# Machine Scheduling Performance with Maintenance and Failure

Y. Guo[4], A. Lim[1], B. Rodrigues[2], S. Yu[3]

[1]Department of IELM, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

[2]School of Business, Singapore Management University, 50 Stamford Road, Singapore 178899

[3]School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543

[4]Department of Computer Science, Cornell University, Ithaca, New York, USA 14853

## Abstract

In manufacturing control, machine scheduling research has mostly dealt with problems either without maintenance or with deterministic maintenance when no failure can occur. This can be unrealistic in practical settings. In this work, an experimental model is developed to evaluate the effect of corrective and preventive maintenance schemes on scheduling performance in the presence machine failure where the scheduling objective is to minimize schedule duration. We show that neither scheme is clearly superior, but that the applicability of each depends on several system parameters as well as the scheduling environment itself. Further, we show that parameter values can be chosen for which preventive maintenance does better than corrective maintenance. The results provided in this study can be useful to practitioners and to system or machine administrators in manufacturing and elsewhere.

**Key Words**: machine scheduling, experiments, manufacturing control, maintenance

# 1. Introduction

In machine scheduling control, good bounds are available for the problem of minimizing schedule durations, or the *makespan*. Graham [8] provided a worst-case bound for the approximation algorithm, Longest Processing Time, and Coffman, Garey and Johnson [6] provided an improved bound using the heuristic, MULTIFIT. By combining these, Lee and Massey [13] were able to obtain an even tighter bound. These studies, however, assumed the continuous availability of machines, which may not be justified in realistic applications where machines can become unavailable due to deterministic or random reasons.

It was not until the late 1980's that research was carried out on machine scheduling with availability constraints. In a study, Lee [15] considered the problem of parallel machine scheduling with non-simultaneous available time. In another work, Lee [14] discussed various performance measures and machine environments with single unavailability. For each variant of the problem, a solution was provided using a polynomial algorithm, or it was shown that the problem is NP-hard. Turkcan [24] analyzed the availability constraints for both the deterministic and stochastic cases. Qi, Chen and Tu [21] conducted a study on scheduling the maintenance on a single-machine. The reader is referred to Turkcan [24] for a detailed literature review of machine scheduling with availability constraints. Other work on scheduling with maintenance is available, but with different scheduling environments and/or objectives. Lee and Liman [16] studied single-machine flow-time scheduling with maintenance while Kaspi and Montreuil [12] attempted to minimize the total weighted completion time in two

machines with maintenance. Schmidt [23] discussed general scheduling problems with availability constraints, taking into account different release and due dates.

These studies addressed the problem of maintenance, but in a limited way. They either considered only one deterministic maintenance (or availability) constraint or maintenance without machine failures. The results, however, are inadequate for solving real problems. In industrial systems, machines can fail due to heating or lack of lubrication, for example; in computer systems, the Internet is a typical example of system instability and breakdowns due to both hardware and software problems. In such cases, maintenance needs to be carried out, either periodically or after failure. Yet, even with maintenance, failures are not completely eliminated. Further, the overall performance, rather than the worst-case performance, is of greater relevance to the users and administrators of these systems.

In this work, we address this need and study the average or expected performance of machine scheduling with both maintenance and failures. Since maintenance as well as failure are everyday occurrences in industry, this study is particularly relevant to practitioners and systems administrators.

### 1.1 Scope and Preliminaries

We first discuss the scheduling environment, and the rules and maintenance schemes used in this study. For basic notations and definitions, see, for example, Pinedo [20]

*Scheduling environment:* This study focuses on two areas: one, on single-machine scheduling, and the other on multiple (and identical) machine scheduling. All jobs are

released at zero time. The objective is to minimize the makespan. In short, the scheduling environments studied are described as $1||C_{max}$ and $Pm||C_{max}$.

*Scheduling rules:* In view of the simple structure in the problem, we use two scheduling rules, namely Longest Processing Time first (LPT) and Service In Random Order (SIRO), both of which are list scheduling rules. LPT sorts all jobs into a list in non-increasing order; SIRO generates a list of random permutation of all jobs, and if the input jobs are already in random order, then there is no need to generate another random permutation. Each time a machine is freed, the job at the beginning of the list is assigned to the machine.

*Maintenance schemes:* We first define an "interruption" to be a stoppage of a machine either due to machine failure or to maintenance. There are two maintenance schemes in general: corrective maintenance (CM) and preventive maintenance (PM). CM executes maintenance only after each failure. It is assumed (regardless of the maintenance scheme) that the time spent on maintenance after failure is linearly proportional to the time between two failures. PM executes maintenance after a fixed period of time from the last interruption, and after failures. It is assumed that in PM the time spent on maintenance after a fixed period of time is a constant.

The objective of this work is to evaluate the performance of CM and PM with a scheduling rule LPT or SIRO in the environment $1||C_{max}$ and $Pm||C_{max}$. In the following, how the problem is modeled is discussed in Section 2, and experiment design is explained in Section 3. Following this, single-machine experiments are developed in section 4 and multiple-machine experiments in Section 5. Conclusions drawn for the experiments are provided in Section 6. In Section 7, the work is summarized.

# 2. Modeling the Problem

**2.1 System modeling** Most aspects of system modeling are straightforward except machine maintenance and failure. In CM, a Random Number Generator (RNG) is used to generate the Time Between Failures (TBF). The time between failures follows normal distribution with a specified Mean Time To Failure (MTTF) and standard deviation *std*. The philosophy behind this is that most TBF values will be in the neighborhood of a mean time, and extremely long or short TBFs are unlikely. The Maintenance Time (MT) is a linear factor of the duration between failures. In PM, the failure time is generated in the same way (see section 4.3). Each interruption is caused by the smaller of the fixed period for maintenance and TBF. The MT after fixed time is a specified constant, and the MT after failure is a linear factor of the Time Between Interruptions (TBI). Figure 1 illustrates this:

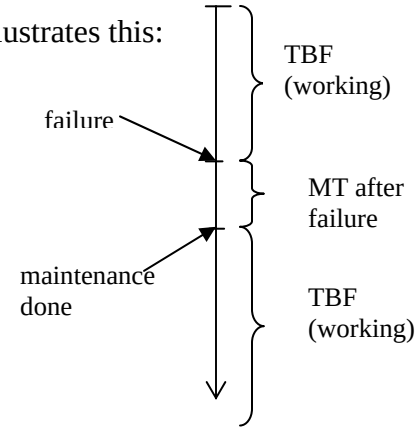


**Figure 1 (a)**

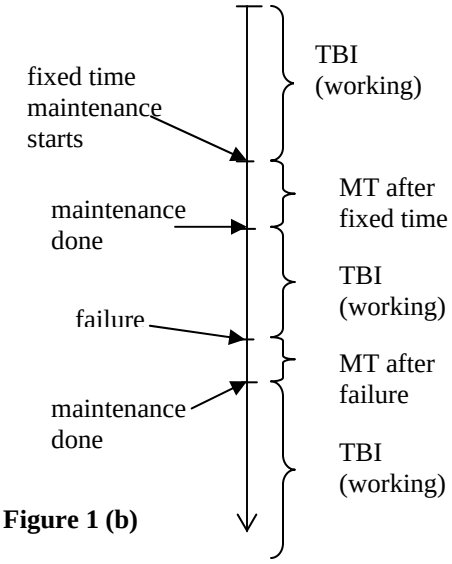


**Figure 1 (b)**

**Figure 1 (a) and (b) are sample time lines for corrective maintenance and preventive maintenance, respectively**

**2.2 Job modeling** Jobs are described by their processing times. The processing times follow an exponential distribution with a specified mean and are generated by a random number generator. All jobs are non-resumable, that is to say, if some job is interrupted before completion, whether because of machine failure or maintenance, it must be processed again from the very beginning after the machine restarts.

# 3. Experiment Design

Here, we provide the key characteristics of the experiment design. These are the performance metrics, parameters, factor and levels and lastly, experimental procedures. For related and general descriptions see [1], [2], [3], [4], [9], [10], [11], [17], [18], [19], [22].

**3.1 Performance Metrics** The performance metric used is the scheduling objective itself, the makespan ($C_{max}$), which is the completion time of the job that finishes last. A small $C_{max}$ indicates balanced load on multiple-machines and is consequently desirable. Another metric we adopt is the Total Maintenance Time (TMT). In CM, TMT only consists of the maintenance time used after failure, whereas in PM, it consists of the maintenance time used after fixed period as well as after failure. The maintenance time is actually akin to an overhead, so that a smaller TMT is preferable.

