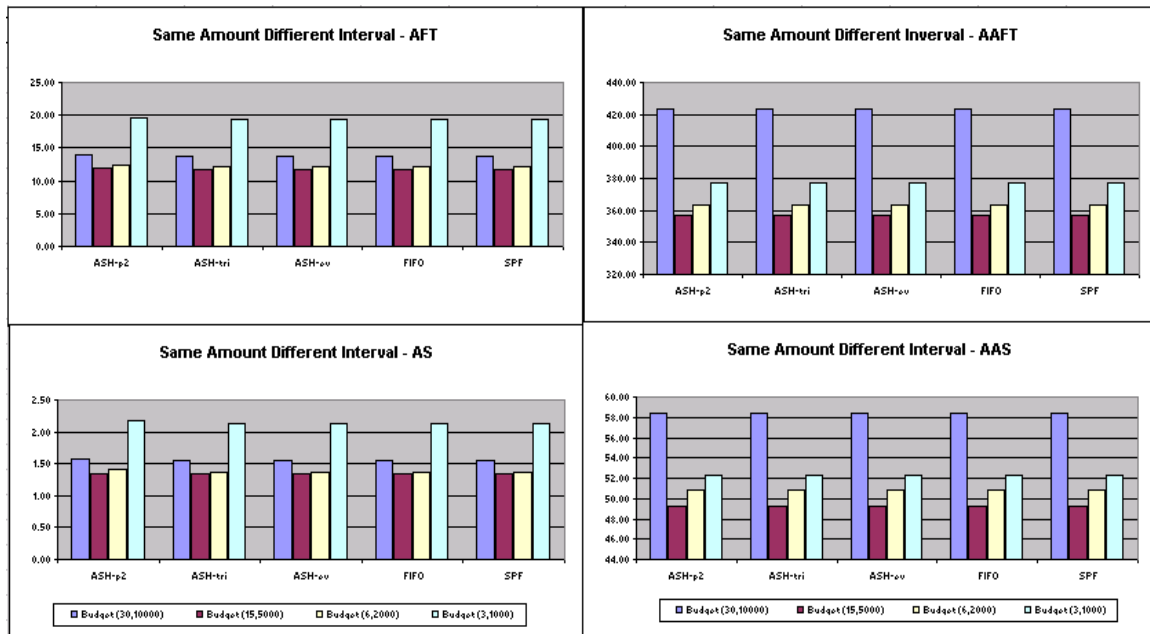


Figure 4-4-2. Same amounts different intervals for high cost jobs - AFT, AAFT, AS and AAS

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

Although the amounts are the same \$10,000, we can get different scheduling performance by setting different intervals. Budget (15, 5000) is the best setting over the others. The appropriate deposit frequency and amount is the best choice for high cost jobs.

Low Cost - Same Amounts Different Intervals -1										
	Budget (30, 10000)					Budget (15, 5000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	13. 51	13. 11	13. 11	13. 11	13. 07	24. 02	23. 60	23. 60	23. 60	23. 56
A.A.F.T	306. 49	306. 09	306. 09	306. 09	306. 04	172. 67	172. 24	172. 24	172. 24	172. 20
A.S	1. 63	1. 56	1. 56	1. 56	1. 55	3. 17	3. 09	3. 09	3. 09	3. 08
A.A.S	85. 02	84. 95	84. 95	84. 95	84. 94	43. 21	43. 13	43. 13	43. 13	43. 12
Low Cost - Same Amounts Different Intervals - 2										
	Budget (6, 2000)					Budget (3, 1000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	62. 20	60. 53	60. 47	60. 47	60. 20	67. 16	64. 02	63. 91	63. 91	63. 58
A.A.F.T	153. 33	151. 67	151. 60	151. 60	151. 33	140. 96	137. 82	137. 71	137. 71	137. 38
A.S	6. 93	6. 62	6. 61	6. 61	6. 53	7. 08	6. 48	6. 46	6. 46	6. 39
A.A.S	30. 91	30. 59	30. 58	30. 58	30. 50	26. 10	25. 50	25. 49	25. 49	25. 41

Table 4-5. Same amounts and different intervals for low cost jobs

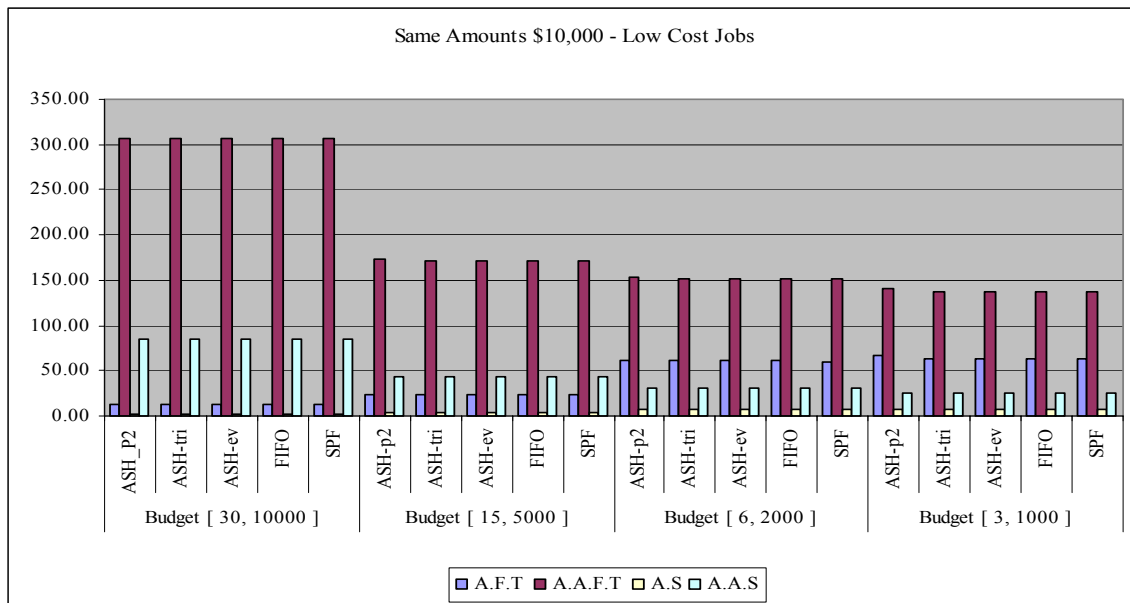


Figure 4-5-1. Same amounts \$10,000 different intervals for low cost jobs
Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

