Replication of Data Placement for Uncertain Scheduling

Manmohan Chaubey, Erik Saule
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, USA
Email: mchaubey@uncc.edu, esaule@uncc.edu

***Abstract*—The abstract goes here. DO NOT USE SPECIAL CHARACTERS, SYMBOLS, OR MATH IN YOUR TITLE OR ABSTRACT.**

***Keywords*-component; formatting; style; styling;**

I. Introduction

Many real world scheduling problems related to task allocation in parallel machines are uncertain in nature. This fact makes the class of problem known as 'scheduling with uncertainty' or 'robust' scheduling an important and most studied problem in Scheduling. Robustness is the measure of to what extent an algorithm can cope with uncertainties in a scheduling problem. Often in scheduling problems the exact processing time of a job is not known initially at the time of planning for job- allocation to different machines, but they have a range outside which their values cannot lie. The fact gives a basis to estimate the processing times and to take job assignment decision based on these estimated values. The actual processing time of a job can vary drastically from its estimated one. So, any algorithm have to incur the cost of estimation which impact its performance negatively. A robust approach deals with uncertainty in input parameters to minimize this cost.

The objective of our research is to study the affect on performance of parallel machine scheduling when jobs are scheduled based on their estimated values of processing times, and to propose scheduling approaches that optimizes the performance in this environment. Data placement and replication techniques can be useful in the scenario of uncertain processing times of tasks. Effective data placement using replication strategy allows better load balancing and hence reduces turnaround job time. This paper is important in the sense that it proposes different models depending upon different scenarios and compares them based on approximation ratio and replication they allow. Replication strategy that allows to place data in a wisely manner offers a faster access to files required by jobs, hence increases the job execution's performance. Replication helps in load balancing but it have certain cost attached with it as it usually increases resource usage [?]. The paper deals with this problem and chooses the scenario in which replication is beneficial. Thus our research focus on two area of scheduling problem i.e. task uncertainty and replication using data placement. The scheduling problem related to data placement and task allocation is common in heterogeneous systems and often dealt with common approach. This paper draws qualitative analysis of the effect of replication in data placement for uncertain scheduling with inexact processing time of a task.

The remaining of the paper is organized as follows: we describe system model and notations in section 2. Sections 3-5 describes different problem models- 1)No replications, 2)replication is done everywhere and 3)replication is done within a group. The respective sections describe derivation of competitive ratios for each of these models. Section 6 summarizes the 3 models and based on experimental results.

II. Problem Definition

We have set J of n jobs which need to be scheduled to set M of m machines such that makespan, C_{max} is minimized. C^*_{max} denotes optimal makespan of a schedule S. We are considering the problems where the scheduler do not know the processing time of a task exactly before it completes, but we have some estimation of the processing time of a task before it is assigned to a processor. We know that the actual processing time of a task i is between $\frac{1}{\alpha}$and α times of its estimated processing time. p_i denotes the actual processing time and $\tilde{p_i}$ denotes estimated processing time for the task i. We have this estimate:

$$\frac{1}{\alpha} \leq \frac{p_i}{\tilde{p_i}} \leq \alpha \tag{1}$$

[Note:give justification]

We consider various problem models incorporating the above mentioned scenario of uncertain processing times of tasks with an estimate. Depending upon a problem model the scheduling is done in two phases or just in one phase. In Phase 1, we allocate tasks to certain processors or group of processors depending upon a problem model. Phase 1 chooses where data to be replicated using estimated processing time $\tilde{p}_i$, for each of the task i. Phase 1 takes $\tilde{p}_i$, m and α as input and outputs set of machines, $M_j \subseteq M$ where a task j can be scheduled or is replicated. Phase 2 takes output of phase 1 as input and outputs set of task assigned to a machine i, $E_i \subseteq J$. In Phase 2, we choose actual schedule with semi-clairvoyant algorithm which uses only approximate knowledge of initial processing time, and after scheduling the task the actual p_i is known. With objective to find schedule which minimizes makespan, we investigate greedy algorithms for each of the problem models and prove their comptetitive ratios.

We have used Graham's List Scheduling (LS) and Largest Processing Time (LPT) algorithms to derive approximation ratios in different scenarios. The LS algorithm takes tasks one at a time and assign it to the processor having least load at that time. LS is 2-approximation algorithm and is widely used in online scheduling problems. The LPT sorts tasks in decreasing order of processing time and assign them one at a time in this order to the processor with the smallest current load. The LPT algorithm have worst case performance ratio as $4/3 - 1/(3m)$ in offline setting . Depending upon which among these two algorithms suits more for a problem model we have used these algorithms accordingly.

III. Related Work

We begin with the literature on scheduling problems under uncertainty. based on various models of describing uncertainty in input parameter, uncertainty problems can be approached by various methodologies including Reactive, stochastic, fuzzy and Robust approach [4]. The bounded uncertainty model assumes that an input parameter have value between a lower and upper bound and is usually dealt with robust approach or sensitivity analysis. We are more interested in robust approach to deal with uncertainty. There is very less literature available for scheduling with estimated processing time of a task. Most of the literature till now focuses on job size estimation with small error. Very less work is done which focuses on coping with less restrictive estimation for task size and which analyzes the effect of estimation on scheduling quantitatively. Wierman and Nuyens introduce SMART as classification to understand size based policies and draw analytical co-relation between response time and estimated job size in single server problem [6]. [?] presents sensitivity analysis for scheduling trees with communication delays on two identical machines. Wagelmans give supporting evidence for the notion that a good performance of an algorithms is identical with a high degree of sensitivity [?]