**3.2 Parameters** There are roughly three types of parameters that can affect the above performance metrics. The first kind is system and workload parameters: the number of machines *m*, the number of jobs *n*, and the mean processing time of jobs *u*. Parameters of this kind are usually not in the control of the system or machine administrator, or at least,

cannot be changed or modified at discretion. The second kind is scheduling parameters: the scheduling rule in this case – whether we choose SIRO or LPT. This parameter is at full control of the administrator, and can be changed with little effort. The last type of parameters are maintenance parameters. In CM, we have three maintenance parameters. The first is the mean time to failure (denoted by $MTTF_c$, where the "*c*" subscript indicates "corrective"), which is negatively related to failure rate. The second is the standard deviation, *std*, of the time to failures ($MTTF_c$ and *std* together determine the normal curve from which we compute time between failures). The third is the linear factor *k*, which specifies how much maintenance time is needed for each period of time between failures, i.e. MT = $k \times$ TBF. All three are usually decided by the working environment and cannot be easily changed. In PM, we also have the above-mentioned three parameters. The Mean Time To Failure in PM (denoted by $MTTF_p$) is assumed to be larger than $MTTF_c$, because more diligent maintenance is expected to result in a smaller failure rate and thus a larger $MTTF_p$. ($MTTF_p$ will be discussed again in this sub-section) The standard deviation *std* and the linear factor *k* in PM are assumed to be the same as in CM. Besides the three, we use other parameters for PM. In PM, maintenance is carried out at fixed intervals, and the time between two maintenance events is called Fixed Maintenance Period (FMP). The parameter, *a*, is a linear factor between FMP and $MTTF_c$, i.e., FMP = $a \times MTTF_c$. Fixed interval maintenance takes a fixed amount of time, and this time is called Fixed Maintenance Time (FMT). FMT is expected to be smaller than the mean of MT in CM, because the maintenance is carried out when machines are still in working condition and therefore less maintenance needs to be done. However, there is also no reason why this cannot be greater than the mean of MT. The

parameter, *b*, is a linear factor between FMT and **E**(MT), i.e., FMT = $b \times$ **E**(MT), where **E**(·) is the expectation function. We use another parameter *c*, to describe the linear relationship between $MTTF_c$ and $MTTF_p$, i.e., $MTTF_p = c \times MTTF_c$. In summary, we have five parameters in PM: *std*, *k*, *a*, *b* and *c*. The parameters *a*, *b*, and *c* can be controlled: to vary *a*, corresponds to vary the FMP, is easily done; to vary *b*, corresponding to varying FMT, can be done by selecting error-prone parts of the machine to maintain, and ignoring stable parts; to vary *c* is similar to varying *b*.

Figure 2 is shows parameters and performance metrics involved and Table 1 is a summary of all notation and parameters, their meanings, and values taken or levels (discussed next).

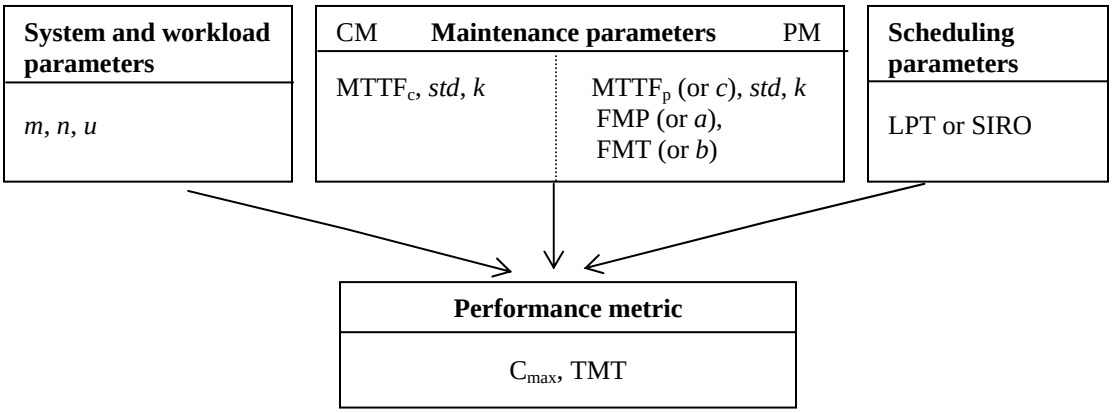


**Figure 2 The relationship between parameters and performance metrics**

| **Name** | **Symbol** | **Meaning** | **Value or levels** (if any) |
|---|---|---|---|
| total maintenance time | TMT | the sum of all maintenance time used | performance metric |
| maximum makespan | $C_{max}$ | the completion time of the last finished job | performance metric |
| corrective maintenance | CM | maintaining after failure | |
| preventive maintenance | PM | maintaining after failure and after fixed period | |

| | | | |
|---|---|---|---|
| longest processing time first | LPT | doing the longest-processing time job first | |
| service in random order | SIRO | doing the jobs randomly | |
| *m* | *m* | the number of machines | 1 ($1\|\|C_{max}$), 10 ($Pm\|\|C_{max}$) |
| *n* | *n* | the number of jobs | 10000 ($1\|\|C_{max}$)<br>100000 ($Pm\|\|C_{max}$) |
| *u* | *u* | mean processing time | 10 |
| mean time to failure corrective | $MTTF_c$ | mean of the time between failures in CM | 1000 |
| mean time to failure preventive | $MTTF_p$ | mean of the time between failures in PM | depending on *c* |
| standard deviation | *std* | standard deviation of the time between failures | 200 |
| time between failure | TBF | the amount of time between two failures | depending on $MTTF_c$ or $MTTF_p$, and *std* |
| time between interruption | TBI | the amount of time between two interruptions | depending on $MTTF_p$ and *std* |
| maintenance time | MT | maintenance time after failure | depending on *k* |
| fixed maintenance period | FMP | time between two scheduled consecutive maintenances | *constant, depending on a* |
| fixed maintenance time | FMT | maintenance time at a fixed interval | constant, depending on *b* |
| *k* | *k* | linear factor of MT from TBI or TBF | 0.01 |
| *a* | *a* | linear factor of FMP from $MTTF_c$ | 0.2, 0.4, 0.6, …, 2.8, 3.0 |
| *b* | *b* | linear factor of FMT from the mean of MT in CM | 0.2, 0.4, 0.6, …, 1.8, 2.0 |
| *c* | *c* | linear factor of $MTTF_c$ from $MTTF_p$ | 1.00, 1.25, 1.50, …, 2.75, 3.00 |

**Table 1 Summary of terminology and parameters used**

**3.3 Factors and Levels** Factors are selected parameters that are believed to have significant impact on the performance, or that are at the control of end users and need to be optimized. Levels are a set of values taken by each factor. In our case, we will select

parameters that can somehow be changed or modified by the machine administrators. In CM, all parameters except scheduling rules are inherently determined by the system and workload, which are not easily changeable. In PM, in addition to scheduling rules, the parameters *a*, *b*, and *c* can be modified by the administrator to different extents. Therefore, *a*, *b*, *c* can be selected and scheduling rules taken to be our factors.

The factor *a* corresponds to FMP. Because of it is easy to change, its level will range from very small, 0.2, to a reasonably large value 3, with a step size of 0.2. We choose a to range from 0.2 to 3 because FMP= $a \times$ MTTF, and we want to make FMP and MTTF approximately comparable, otherwise either preemptive maintenance or failure would almost never happen in the PM model.

The factor *b* corresponds to FMT, which is expected to be smaller than $\mathbf{E}$(MT), because maintaining the machine at working condition is easier than doing so at machine failures. But we do not exclude the possibility of FMT > $\mathbf{E}$(MT). Meanwhile, we note that FMT is not as flexible as FMP: it is varied by maintaining different components of the machines. Usually error-prone components are maintained periodically and other stable components are attended only after their failure. By doing so, we can shorten FMT. Based on the above, the level of *b* will be taken to range from 0.2 to 2.0, with each step of 0.2.

The factor *c* corresponds to $MTTF_p$, which is expected to be larger than $MTTF_c$, because after all, this is why we may want to adopt PM. $MTTF_c$ is around the same level of flexibility as FMT for exactly the same reason. Usually error-prone components are maintained periodically because this will improve *c* more effectively than other components. Hence, the level of *c* will range from 1.0 to 3.0, with step-size of 0.25.