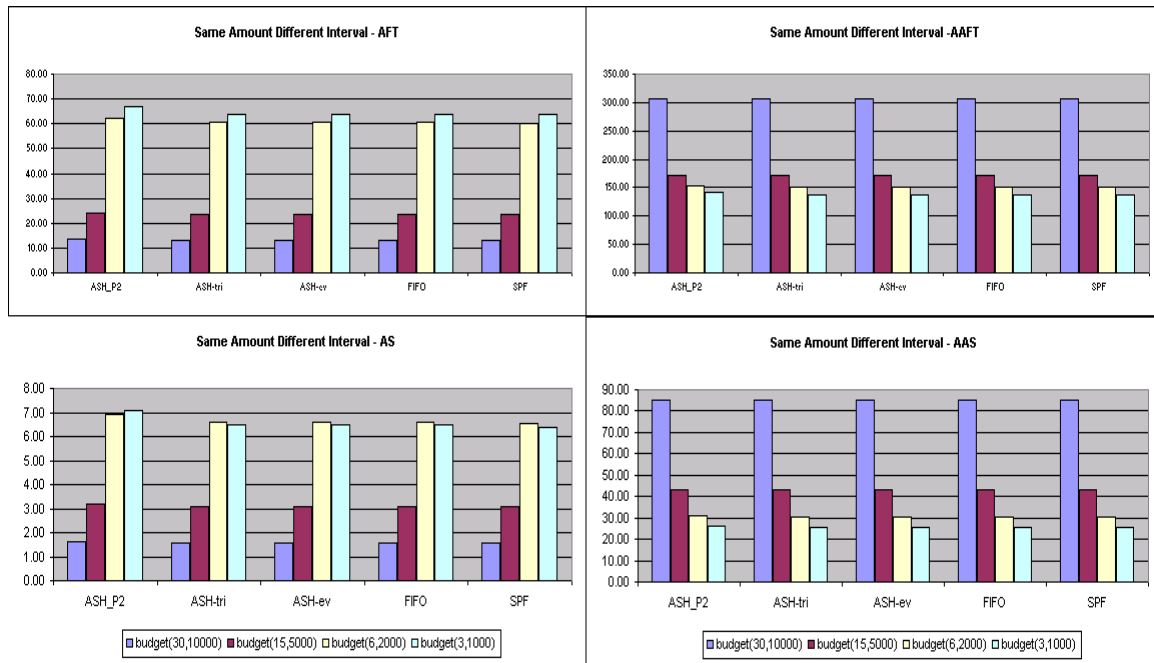


Figure 4-5-2. Same amounts \$10,000 different intervals for low cost jobs with AFT, AAFT, AS and AAS

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

Compared with the high cost job scheduling, the low cost job scheduling is more sensitive to the changes in the deposit frequency and amount. Adding to the budget more frequently will result in better scheduling performance. With more frequently added funds, the difference between AFT and AAFT, AS and AAS gets smaller.

High Cost - Same Intervals Different Amounts -1										
	Budget (6, 2000)					Budget (6, 5000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	14. 21	13. 98	13. 98	13. 98	13. 98	93. 97	93. 07	93. 02	93. 02	92. 72
A.A.F.T	368. 79	368. 57	368. 57	368. 57	368. 57	227. 29	226. 40	226. 34	226. 34	226. 05
A.S	1. 59	1. 55	1. 55	1. 55	1. 55	9. 42	9. 25	9. 25	9. 25	9. 19
A.A.S	51. 48	51. 44	51. 44	51. 44	51. 44	27. 67	27. 50	27. 50	27. 50	27. 44
High Cost - Same Intervals Different Amounts -2										
	Budget (6, 10000)					Budget (6, 100000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	145. 48	139. 67	139. 34	139. 34	138. 33	207. 67	199. 41	198. 28	198. 28	196. 66
A.A.F.T	214. 50	208. 69	208. 36	208. 36	207. 34	207. 67	199. 41	198. 28	198. 28	196. 66
A.S	14. 48	13. 38	13. 28	13. 28	13. 07	21. 73	20. 18	19. 81	19. 81	19. 46
A.A.S	24. 05	22. 95	22. 85	22. 85	22. 64	21. 73	20. 18	19. 81	19. 81	19. 46

Table 4-6. Same intervals and different amounts for high cost jobs

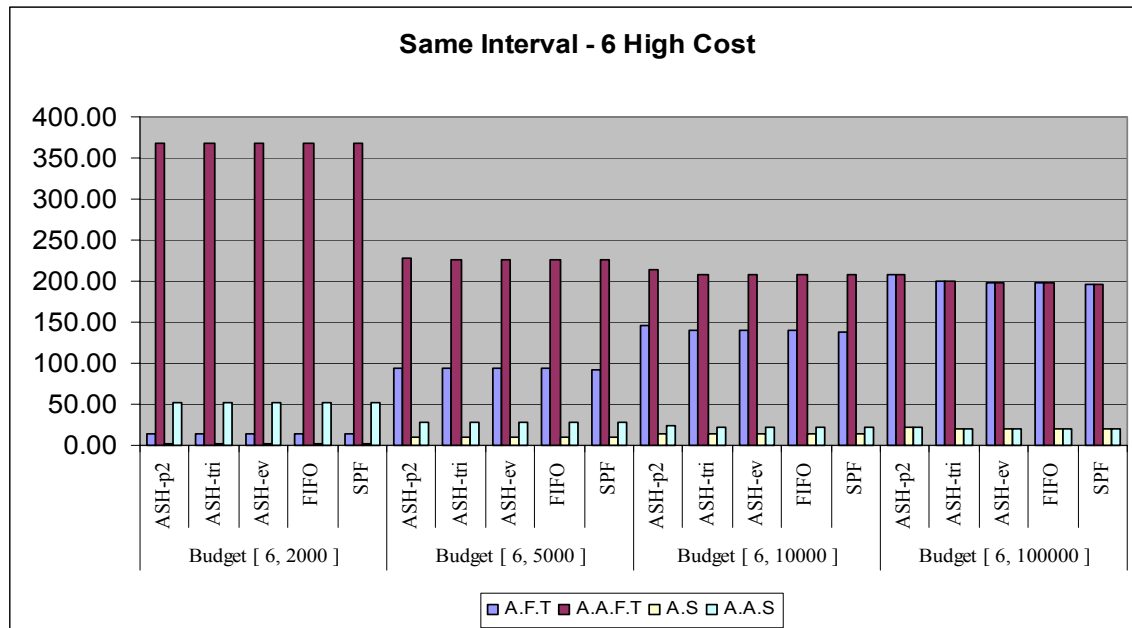


Figure 4-6-1. Same interval 6 different amounts for high cost jobs

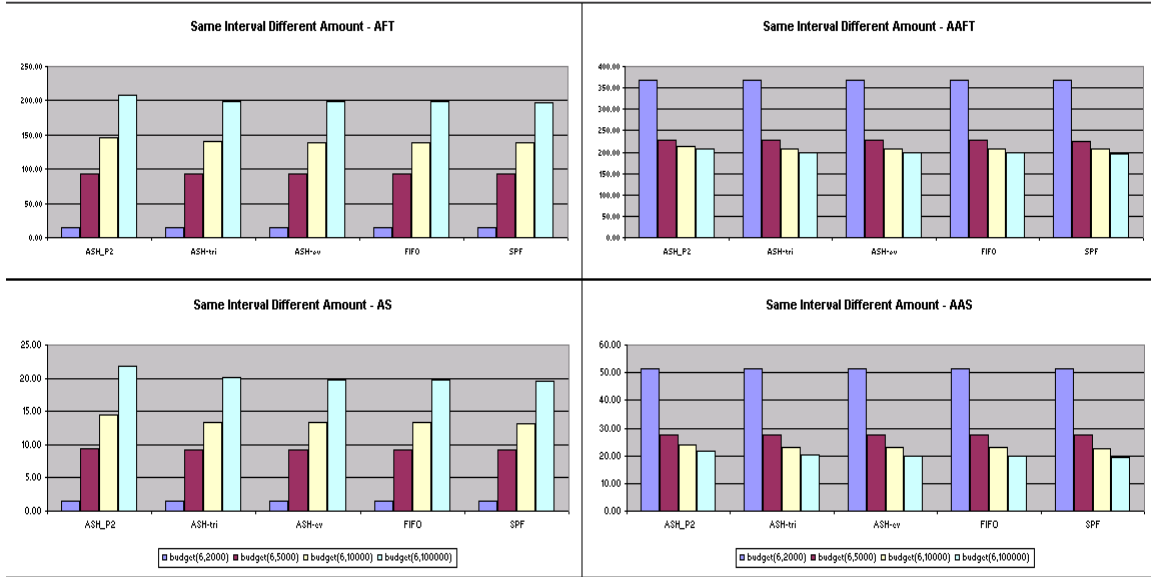


Figure 4-6-2. Same interval 6 different amounts for high cost jobs with AAFT, AFT, AS and AAS

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

Obvious performance improvement appears from the budget (6, 2000) to the budget (6, 5000). Slight changes are seen in the budgets (6, 10000), (6, 1000000). Both AFT and AS increase as the amount increases, and AFT equals to AAFT, AS equals to AAS when the amount in the budget is enough to pay off all incoming jobs. In other words, the job release times are equal to their receive times. Moreover, the AFT will increase with the increment in the amount because more jobs get into the contractor's ready queue will generate more competition and longer waiting time in the ready queue.