Here, the scheduling rule can be either LPT or SIRO, and our objective is to compare the performance of CM and PM in machine scheduling with machine failures and observe how much parameters and factors affect performance.

**3.4 Experiment Procedure** Because of the large number of factors and their levels, it is impossible and impractical to plan a full experiment. For each environment (single or multiple-machines), we conducted the experiment in the following way. First, we ran test cases on CM with LPT and SIRO respectively. The rule yielding in better results was applied in PM. Then, at each level of a factor, we ran the same test cases on PM, compared the results with that of CM, and observed how performance varied with each factor. For each level of each factor, there were 100 test cases.

**3.5 The necessity of experiments** An experimental approach is taken in this study for several reasons. First, practitioners are more interested in average-case performance than the worst-case performance, whereas traditional analysis focuses on the worst cases and can provide only limited insights to the average case. Second, taking machine failure into account complicates the problem by introducing random variables with known or unknown distributions. For example, the time between failures is a normal random variable, but the failure count (the number of failures) and the wasted time due to interruption have unknown distributions and are not amenable to the simple worst-case and statistical analysis. Third, in order to compare the two maintenance schemes, we need to vary some parameters and factor levels, which is almost impossible by analytical

characterizations alone. Only by running tests and analyzing the collected data is it possible to observe the performance in various situations.

**3.6 Reproducibility** Since reproducibility is a crucial part of any experimental study, we describe the random number generator (RNG) used in the experiments. A Java software package, COLT, consisting of six libraries for scientific and technical computing, is used to generate exponential and normal distributions (http://hoschek.home.cern.ch/hoschek/colt/V1.0.3/doc/index.html). Ranecu, which is an advanced multiplicative linear congruential random number generator with a period of approximately $10^{18}$, is used. Using this, we generated all job processing times and the time between failures. Test cases were run on a Pentium III (600MHz) with 128MB memory and programs were written in Java, and compiled and run with Java sdk1.4.0 and COLT1.0.3.The experiments' results are obtained in very short run time.

# 4. Single-machine Experiments

**4.1 Selecting scheduling rule** For single-machine scheduling minimizing $C_{max}$ with no maintenance and failure, any non-delay schedule is an optimal solution. Data on $C_{max}$ and TMT were collected from 100 test cases using a CM maintenance scheme. A 5% significance hypothesis testing was carried out to confirm or reject the conjecture. Table 2 shows the sample means and standard deviations of $C_{max}$ and TMT respectively of 100 test cases used.

| Single machines | $C_{max}$ | | TMT | |
|---|---|---|---|---|
| | $\hat{\mu}$ | $\hat{\sigma}$ | $\hat{\mu}$ | $\hat{\sigma}$ |
| SIRO | 101964.37 | 1155.2 | 1004.57 | 12.0 |
| LPT | 101965.28 | 1139.2 | 1004.07 | 12.1 |

**Table 2 Summary of data from 100 test cases in single-machine environment**

From the test data and the results of hypothesis testing, we conclude with 5% significance level that there is a negligible difference between the scheduling rule SIRO and LPT in terms of $C_{max}$ and TMT. However, in LPT the jobs are first to be sorted in non-decreasing processing time, so it takes more time and computational cost; whereas in SIRO, jobs can simply be processed in input order because they are randomly generated, and thus are in random order. Therefore, SIRO is preferable and is used here for the single-machine environment.

### 4.2 Effects of the factors *a*, *b* and *c*

*Factor a*: Before experimentation, we first analyzed how performance will be affected by *a* using PM. Since *a* is positively proportional to FMP, a small *a* indicates frequent maintenance, which can reduce failure rate but also incur more maintenance time. So a small *a* is expected to offset the effect of increased $MTTF_p$ and even cause performance to be worse off than that using CM. On the other hand, if *a* becomes large, the performance of PM will converge to that of CM, because failure frequently occurs before fixed-time maintenance is carried out. Based on the preliminary analysis, we inferred that there should be an optimal value of *a* which will maximize the performance difference

between PM and CM. Experiments with various values of *a* were performed to collect data of $C_{max}$ and TMT for CM and PM respectively; meanwhile, other parameters were set as shown in Table 1, and factor *b* and *c* fixed at 0.6 and 2 respectively. Two graphs were plotted for the test data.

In Figure 3(a), the x-axis corresponds to the *a* value, and the y-axis, called the ratio of $C_{max}$ gain, corresponds to the values $(C_{max,CM} - C_{max,PM})/C_{max,CM}$. In Figure 3(b), the x-axis is still the *a* value, and the y-axis, called the ratio of TMT gain, corresponds to the values $(TMT_{CM} - TMT_{PM})/TMT_{CM}$. Above the dash-dot line at 0, PM outperforms CM, and vice versa. From the graphs, we see that the two curves have similar shape: when *a* is less than 1, CM is better; when *a* is between 1.5 and 2, the ratio reaches maximum, indicating the optimal value of *a* for PM; when *a* is greater than 2, the ratio converges to 0, indicating the close performance of CM and PM. There are several points worth noting here. First, the two curves are similar because in single-machine scheduling, $C_{max} = \sum pt+TMT+wt$, where pt is processing time, and wt is wasted time due to the resumption of jobs, and for every test case, $\sum pt$ is a constant, wt is usually small and dependent directly on *a* only. Second, for most *a* values the points of the ratio of TMT gain cluster more densely than those of $C_{max}$. In this figure and the followings, "x" represents individual sample value, and the purple "X" stands for the sample mean; and the purple line is simply the linear interpolation of sample means.

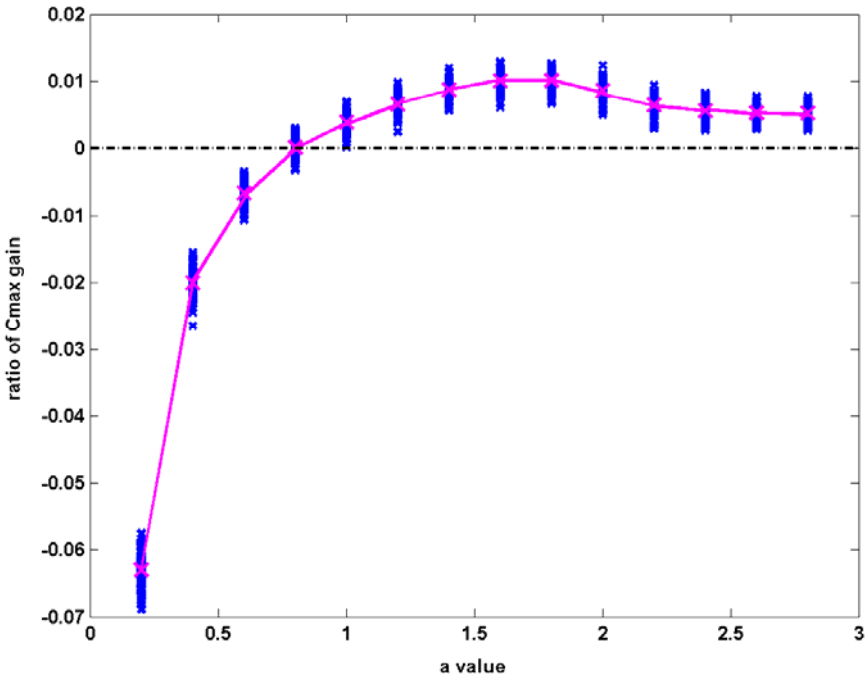


**Figure 3(a)**
x: individual sample
X: sample mean
– : line connecting means

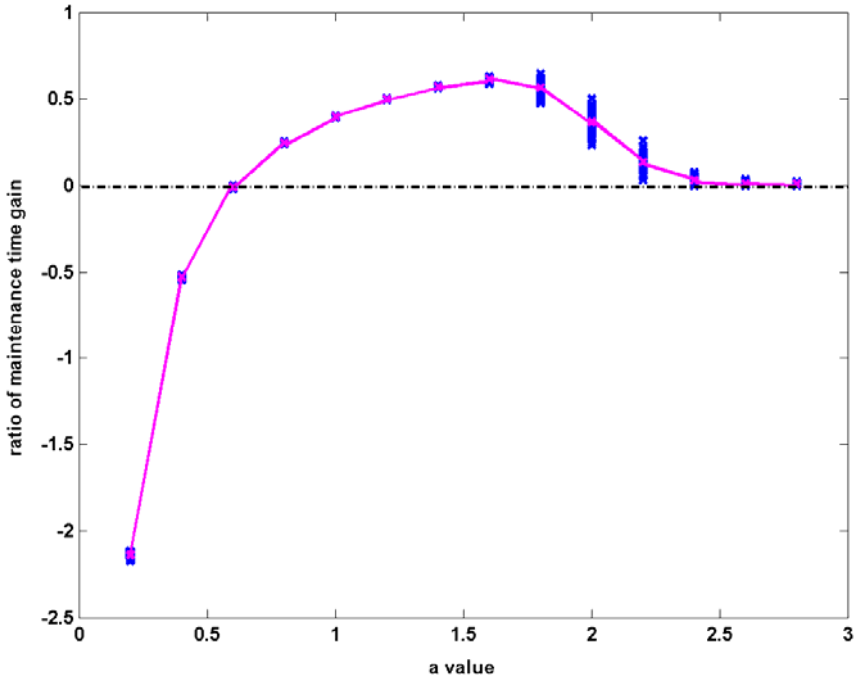


**Figure 3(b)**
x: individual sample
X: sample mean
– : line connecting means

**Figure 3 The effects of factor *a* on performance metrics.**

**(a) shows how $C_{max}$ varies with *a*; (b) shows how TMT varies with *a*.**

*Factor b*: Since *b* is positively proportional to FMT, a smaller *b* is considered desirable. When *b* becomes large, the performance of PM deteriorates and can be worse off than that of CM. Because of this, we infer that the performance is negatively related to the *b* value, and we would want to determine experimentally whether the relationship is linear.

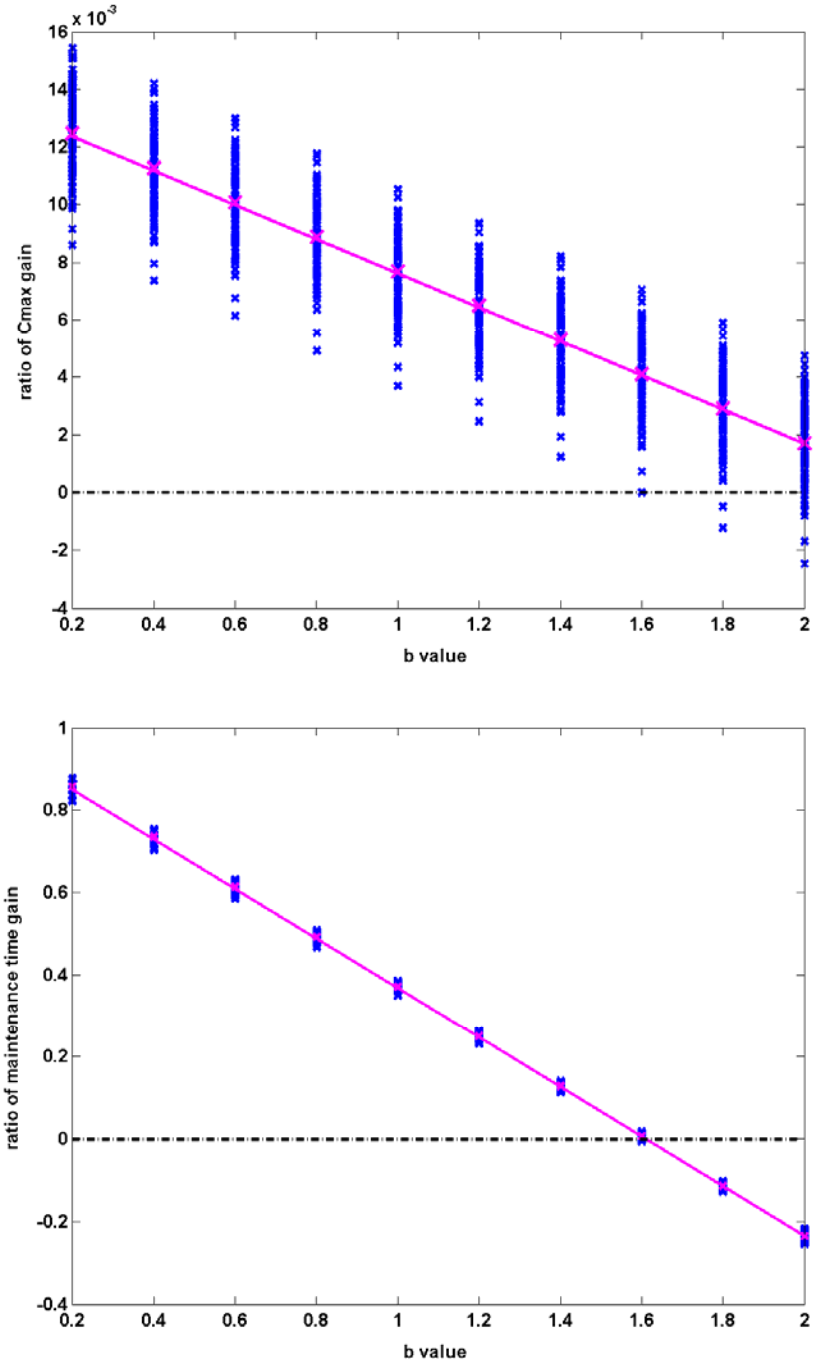


**Figure 4 (a)**
x: individual sample
X: sample mean
– : line connecting means

**Figure 4 (b)**
x: individual sample
X: sample mean
– : line connecting means

**Figure 4 The effects of *b* on performance metrics.**

**(a) shows how $C_{max}$ varies with *b*; (b) shows how TMT varies with *b*.**

Experiments with various values of *b* were done to collect data of $C_{max}$ and TMT for CM and PM respectively; meanwhile, other parameters were set as shown in Table 1, and factor *a* and *c* fixed at 1.6 and 2 respectively (this is the optimal value for *a* in our case). As already mentioned, there is no definite reason why *b* cannot be greater than 1, although this is unlikely. Two graphs were plotted for the test data. In Figure 4(a) is a plot of *b* value against ratio of $C_{max}$ gain; Figure 4(b) is a plot of *b* value against ratio of TMT gain. From the graphs, we can make the following observations. First, in a reasonable range of *b*, i.e. from 0.2 to around 1.6, both metrics in PM are better than those in CM (Of course, this claim is valid when other parameters were set as we do here). However, when $b > 1.6$, the ratio for TMT is negative but the ratio for $C_{max}$ is positive. This is because the wasted time in CM is more than the wasted time in PM (as a result of $c$=2). Furthermore the performance is almost perfectly negatively linear with *b*. So it makes sense to minimize the *b* value. For example, based on past experience or historical data, we can identify the most frequently failed components, and carry out fixed-time maintenance only on these components.

*Factor c:* The factor *c* is positively proportional to $MTTF_p$, and intuitively, the larger the *c* value the better it is. If we fix the value of other parameters and factors and increase *c*, then once *c* reaches a certain value, we expect little performance improvement as *c* continues to grow because most of the failure will be filtered out by PM. Based on this analysis, we decided to modify the experiment. First, we fixed the factor *a* and all other parameters, and attempted to find the critical *c* value after which little improvement is

obtained (this will be done for TMT only). Next, we set $a$ to be $c - 2\times(std/\text{MTTF}_c)$ as $c$ varies, and observed how $c$ affects performance.

Figure 5 shows the ratio of TMT gain against the $c$ value. The ratio increases until $c$ reaches 2 after which it remains relatively constant. This again substantiates our previous inference that PM has best performance when $c = a + 2\times(std/\text{MTTF}_c)$ or equivalently, $a = c - 2\times(std/\text{MTTF}_c)$. Furthermore we claim that once $a$ is fixed, there is no point to make $c$ too large. In other words, if somehow FMP is decided, we do not need to reduce the failure rate too much, as long as $\text{MTTF}_p \approx \text{FMP} + 2\times std$.