Unlike the previous test cases, Super-ASH cannot beat the other scheduling algorithms in this extreme case. However, so far, it still has a good overall average performance.

Low Cost - Same Intervals Different Amounts -1										
	Budget (6, 2000)					Budget (6, 5000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	67.24	64.64	64.58	64.58	64.29	80.76	77.82	77.69	77.69	77.20
A.A.F.T	158.84	156.24	156.18	156.18	155.89	127.96	125.02	124.89	124.89	124.40
A.S	7.37	6.87	6.86	6.86	6.79	8.76	8.20	8.18	8.18	8.07
A.A.S	31.52	31.02	31.01	31.01	30.94	21.70	21.14	21.12	21.12	21.01
Low Cost - Same Intervals Different Amounts -2										
	Budget (6, 10000)					Budget (6, 100000)				
	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF	ASH_P2	ASH_Tr	ASH_Ev	FIFO	SPF
A.F.T	84.40	82.02	81.93	81.93	81.38	114.53	107.38	106.38	106.38	105.64
A.A.F.T	127.69	125.31	125.22	125.22	124.67	114.53	107.38	106.38	106.38	105.64
A.S	8.83	8.37	8.35	8.35	8.24	12.44	11.07	10.71	10.71	10.54
A.A.S	21.75	21.28	21.27	21.27	21.15	12.44	11.07	10.71	10.71	10.54

Table 4-7. Same intervals and different amounts for low cost jobs

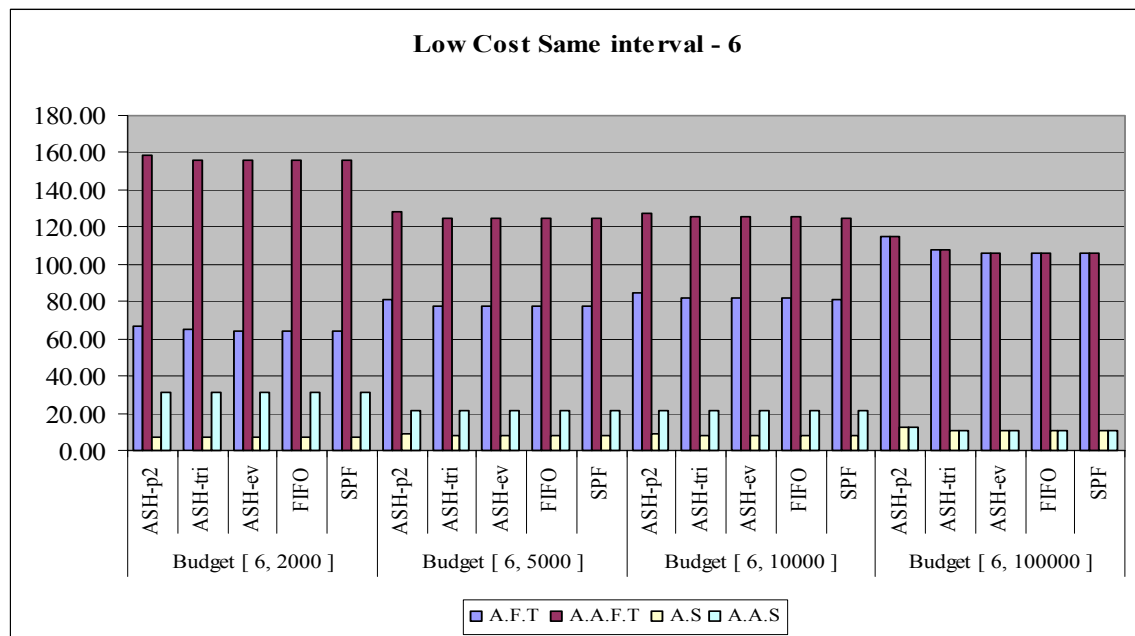


Figure 4-7-1. Same interval 6 different amounts for low cost jobs

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

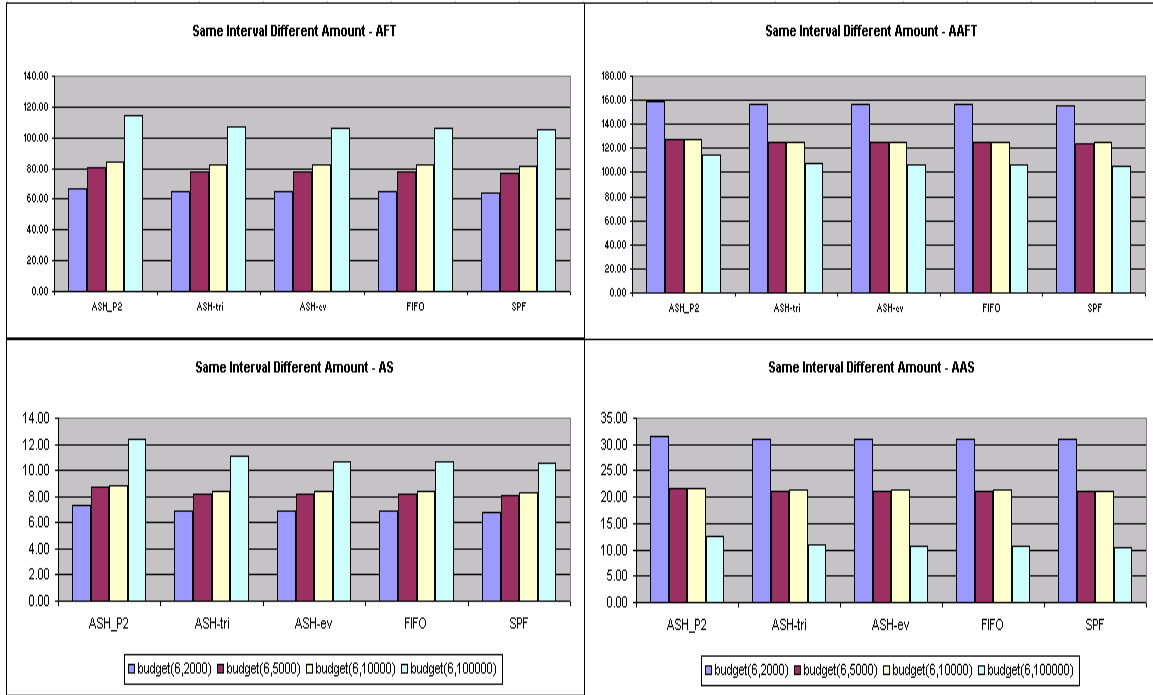


Figure 4-7-2. Same interval 6 different amounts for low cost jobs with AAFT, AFT, AS and AAS

Note: The number on Y-axis stands for processing time units per job if it represents AFT or AAFT. Otherwise, the number on Y-axis stands for the ratio if it presents AS or AAS.

In case of low cost jobs, an increased budget amount does not improve the scheduling performance. But, it still works to decrease the difference between actual average flow time/stretch and average flow time/stretch. The small interval made Super-ASH lose its outstanding performance among those scheduling algorithms.

Another obvious fact is that the performance will be improved by increasing the initiation amounts in HOAs account. So, if the HOAs can ask the homeowners to prepay the homeowner association fees, it can also help the ASH achieve better AAFT and AAS.

4. 1. 4 Different Scheduling Queue Models

Power2, UpdownTriangle and Even are the three models used during the whole process of testing. They have been tested under all of the above conditions, such as different number of jobs, different budget intervals and amounts, different type of jobs and so on. The differences between the models' shapes are demonstrated by their different characteristics. The most important reason why their performances differ is the queue level classification. Each queue model has its own different level definitions, such as the Power2 queue model, its 1st level is defined as [1,2), and 2nd level is defined as [2,4) and so on. Different levels have different priorities, the lowest numbered level with the highest priority and the highest numbered level with the lowest priority. Every time the scheduler goes through the ready-to-process queue it prioritizes jobs from the lowest numbered level to the highest numbered level. If there are always jobs in the lowest numbered level, then the jobs in the higher numbered level will starve. Therefore, a design based on the width of each level and the shape of the queue will make the scheduler behave differently during the processing procedure.

The detailed sample test results are shown in the above Tables and Figures 4-1, 4-2 and 4-3. They are divided into three parts: low-cost jobs, high-cost jobs and the mixed-cost jobs.

4. 1. 5 Yearly Performance Comparison

To evaluate whether an algorithm is applicable or not depends on not only its current performance (performance in a short time period) but its long term overall performance

and performance trend changes. If this trend is exponential, it means the jobs in the waiting queue can never be finished. It is surely unfavorable one. If the performance trend sticks around certain expected value, it is favorable performance trend. Our tests covered a span of one hundred years on different kinds of jobs under different conditions.

As we have said before, job scheduler performance is closely related to the incoming jobs. To get the realistic yearly performance test results, we developed a simulator to randomly generate the repair jobs at a given frequency. Different *aveRxInterval* and *IntervalDelta* can change the frequency. The code where this is handled is below:

```
...    type = admin. getJobType(name);
        if(type != null)
        {
            aveProcTime = (type. getMinProcTime() + type.
getMaxProcTime()) / 2;
            aveRxInterval = aveProcTime * 5 / 4;
            IntervalDelta = aveRxInterval / 2;
            time = randomCount. nextInt(3) + 1;
            while(time <= term)
            {
                admin. addJob(time, type);
                time += aveRxInterval
                    + randomCount. nextInt(IntervalDelta * 2) + 1
                    - IntervalDelta;
            }
        }
        else
            System. out. println("\r\n" + name + "job type is not found!"); ....
```

The yearly performance test is performed over 4594 jobs that include 3843 electricity jobs and 751 roofing jobs over one hundred years. In addition, all the scheduling is under the budget with interval 15 and an amount of \$5,000.

Yearly Performance Test on Super-ASH Power 2												
<i>Elec</i>	1st	2nd	3rd	4th	5th	6th	7th	...	97th	98th	99th	100th
A.F.T	10. 82	9. 17	10. 15	8. 36	10. 19	7. 16	8. 58	...	9. 78	7. 23	10. 27	9. 32
A.A.F.T	11. 13	9. 17	10. 15	8. 36	10. 19	7. 16	8. 58	...	9. 78	7. 23	10. 27	9. 32
A.S	1. 81	1. 61	1. 89	1. 55	1. 66	1. 30	1. 23	...	1. 33	1. 23	1. 67	1. 35
A.A.S	1. 96	1. 61	1. 89	1. 55	1. 66	1. 30	1. 23	...	1. 33	1. 23	1. 67	1. 35

Table 4-8 Yearly performance test on high competition jobs

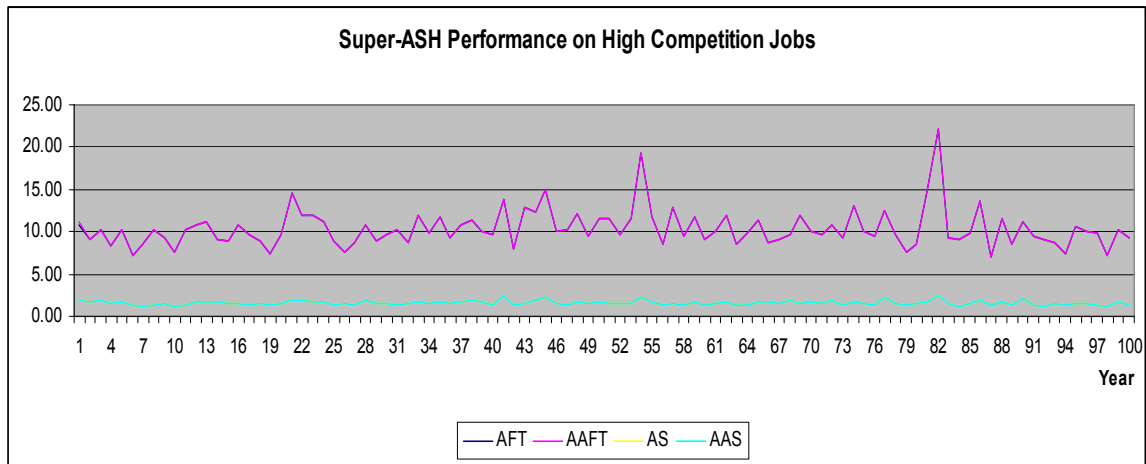


Figure 4-8 Yearly performance test on high competition Jobs

The performance varies with changes in incoming jobs as shown above. This demonstrates an unpredictable performance change trend. Moreover, the budget (15, 5,000) can satisfy the low cost jobs; the budget will not delay the low cost jobs, so it avoids big gaps between the actual average flow time/stretch and the average flow time/stretch.

Yearly Performance Test on Super-ASH Power 2												
<i>Roof</i>	1st	2nd	3rd	4th	5th	6th	7th	...	97th	98th	99th	100th
A.F.T	9. 00	8. 17	8. 71	7. 88	7. 75	9. 11	7. 67	...	9. 11	9. 00	9. 14	10. 38
A.A.F.T	10. 88	8. 17	8. 71	7. 88	7. 75	9. 11	7. 67	...	9. 11	9. 00	9. 14	10. 38
A.S	1. 00	1. 00	1. 00	1. 00	1. 00	1. 00	1. 00	...	1. 00	1. 00	1. 00	1. 00
A.A.S	1. 17	1. 00	1. 00	1. 00	1. 00	1. 00	1. 00	...	1. 00	1. 00	1. 00	1. 00