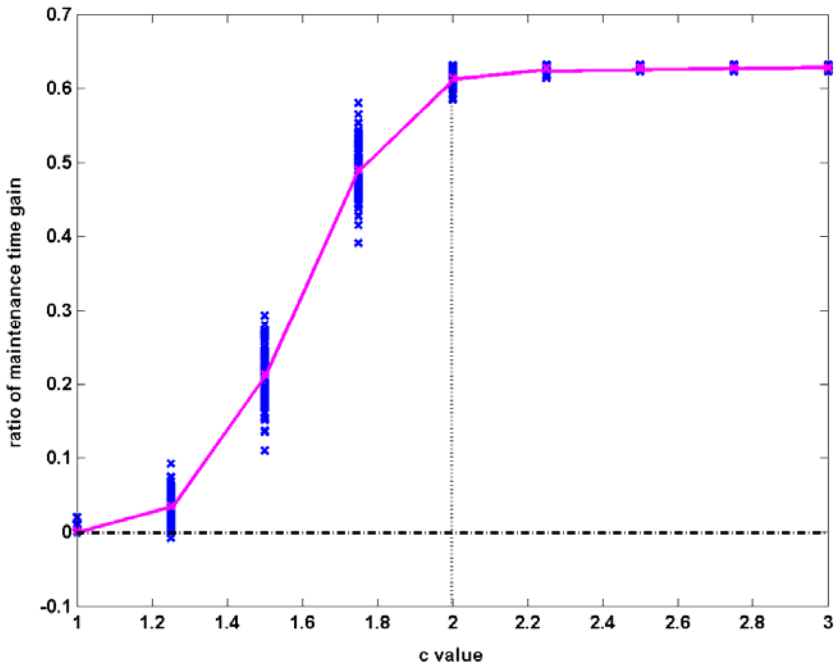


**Figure 5**
x: individual sample
X: sample mean
–: line connecting means

**Figure 5 The effects of factor *c* on TMT when factor *a* and other parameters are fixed**

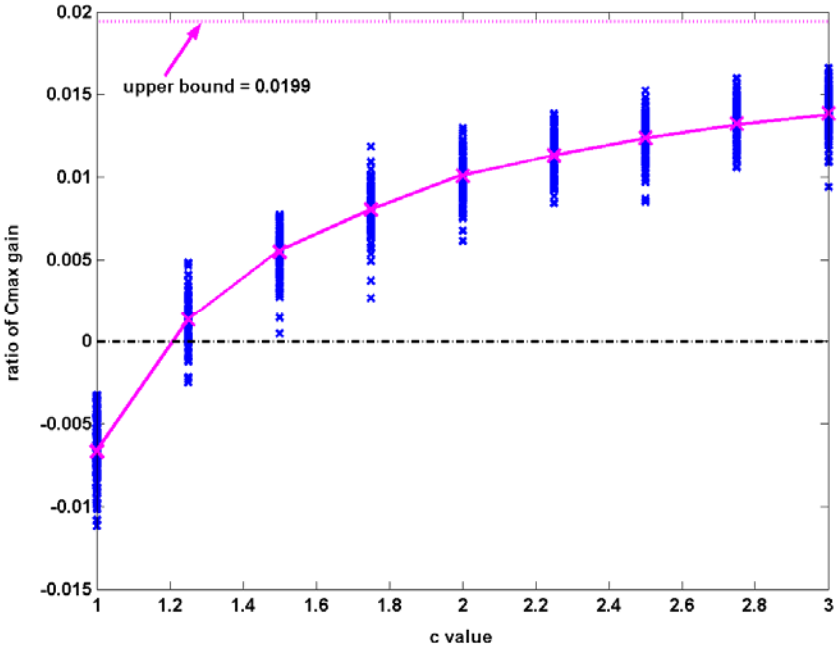


**Figure 6(a)**
x: individual sample
X: sample mean
– : line connecting means
--- : upper bound

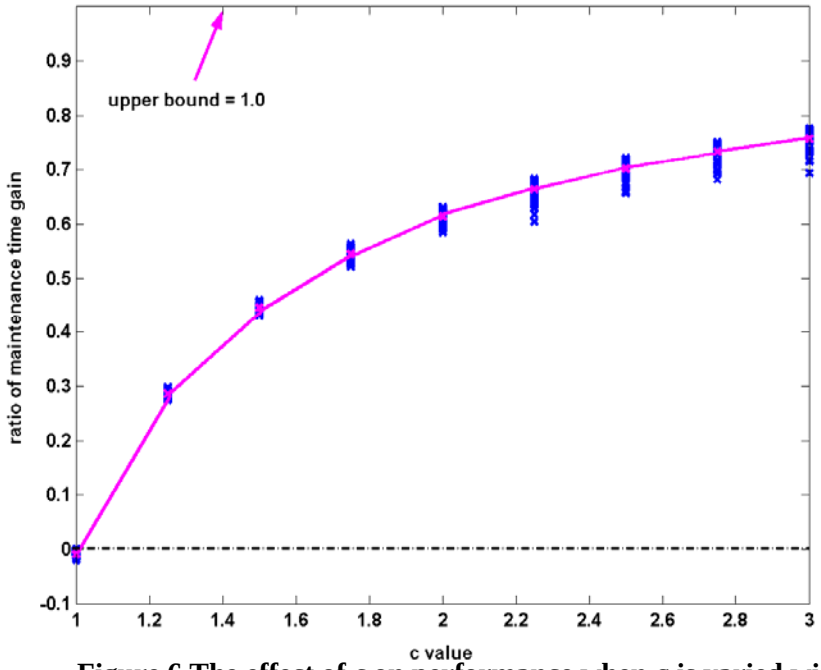


**Figure 6(b)**
x: individual sample
X: sample mean
– : line connecting means

**Figure 6 The effect of *c* on performance when *a* is varied with *c***

**(a) shows how $C_{max}$ varies with *c*; (b) shows how TMT varies with *c***

Figure 6 shows how *c* affects the performance if we allow *a* to be $c - 2\times(std/\mathrm{MTTF}_c)$ as *c* varies from 1 to 3. The curves indicate that the performance monotonically increases

with *c*. Both metrics have an upper bound on their means. The bound on ratio of $C_{max}$ gain is $(C_{max,CM} - \sum pt)/C_{max,CM}$; the bound on ratio of TMT gain is $(TMT_{CM} - 0)/TMT_{CM}$ =1. The bounds were reached when *c* and *a* were so large that no maintenance is ever carried out during the processing of all jobs. In Figure 6(a), there is a small range of *c* in which the ratio of $C_{max}$ gain is negative, whereas in the same range, the ratio of TMT gain in Figure 6(b) is above 0. This is because when *c* and *a* are small, the wasted time in PM is more than that in CM due to the frequent interruptions for maintenance. For most values of *c* (and corresponding *a*), the ratio of TMT gain is greater than 0, which may not be always true because the ratio should depend on *b* and other parameters as well

### 4.3 Analysis of the PM procedure

To further understand the effect of the PM procedure, we have also developed a different model that more directly incorporated the idea that preventive maintenance would have a significant impact on the MTTF. In this PM model, when a preemptive maintenance happens, the MTTF is elongated by 5% (reflecting the fact that after preemptive maintenance the failure rate drops); and if a failure occurs, the MTTF is reduced by 5% because a second failure is more likely to happen if the first has already occurred. The same parameter settings as the previous experiments are used.

In experiments, we analyze how and why the factors *a* and *b* will affect the performance. Because $MTTF_p$ is no longer a constant in our model, the factor *c* (which is the linear factor between $MTTF_c$ and $MTTF_p$ does not exist anymore. To simulate the happenings of failure, we would generate the TBF according to the described normal distribution and

compare it with FMP. If TBF is smaller, it means the next failure happened earlier than the next scheduled maintenance, so the maintenance immediately took place after the failure and MTTF is shortened by 5%; if TBF is larger than FMP, the maintenance would happen before the failure and thus prevent the failure and increase MTTF by 5%. No matter which of TBF or FMP is smaller, the next time a new TBF will be generated according to the new MTTF value and compared with FMP and continue the process until all jobs are finished. It turns out that since we do not allow preemption the LPT rule or SIRO rule would not affect the performance in single-machine environment as only the total length of all jobs would matter. So in our experiment we will test on the effects of a and b only.

We used 100 test cases with the same distribution as before, i.e., in each test case there are 10000 jobs with job processing time sampled from an exponential distribution with mean 10. The original CM model and this new PM model are then applied to each test case with parameter *a* varying from 0.2 to 3 and *b* from 0.2 to 2 (refer to Table 1). Since *a* and *b* are only used to determine the parameters in the PM model, the CM model's performance is independent of *a* and *b*. The performance of PM on the 100 test cases are similar and they vary a lot across the choices of *a* and *b* with a mean standard deviation of 173163. Because the results on all 100 cases are similar, for simplicity we present results of one of the test case (picked at random) in details in Figure 8:

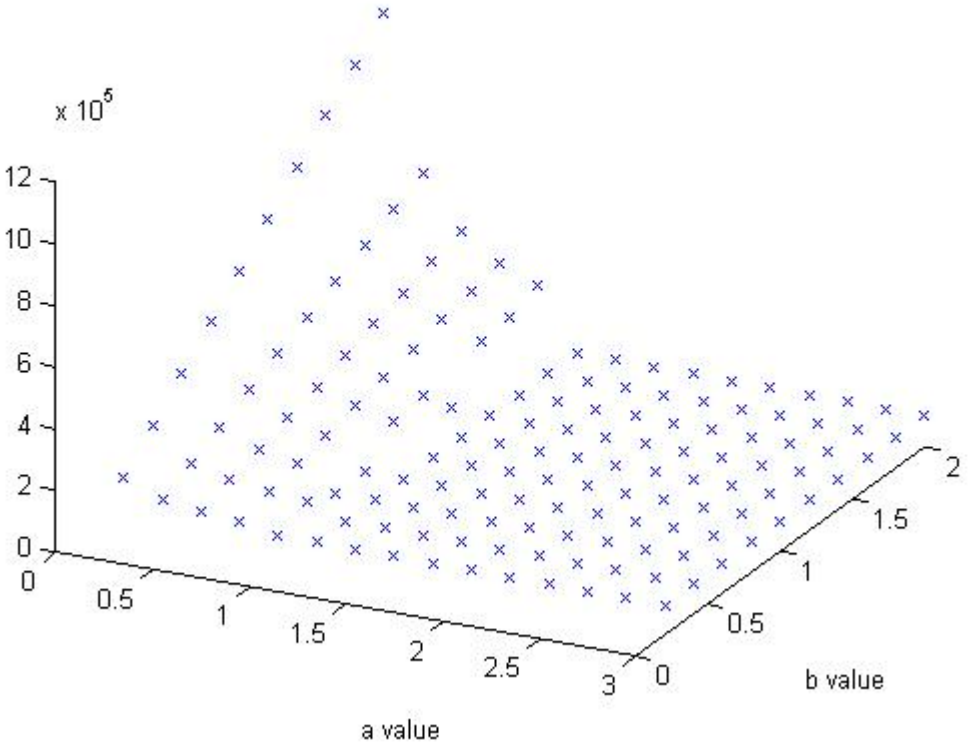


**Figure 7 Performance of PM**

In Figure 7, the z-axis is the $C_{max}$ value with the corresponding *a*, *b* parameters. We clearly see a pattern in the figure: the region corresponding to small a values (less than 1) has a slope shape bending towards the region with small b value and relatively large a values, while the region corresponding to a larger than 1 is a plateau. For the small a-value region this is as expected because when *a* is small, FMP= $a \times$ MTTF is small, resulting in frequent maintenances that a failure would hardly happen and the Cmax time is largely consisting of maintenance time. So the smaller maintenance time, the better the performance. As FMT= $b \times$ MT, so the smaller the *b* value the better the performance in this region.

In the regions of a “plateau”, the performance has no statistical difference from the CM model. This can be understood in the following sense: in this region a value is large, so failures are more likely to happen. After several failures happened before any maintenance, the MTTF value will be so small that future interruption of jobs would almost totally consist of failures rather than preemptive maintenance. Assuming all interruptions are failures, we can prove that the performance is independent of a or b (thus resulting in the plateau region) and converges to the CM model, implying performance is the same as the CM model.

It is trivial to argue that the performance is independent from a-value or b-value, since a or b only determine FMP or FMT which are irrelevant here because no maintenance can take place. To show that the model is the same as the CM model, we need to argue that the 5% MTTF increment after each failure somehow has no effect on $C_{max}$. This is true because the maintenance time after each failure MT= $k \times$ TBF. And $\mathbf{E}$(TBF) = MTTF. So MT is linear in MTTF. No matter how many times MTTF is decremented by 5% after each failure, the linear relationship between MT and MTTF is maintained. By a simple calculation one can show that the total time spent on maintenance after failure (sum of all MTs) is independent of the 5% change of MTTF after failure. This explains the reason of having a plateau when a is large. In addition, because the 5% bias in MTTF each time has no effect on $C_{max}$ in this region, together with the fact that almost no maintenance actually took place, the PM model is the same as the CM model when a is large.

In particular, the a value that divides the “slope” and “plateau” is 1. This can be explained as follows: $a$=1 indicates FMT=MTTF. So at the beginning the first interruption of job processing is equally likely to be a preemptive maintenance or a failure. But either one would result in a bias of 5% of MTTF value towards either the next interruption is a preemptive maintenance or failure. So $a$=1 is the dividing line of these two regions.

In summary, in the above model where maintenance and failure both have very direct impact on MTTF, we propose CM as a more appropriate model from the experiments and analysis we performed.

# 5. Multiple-machine Experiments

**5.1 Selecting scheduling rule** For multiple-machine scheduling minimizing $C_{max}$ with no maintenance and failure, LPT is a well-studied heuristic that has a worst-case bound of 4/3. It is expected that LPT will produce better schedules than SIRO does in $Pm||C_{max}$. Data on $C_{max}$ and TMT were collected from 100 test cases using CM maintenance scheme. A 5% significance hypothesis testing is carried out to confirm or reject the above conjecture. Table 3 shows the sample means and standard deviations of $C_{max}$ and TMT respectively.

| Multiple machines | $C_{max}$ | | TMT | |
|---|---|---|---|---|
| | $\hat{\mu}$ | $\hat{\sigma}$ | $\hat{\mu}$ | $\hat{\sigma}$ |
| SIRO | 104391.45 | 831.04 | 10053.84 | 34.44 |
| LPT | 102053.37 | 335.83 | 10054.05 | 35.43 |

**Table 3 Summary of data collected from 100 test cases in multiple-machine environment**

Because LPT is assumed to be better, the hypothesis testing is changed to a one-tailed test. From the test data and the results of hypothesis testing, we concluded with 5% significance level that LPT yields better results than SIRO w.r.t. $C_{max}$, but has similar performance as SIRO w.r.t. TMT. Hence, LPT will be used to compare the performance of PM and CM in the following study.

### 5.2 Effects of factors *a*, *b* and *c*

*Factor a:* As inferred for the single-machine case, there should exist an optimal *a* value for PM in a preset scheduling environment. Similar experiments with various values of *a* were carried out to collect data of $C_{max}$ and TMT for CM and PM respectively. Note that the number of machines in this test is 10, and the number of jobs is increased to 10 times the original, so that each machine still has 10000 jobs on average. Unlike in single-machine case, there is no direct or explicit relationship between $C_{max}$ and TMT in multiple-machine case. Figure 8 shows the effect of *a* on performance in multiple-machine environment. Most of the observations and hypotheses we make are similar to those in single-machine. In spite of the increased number of machines, the two curves are

similar to the two given in Figure 3: when *a* is less than 1, CM outperforms PM; when *a* is between 1.5 and 2, the ratio reaches maximum at about 0.01, indicating the optimal value of *a* for PM; when *a* is greater than 2, the ratio converges to 0, indicating the comparable performance of CM and PM. Also, we again observe that the optimal value of *a* for PM is around $c - 2\times(std/\mathrm{MTTF_c})$.

*Factor b:* As previous tests on *b* reveal a negative linear relationship between the *b* values and the performance, we believe that in multiple-machine case a linear relationship will still hold. Similar experiments with various values of *b* were carried out to collect data on $C_{max}$ and TMT for CM and PM. Two graphs were plotted based on the test data. Figure 9 verifies the negative relationship between *b* and the two metrics, and the two lines are almost the same as in Figure 4 (in terms of slope and intersects). Other observations were similar to those in single-machine case, and minimizing *b* as much as possible is still applicable.