Table 4-9 Yearly performance test on low competition jobs

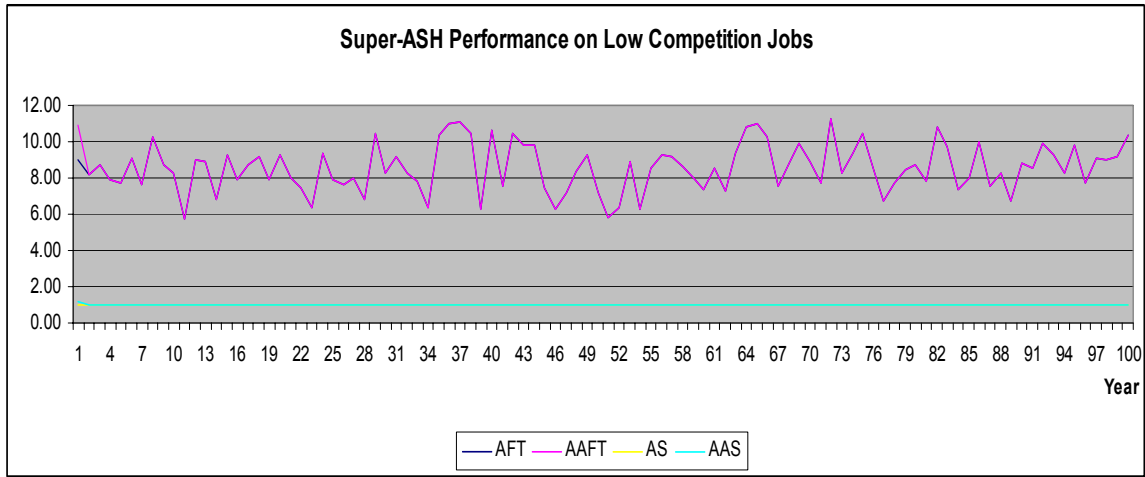


Figure 4-9 Yearly Performance test on low competition jobs

However, the budget (15, 5,000) is insufficient to pay off all high cost jobs, and many of the jobs cannot be completed in one year and are thus delayed until the next year. Consequently, the scheduling burden grows heavier as more high cost jobs are postponed. Fortunately, HOAs usually have enough funds to feed those high cost jobs, and as a matter a fact, the high cost jobs do not really high appearance frequency so that HOAs cannot afford the expense. So, the pressure on high cost jobs scheduling is not serious, as Figures 4-9 shows.

4. 2 Test Results Analysis

As shown in Section 4. 1. 2, Super-ASH always stands out from the classical scheduling algorithms FIFO and SPTF no matter which queue model is chosen. With an increase in the job numbers, all the performance variables values increase especially the AAF and

AAS . That is, the gaps between AF and AAF, AS and AAS increase in size. Some reasons for this are:

- An insufficient budget cannot afford all of the jobs, and some of the jobs release times are delayed;
- The jobs' costs are so high that the budget cannot take over all of them at the same time;
- The emergency jobs (priority =1) skew the budgets and keep all other jobs waiting until they can proceed.

As demonstrated in the Section 4. 2. 2, the AAF and AAS are highly affected by budget intervals and amount. Nevertheless, the budget does not affect AF and AS that much compared with actual performance variables. If we can set the budget interval small enough and budget amount high enough, the AAF, AAS will be equal to AF and AS respectively. As a result, the overall performance will be the same as the individual contractor schedulers.

The yearly performance test results proved that Super-ASH has a favorite change trend on AAF, AAS, AF and AS with respect to the yearly comparisons. Moreover, Super-ASH can beat the original scheduling algorithms FIFO and SPTF under various tests including the long term one. Thus, this is a worthwhile scheduling algorithm in the HOAs' domain.

5. CONCLUSION and FUTURE WORK

First of all, we can see that scheduling performance is highly job dependent. Even the Super-ASH algorithm with its configuration options set correctly cannot be guaranteed to perform well on all job sequences. As far as is known, from our studies there is no one set of rigid configuration data that can achieve the best performance under arbitrary conditions. However, this does not mean there is no way to make a set of smart configurations to try to optimize performance. Experience and knowledge of the received or the coming jobs will greatly benefit the scheduling performance. With accumulated experience and knowledge of the repair jobs, the job type table will be more accurate and applicable. If we set the costs of the jobs more-than their actual costs, then the budget will not be able to release as many as possible jobs to the contractors as early as possible. If we set the costs of the jobs less-than their actual costs, and then the budget cannot afford those jobs we released. Both wrong estimates of the job cost will cause unfavorable scheduling results. On the other hand, we can dramatically improve the scheduling performance by accurately estimate repair jobs' costs.

Also, experience and knowledge of those repair jobs can help the homeowner associations to make a good classification of the jobs. So far, there are only two categories to distinguish the jobs, emergency ones and normal ones. If they can apply their experience and knowledge to better classify those jobs in terms of their actual priorities, they can get more elaborate and more efficient scheduling results.

Therefore, the most effective way to improve Super-ASH will depend on this heuristic method applied during the scheduling process. Applying the heuristic strategy can help an HOA to accumulate the knowledge about the processed and processing jobs; as a result, it makes the future jobs more predictable. This is also the way to help HOAs to build up an applicable and precise job type table. Heuristic queue model generation can be another way to improve Super-ASH. It can help HOAs to pick the appropriate scheduling queue model depending on processing times without any experience from the HOAs as a requirement.

6. ASH User's Guide

6.1 Creating Job Type Table

The Job Type Table is maintained and updated by the Homeowner Associations, it includes the job type name, job ID, job maximum cost, job minimum cost, job maximum process time and job minimum process time. There are two ways to create the job type table; either by clicking the “Add” button in “Type” category to add types of job one by one, or by just clicking the “Gen” button in “Type” category to generate the job type table for you. The screen shots below illustrate this:

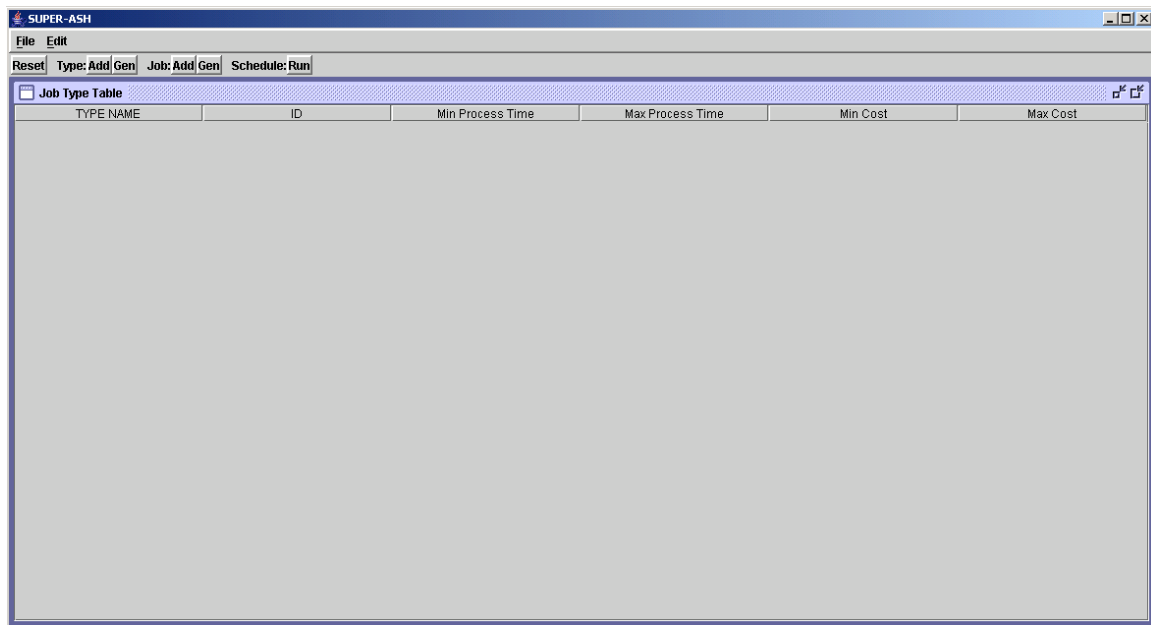
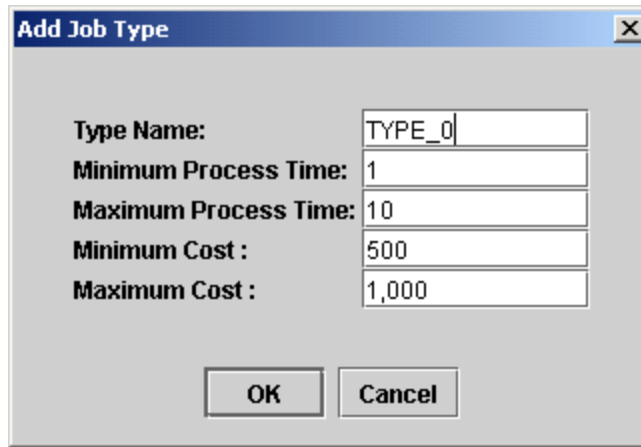