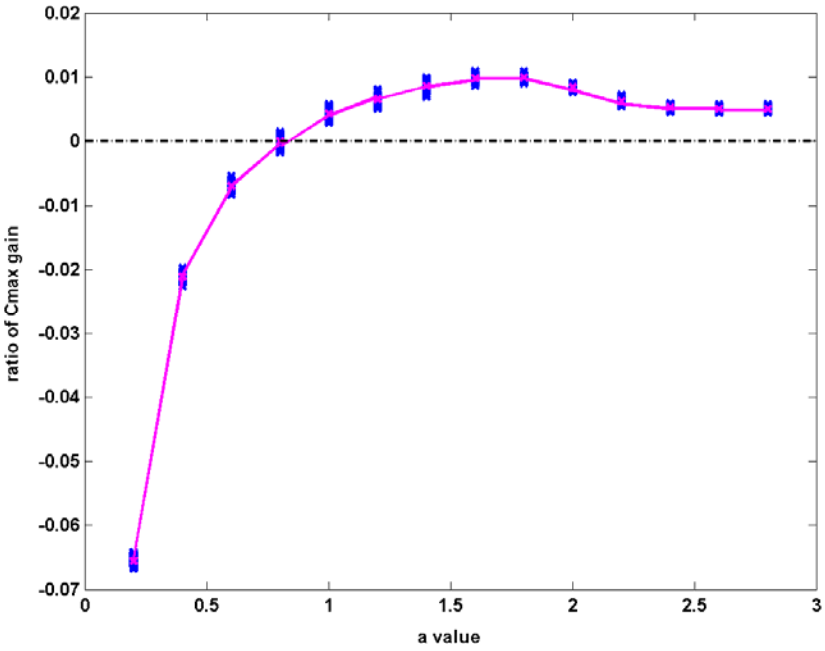


**Figure 8(a)**
x: individual sample
X: sample mean
– : line connecting means

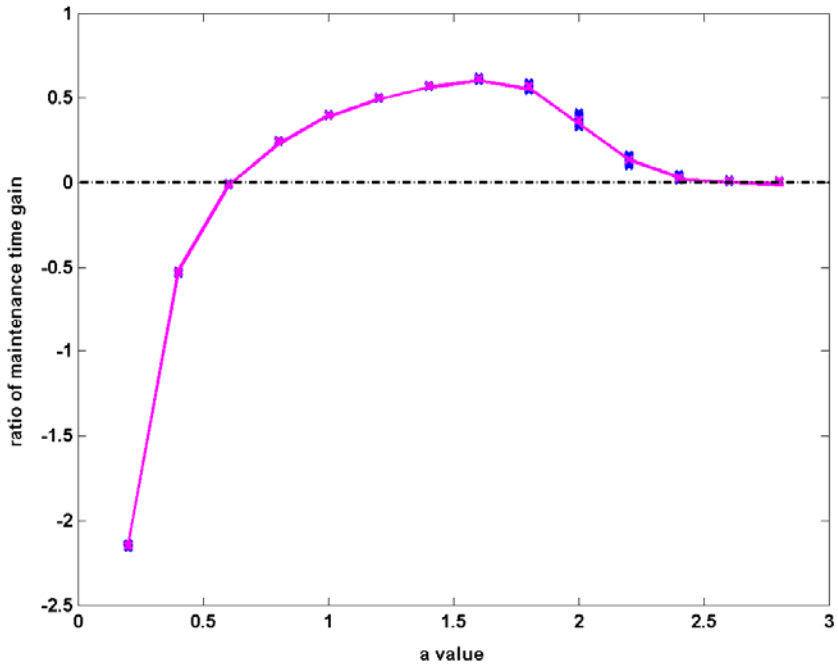


**Figure 8(b)**
x: individual sample
X: sample mean
– : line connecting means

**Figure 8 The effect of *a* on performance in multiple-machine case**

**(a) shows how $C_{max}$ varies with *a*; (b) shows how TMT varies with *a***

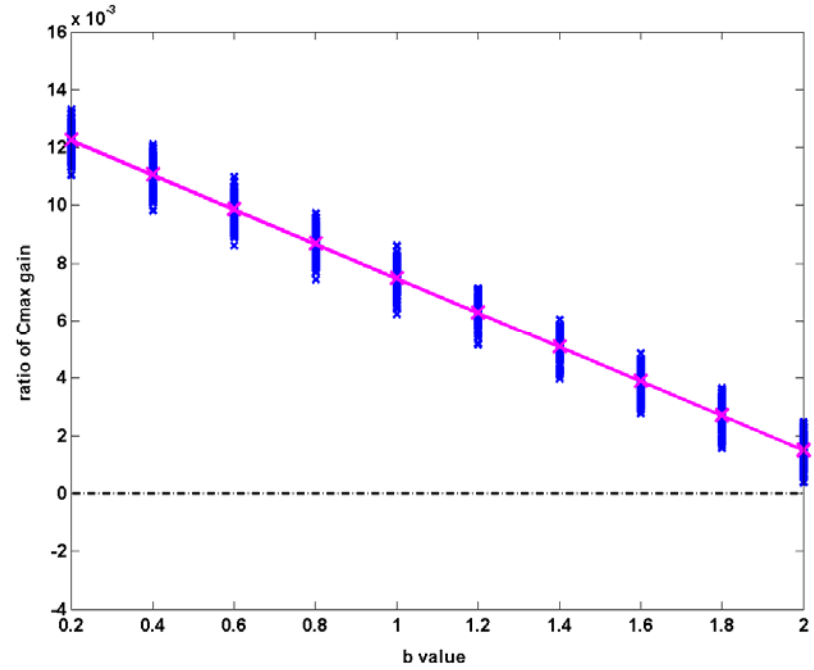


**Figure 9(a)**
x: individual sample
X: sample mean
– : line connecting means

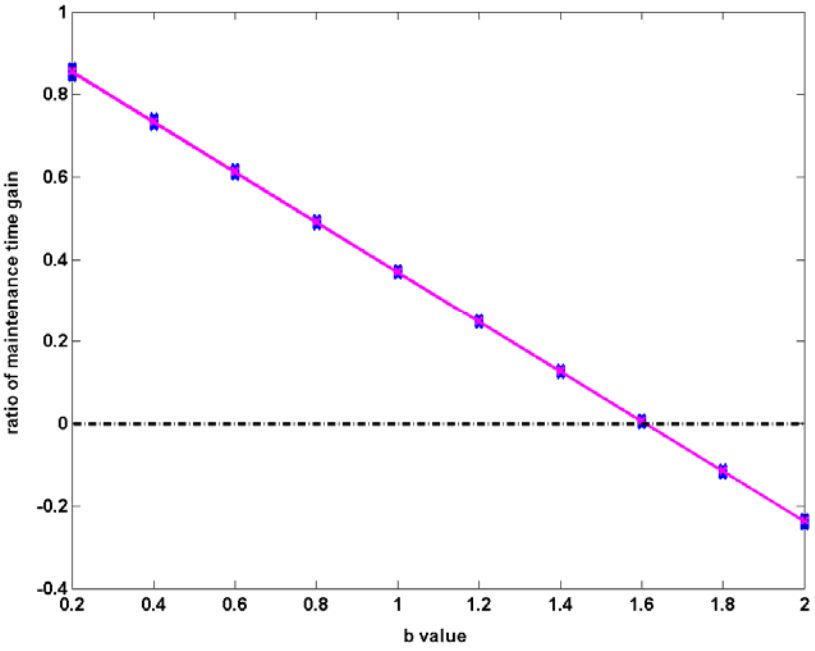


**Figure 9(b)**
x: individual sample
X: sample mean
– : line connecting means

**Figure 9 the effect of *b* on performance in multiple-machine case**

**(a) shows how $C_{max}$ varies with *b*; (b) shows how TMT varies with *b***

*Factor c:* In the multiple-machine case, we omit the experiment of varying *c* while *a*, *b* and other parameters were fixed, because each machine is assumed to work independently and their failure rates were not affected by the number of machines. (In our case, the failure rates are equal.) The claim that there is a critical value for *c* after which the performance rarely improves is still valid. We proceed to the experiment of varying both *c* and *a* by setting a to be $c - 2\times(std/\mathrm{MTTF_c})$. Figure 10 shows the effects of c on performance. The bound on the mean of $C_{max}$ is 0.0196 and the bound on the mean of TMT remains at 1.