Figure 6-1-1. Job Type Table without adding any job types

To create a Job Type Table by adding the job type one by one, you can click the “Add” button in the “Type” category. The screen shot shows the edit window once you click the “Add” button:

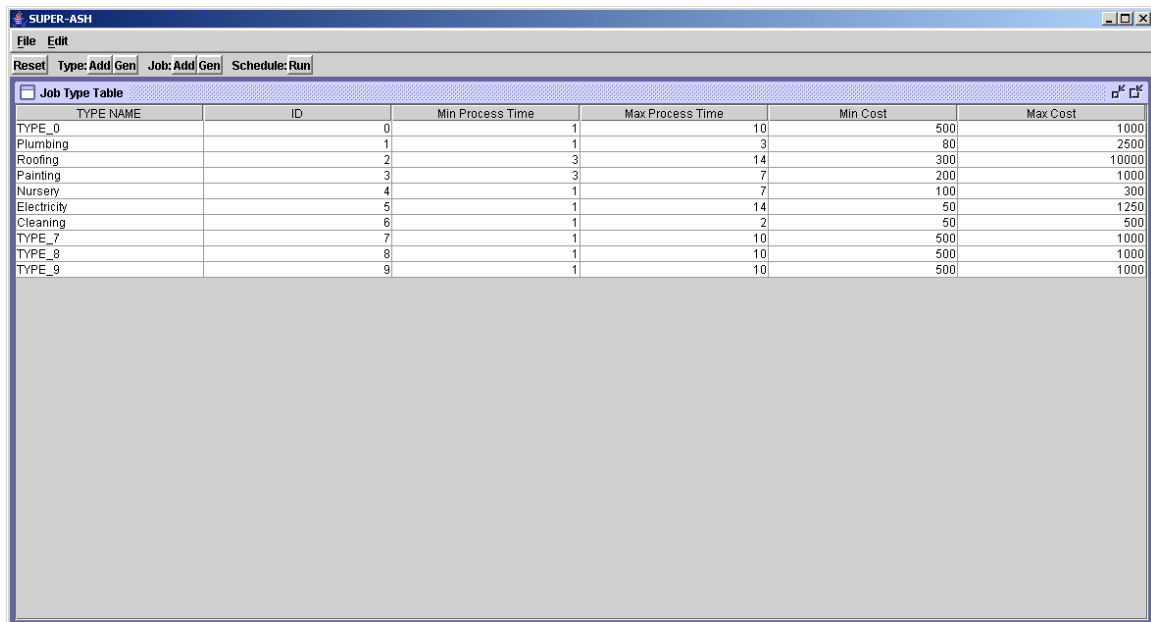


The 'Add Job Type' dialog box contains the following fields and buttons:

- Type Name: TYPE_0
- Minimum Process Time: 1
- Maximum Process Time: 10
- Minimum Cost : 500
- Maximum Cost : 1,000
- Buttons: OK, Cancel

Figure 6-1-2. Add job type one by one once click the “Add” button in “Type” category

In this window, you can edit the job “Type name”, “Minimum Process Time”, “Maximum Process Time”, “Minimum Cost” and “ Maximum Cost”. After you finish the editing, you can click “OK” button to proceed; or you can click “Cancel” button to cancel the adding action.



The screenshot shows the 'Job Type Table' window with the following data:

TYPE NAME	ID	Min Process Time	Max Process Time	Min Cost	Max Cost
TYPE_0	0	1	10	500	1000
Plumbing	1	1	3	80	2500
Roofing	2	3	14	300	10000
Painting	3	3	7	200	1000
Nursery	4	1	7	100	300
Electricity	5	1	14	50	1250
Cleaning	6	1	2	50	500
TYPE_7	7	1	10	500	1000
TYPE_8	8	1	10	500	1000
TYPE_9	9	1	10	500	1000

Figure 6-1-3. The created Job Type Table after one by one “Add” and default generator

6.2 Creating Job List

After creating the Job Type Table, we need to use the “Job Type Table” as the reference to generate the “Job List”. There are also two buttons in the “Job” field, that is, “Add” and “Gen”. It is capable of adding jobs one by one with “Add” steps, or just clicks the “Gen” to generate a job list for you. The following figures will show you the detailed steps:

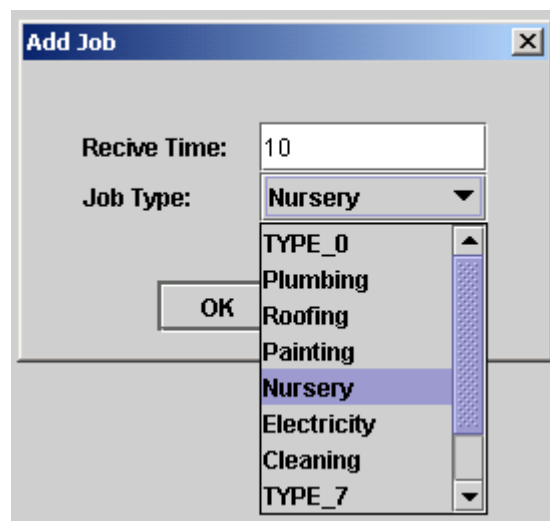


Figure 6-2-1. Adding jobs one by one

In the adding job step, you only need to edit the job “Receive Time” and pick the job type from the scrolling list. Click the “OK” button to confirm the adding behavior. In this example, one “Nursery” job with receive time “10” will be added into the job list table, shown as below:

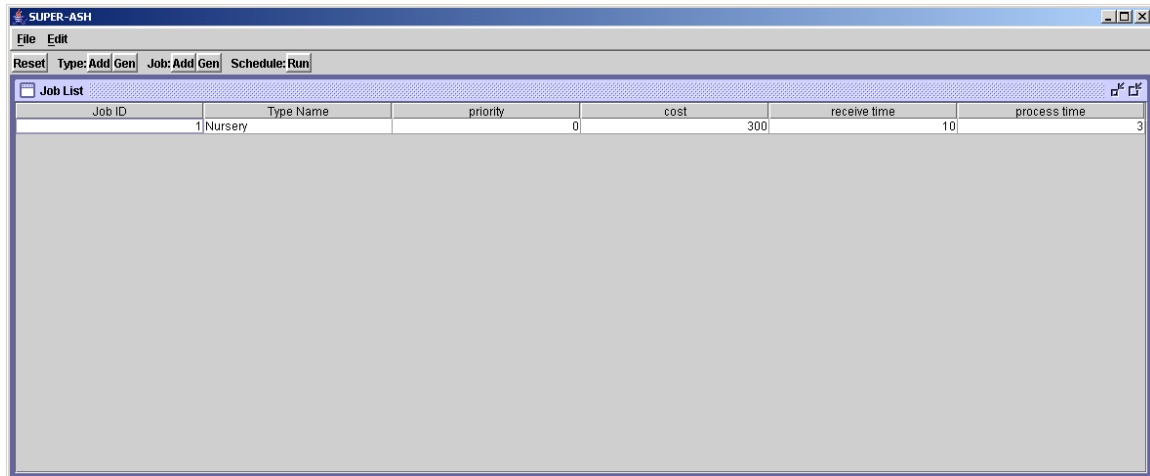


Figure 6-2-2. One job added into the Job List

Or you can use the job generator to help you generate a set of jobs by clicking the “Gen” button in the “Job” field; the pop-up window will show as the following:

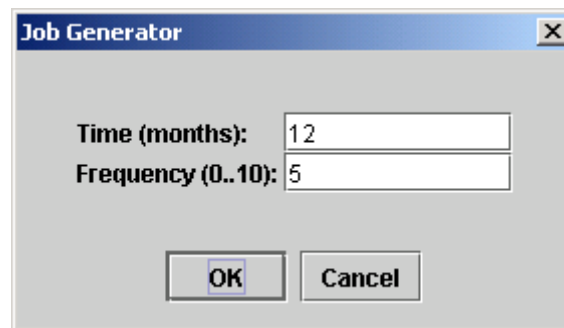


Figure 6-2-3. Using job generator to generate jobs

The “Time (months)” means the time period you would like to create jobs. The “Frequency (0..10)” stands for the frequency of the job appearance. The bigger number means higher density of jobs will be generated. If it is “5”, it means the new job will generate once the previous job is done. In this example, it asks the job generator to create jobs in twelve months period of time. After setting the job generator two parameters, we will get the following job list table:

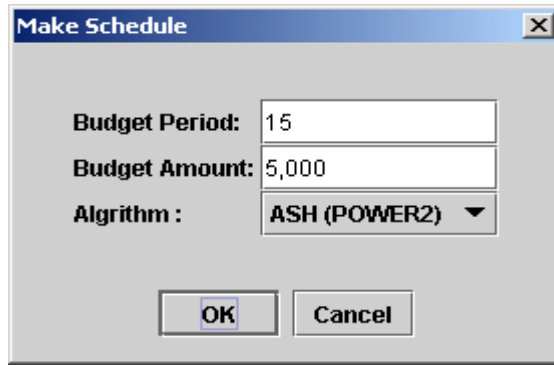
Job ID	Type Name	priority	cost	receive time	process time
1	Nursery	0	300	10	3
2	TYPE_0	0	905	3	8
3	TYPE_0	0	792	15	8
4	TYPE_0	0	879	28	1
5	TYPE_0	0	939	42	6
6	TYPE_0	0	551	56	10
7	TYPE_0	0	533	71	7
8	TYPE_0	0	553	84	2
9	TYPE_0	0	557	98	8
10	TYPE_0	0	523	110	10
11	TYPE_0	0	747	124	4
12	TYPE_0	0	598	140	9
13	TYPE_0	0	527	156	6
14	TYPE_0	0	941	170	3
15	TYPE_0	1	658	185	3
16	TYPE_0	0	730	198	3
17	TYPE_0	0	531	212	5
18	TYPE_0	0	916	227	4
19	TYPE_0	0	513	240	9
20	TYPE_0	0	881	253	6
21	TYPE_0	0	947	267	10
22	TYPE_0	0	882	278	3
23	TYPE_0	1	823	290	8
24	TYPE_0	0	504	301	6
25	TYPE_0	0	927	315	1
26	TYPE_0	0	974	327	6
27	TYPE_0	1	627	342	5
28	TYPE_0	0	519	358	7
29	Plumbing	1	2220	1	1
30	Plumbing	0	1332	8	3
31	Plumbing	1	1499	15	3
32	Plumbing	1	1758	20	1
33	Plumbing	1	949	26	1
34	Plumbing	0	880	31	1
35	Plumbing	1	511	37	1
36	Plumbing	0	1378	44	3
37	Plumbing	0	1001	51	2
38	Plumbing	0	461	57	1
39	Plumbing	0	752	63	1
40	Plumbing	0	1769	68	2
41	Plumbing	0	467	73	3
42	Plumbing	0	939	79	1
43	Plumbing	0	420	84	3
44	Plumbing	0	1339	90	1
45	Plumbing	0	880	95	1
46	Plumbing	0	1946	100	1

Figure 6-2-4. Job List table shows all the added and generated jobs

Once you get your job list table, you can still go back to reset your Job Type Table or add some specific jobs as needed.

6.3 Choosing Scheduling Method

After setting up the Job Type Table and creating the Job List, it is time to choose the scheduling method and edit the budget interval and amount by clicking the “Run” button in the “Schedule” field.

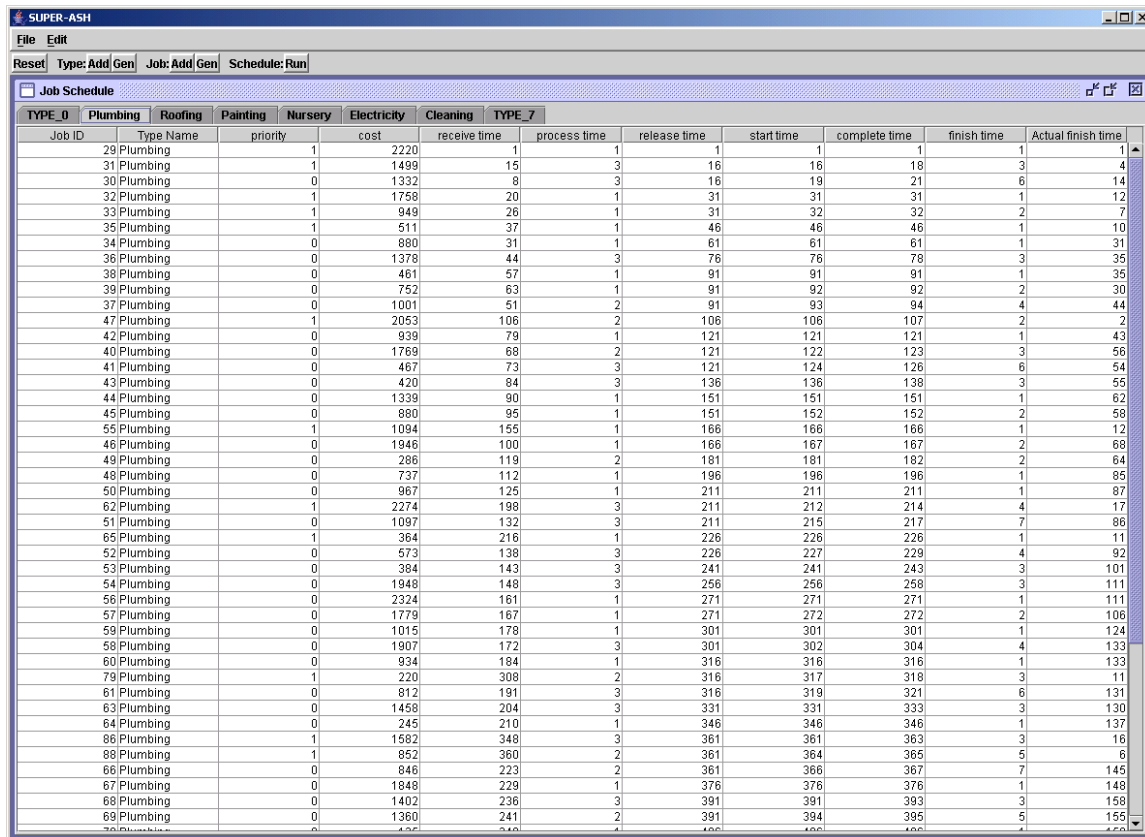


The 'Make Schedule' dialog box is shown with the following settings:

- Budget Period:** 15
- Budget Amount:** 5,000
- Algorithm:** ASH (POWER2) (selected from a dropdown menu)
- Buttons:** OK and Cancel

Figure 6-3-1. Make scheduling window for budgets and scheduling method setting

In this example, we set \$5,000 deposit into the HOA every 15 days, and choose ASH-Power2 as the current scheduling method. After everything is set, the “OK” button will process the scheduling as our configurations.



The 'SUPER-ASH' Job Schedule window displays a detailed schedule for various jobs. The window includes a menu bar (File, Edit) and a toolbar with buttons for Reset, Type: Add, Gen, Job: Add, Gen, and Schedule: Run. The main area shows a table with columns for Job ID, Type Name, priority, cost, receive time, process time, release time, start time, complete time, finish time, and Actual finish time. The table lists 69 jobs, all of which are Plumbing tasks, with their respective costs and scheduled times.

Job ID	Type Name	priority	cost	receive time	process time	release time	start time	complete time	finish time	Actual finish time
29	Plumbing	1	2220	1	1	1	1	1	1	1
31	Plumbing	1	1499	15	3	16	16	18	3	4
30	Plumbing	0	1332	8	3	16	19	21	6	14
32	Plumbing	1	1758	20	1	31	31	31	1	12
33	Plumbing	1	949	26	1	31	32	32	2	7
35	Plumbing	1	511	37	1	46	46	46	1	10
34	Plumbing	0	880	31	1	61	61	61	1	31
36	Plumbing	0	1378	44	3	76	76	78	3	35
38	Plumbing	0	461	57	1	91	91	91	1	35
39	Plumbing	0	752	63	1	91	92	92	2	30
37	Plumbing	0	1001	51	2	91	93	94	4	44
47	Plumbing	1	2053	106	2	106	106	107	2	2
42	Plumbing	0	939	79	1	121	121	121	1	43
40	Plumbing	0	1769	88	2	121	122	123	3	56
41	Plumbing	0	467	73	3	121	124	126	6	54
43	Plumbing	0	420	84	3	136	136	138	3	55
44	Plumbing	0	1339	90	1	151	151	151	1	62
45	Plumbing	0	880	95	1	151	152	152	2	58
55	Plumbing	1	1094	155	1	166	166	166	1	12
46	Plumbing	0	1946	100	1	166	167	167	2	68
49	Plumbing	0	286	119	2	181	181	182	2	64
48	Plumbing	0	737	112	1	196	196	196	1	85
50	Plumbing	0	967	125	1	211	211	211	1	87
62	Plumbing	1	2274	198	3	211	212	214	4	17
51	Plumbing	0	1097	132	3	211	215	217	7	86
65	Plumbing	1	364	216	1	226	226	226	1	11
52	Plumbing	0	573	139	3	226	227	229	4	92
53	Plumbing	0	384	143	3	241	241	243	3	101
54	Plumbing	0	1948	148	3	256	256	258	3	111
56	Plumbing	0	2324	161	1	271	271	271	1	111
57	Plumbing	0	1779	167	1	271	272	272	2	106
59	Plumbing	0	1015	178	1	301	301	301	1	124
58	Plumbing	0	1907	172	3	301	302	304	4	133
60	Plumbing	0	934	184	1	316	316	316	1	133
79	Plumbing	1	220	308	2	316	317	318	3	11
61	Plumbing	0	812	191	3	316	319	321	6	131
63	Plumbing	0	1458	204	3	331	331	333	3	130
64	Plumbing	0	245	210	1	346	346	346	1	137
86	Plumbing	1	1582	348	3	361	361	363	3	16
88	Plumbing	1	852	360	2	361	364	365	5	6
66	Plumbing	0	846	223	2	361	366	367	7	145
67	Plumbing	0	1848	229	1	376	376	376	1	148
68	Plumbing	0	1402	236	3	391	391	393	3	158
69	Plumbing	0	1360	241	2	391	394	395	5	155

Figure 6-3-2 Scheduling results based on the above configurations

If you would like to go back and reset the budget and scheduling method based on the same job list what we got from the first two steps, you can simply re-click the “Run” button and reset as you want. Or you can start from the very beginning by click the “Reset” button on the leftmost concern of Super-ASH window.

REFERENCES

[AI90] A. Allen. Probability, Statistics, and Queueing Theory with Computer Science Applications. Academic Press, inc. 1990.

[AL99] Susanne Albers, Stefano Leonardi, Online algorithm, Volume 31, Issue 3 (September 1999) Article No. 4, 1999 ISSN:0360-0300.

[ATCH] Algorithms and Theory of Computation Handbook, page 10-17, 1999 by CRC Press LLC. Appearing in the Dictionary of Computer Science, Engineering and Technology, 2000 CRC Press LLC.

[BLMP04] B. Luca, L. Stefano, M. Alberto, P. Kirk, Semi-clairvoyant scheduling. Theoretical Computer Science 324 (2004) 325 –335.

[BNR02] Allan Borodin, Morten N. Nielsen, Charles Rackoff, (Incremental) Priority Algorithms, Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms pp: 752 – 761, Year of Publication: 2002, ISBN:0-89871-513-X.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Introduction to algorithms. MIT Press. 1990.

[EJ99] E. Jeff, Scheduling in the Dark, the full version of this paper appears at [www. cs.yorku. cs/jeff/researches](http://www.cs.yorku.ca/jeff/researches), York University, Canada.

[GKLL79] R. L. Graham, A. H. G. Rinnooy Kan, E. L. Lawler, J. K. Lenstra.

Optimization and approximation in deterministic sequencing and scheduling: survey.

Annals of Discrete Mathematics, 5:287-326, 1979.

[KSW] David Karger, Massachusetts Institute of Technology, Cliff Stein, Dartmouth

College, Joel Wein, Polytechnic University. Scheduling Algorithms. Lecture notes.

[MR95] M. Rajeev, R. Prabhakar, Randomized Algorithms, Cambridge University Press,

1995.

[NG02] Nutt G. Operating Systems – A Modern Perspective 2nd Edition LAB UPDATE

Year 2002.

[RP04] Ryan Porter, Mechanism design for online real-time scheduling, pp: 61 - 70, 2004

ISBN: 1-58113-711-0.

[SRRJ97] S. Muthukrishnan, R. Rajaraman, R. Shaheen, J. Gehrke, Online scheduling to

minimize average stretch, IEEE Symp. Foundations of Computer Science, 1997, pp 433 –

442.