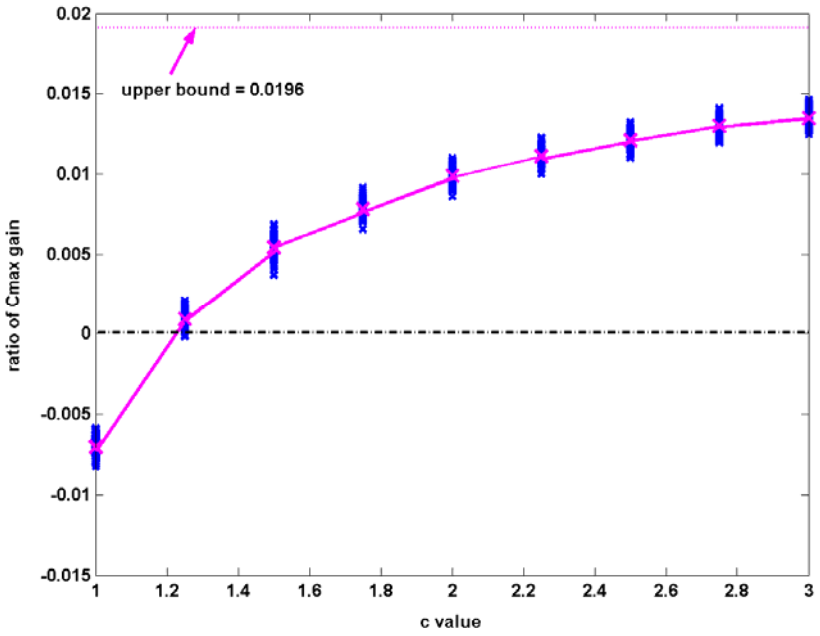


**Figure 10(a)**
x: individual sample
X: sample mean
– : line connecting means

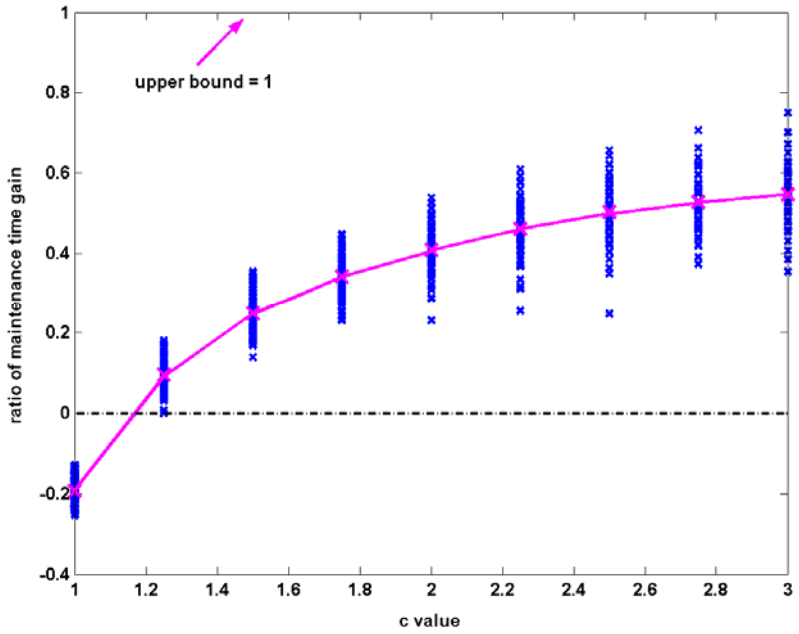


**Figure 10(b)**
x: individual sample
X: sample mean
– : line connecting means

**Figure 10 The effect of *c* on performance in multiple-machine case with**

$$a = c - 2\times(std/\mathrm{MTTF_C}).$$

**(a) shows how $C_{max}$ varies with *c*; (b) shows how TMT varies with *c***

# 6. Summary

In this study, we experimentally evaluated and compared the performance of the two maintenance schemes in single- and multiple- machine scheduling with machine failure and maintenance. We found that there were a number of parameters that affect performance. Based on the flexibility in modification, three factors *a*, *b*, and *c* were used in this study, and tests conducted. The results suggest that PM outperforms CM to the largest extent when *a* is at $c - 2\times(std/\text{MTTF}_c)$, *b* is as small as possible (at most around 1.6), and *c* is reasonably large (near 2 or above). These conclusions, and the analysis provided can be useful to system administrators and practitioners in environments where machine failure and maintenance are factors in performance and provide a basis for further experimental study.

# References

[1] Albers, S. & Schroder, B. (2002), An Experimental Study of Online Scheduling Algorithms, *ACM Journal of Experimental Algorithmics*, 7(3), 123 -154

[2] Baev, J.D. Meleis, W.M. and Eichenberger, A. (2002), An Experimental Study of Algorithms for Weighted Completion Time Scheduling, *Algorithmica*, 33, 34-51

[3] Barr, R.S. Golden, B.L. Kelly, J.P. Resende, M.G.C. & Stewart, W.R. (2001), Designing and Reporting on Computational Experiments with Heuristic Methods, *Journal of Heuristics*, 1(1), 9-32

[4] Bentley, J.L. Johnson, D.S. Leighton, T. & McGeoch, C.C. (1983), An Experimental Study of Bin Packing, *Proc. 21st Annual Allerton Conference on Communication, Control, and Computing*, 51-60

[5] Coffin, M. and Saltzman, M.J. (2000), Statistical Analysis of Computational Tests of Algorithms and Heuristics, *INFORMS Journal on Computing*, 12 (1), 24-44

[6] Coffman, E.G. Jr, Garey, M.R. & Johnson, D.S. (1978), An Application of Bin-Packing to Multiprocessor Scheduling, *SIAM Journal on Computing* 7, 1-17

[7] Gent, I.P. Grant, S.A. MacIntyre, E. Prosser, P. Shaw, P. Smith, B.M. & Walsh, T. (1994), How Not to Do It, Presented at *AAAI-94 Workshop on Experimental Evaluation of Reasoning and Search Methods*

[8] Graham, R.L. (1969), Bounds on Multiprocessing Timing Anomalies, *SIAM Journal of Applied Mathematics*, 17, 263-269

[9] Hooker, J.N. (1994), Needed: An Empirical Science of Algorithms, *Journal of Operations Research*, 42, 201-212

[10] Hooker, J.N. (1996), Testing Heuristics: We Have It All Wrong, *Journal of Heuristics*, 1 (1), 33-42

[11] Johnson, D.S. (2002), A Theoretician's Guide to the Experimental Analysis of Algorithms, In M.H. Goldwasser, M. H., Johnson, D.S. & McGeoch, C.C. (Eds.), *Fifth and Sixth DIMACS Implementation Challenges*, American Mathematical Society, Providence

[12] Kaspi, M. and Montreuil, B. (1988), On the Scheduling of Identical Parallel Processes with Arbitrary Initial Processor Available Time, *Research Report 88-12*, School of Industrial Engineering, Purdue University

[13] Lee, C.Y. and Massey, J.D. (1988), Multiprocessor Scheduling: Combining LPT and MULTIFIT, *Discrete Applied Mathematics*, 20, 233-242

[14] Lee, C.Y. (1996), Machine Scheduling with an Availability Constraint, *Journal of Global Optimization*, 9, 395-416

[15] Lee, C.Y. (1991), Parallel Machines Scheduling with Nonsimultaneous Machine Available Time, *Journal of Discrete Applied Mathematics*, 30, 53-61

[16] Lee, C.Y. & Liman, S.D. (1992), Single Machine Flow-time Scheduling with Scheduled Maintenance, *Acta Informatica*, 29, 375-382

[17] McGeoch, C.C & Moret, B. M. E. (1999), How to Present a Paper on Experimental Work with Algorithms. *SIGACT News* Vol. 30, No. 4, pages 85-90

[18] McGeoch, C.C. (2001), Experimental Analysis of Algorithms, *Notices of the American Mathematical Society*, 48(3), 304-311

[19] Moret, B.M.E. (2002), Towards a Discipline of Experimental Algorithmics, In M.H. Goldwasser, M. H., Johnson, D.S. & McGeoch, C.C. (Eds.), *Fifth and Sixth DIMACS Implementation Challenges*, American Mathematical Society, Providence, 2002.

[20] Pinedo, M. (1995), *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, New Jersey

[21] Qi, X. Chen, T. & Tu, F. (1999), Scheduling the Maintenance on a Single Machine, *Journal of the Operational Research Society*, 50, 1071-1078

[22] Savelsbergh, M.W.P. Uma, R.N. & Wein, J. (1998), An Experimental Study of LP-based Approximation Algorithms for Scheduling Problems, *Proceedings of 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 453-462

[23] Schmidt, G. (1988), Scheduling Independent Tasks with Deadlines on Semi-identical Processors, *Journal of Operational Research Society*, 39, 271-277

[24] Turkcan, A. (1999), Machine Scheduling with Availability Constraints, Available at benli.bcc.bilkent.edu.tr/~ie672/docs/present/turkcan.ps