To obtain robust schedule against uncertainty very few work had been done so far. Most of the work on robust scheduling uses scenarios in problem formulation to structure variability of uncertain parameter. Daniels and Kouvelis [1] used scenarios to formulate branch and bound based robust scheduling to cope uncertainty in processing time of jobs in single machine scenario. Davenport, Gefflot, and Bek used slack based technique to add extra time to cope with uncertainty [2]. Gatto and widmayer derives bounds on competitive ratio of Graham's online algorithm in scenario where processing times of jobs either increase or decrease arbitrarily due to perturbed processing times of tasks [3]. This paper is close to our problem construction. Their notion of perturbed processing times which may increase or decrease arbitrarily is somewhat similar to our definition of estimated processing times which can increase or decrease to actual processing times once the jobs are processed. But unlike our work they deal with increment and decrement of job's processing time separately. We have approached the problem with worst case scenario where some tasks may increase and some may decrease in same schedule.

We have used pro-active approach to deal with uncertainty. Through this research we study effect of load balancing in scenario of uncertain processing time. We propose 3 models offering different degree of task replication to balance the load. Similar work is done by Tse [5] who used selective replication of documents in bi criteria problem of minimizing load and memory in distributed file servers.

Dealing with uncertainty problems with robust approach is widely used in practicality in the area of MapReduce, Hadoop, databases and web servers. [?] [?] proposes schedulers which provides robustness in scheduling against uncertain Job Size. [?] provides scheduling heuristics that optimizes both makespan and robustness in scheduling task graph on heterogeneous system.

A. Our Contribution

We study the effect of load balancing through replication in classic problem of scheduling jobs with uncertain processing times with objective to optimize makespan. We propose 3 models based on different degree of of replication they allow: a)No Replication i.e. $|M_j| = 1$,b) Replication is done every where i.e.$|M_j| = |M|$ and c) Replication in groups i.e $|M_j| = m/k$ where k is number of groups each having m/k processors . For each of the models we propose a two phase algorithm based on Graham's LS and LPT; and derive performance ratio for each. For $|M_j| = 1$, we prove that there is no algorithm which give better performance than α^2. Our algorithm for the model give $\frac{2\alpha^2 m}{2\alpha^2+m-1}$ approximation. For models with replications $|M_j| = |M|$ and $|M_j| = m/k$ the algorthims give performance ratio of $1 + (\frac{m-1}{m})\frac{\alpha^2}{2}$ and $\frac{k\alpha^2}{\alpha^2+k-1}\left[1 + \frac{k-1}{m}\right] + \frac{m-k}{m}$ respectively.

IV. Model 1: No Replication

In this problem model we have considered situation where each task is restricted to be scheduled on only one machine, i.e. $|M_j| = 1$. We have a set J of n jobs, and a set M of m machines. Let $f : J \leftarrow M$ be a function that assigns each job to exactly one machine. Let us denote E_i as the set tasks which is assigned to a machine i. The restriction that each task can be scheduled to only one machine restrict the problem construction to phase one only. There is no replication of tasks in this model.

We have considered List scheduling and Longest processing time algorithms and have assigned each task on machine on which they are restricted to. Based on the estimated processing times of tasks loaded on each machine we will derive makespan.

A. Lower Bound

Thoerem 1.1: For the model having $|M_j| = 1$, there is no online algorithm having competitive ratio better than α^2.

Proof: We use adversary lower bound technique to prove the theorem. We define a lower bound input instance for an unknown correct algorithm. The adversary maximizes the competitive ratio until it becomes feasible to be scheduled by any algorithm.

Figure 1 shows the scenario where $|M_j| = 1$. Let us consider an instance in which total tasks be λm with $\tilde{p_i} = 1$. After scheduling let the most loaded machine j have B tasks. Obviously, $B \geq \lambda$. In phase 2 the adversary increases the tasks on j by α and decreases the other tasks by $\frac{1}{\alpha}$. So, C_{max} = αB and $C^*_{max} \geq \frac{\alpha B + \frac{1}{\alpha}(\lambda m - B)}{m}$. We have,

$$\frac{C_{max}}{C^*_{max}} \leq \frac{\alpha^2 Bm}{\alpha^2 B + \lambda m - B} = \frac{\alpha^2 m}{\alpha^2 + \frac{\lambda m}{B} - 1}$$

From above expression it is clear that smaller the value of B, the value of $\frac{C_{max}}{C^*_{max}}$ decreases. So, any algorithm tries to minimize B to get better performance. But for a schedule to become feasible the condition, $B \geq \lambda$ must be satisfied. Irrespective of the value of λ, for $B \geq \lambda$ the value of expression would be feasible minimum and would equal to $\frac{\alpha^2 m}{\alpha^2+m-1}$. When m tends to ∞ the expression equals α^2. So, any algorithm cannot give competitive ratio better than α^2.

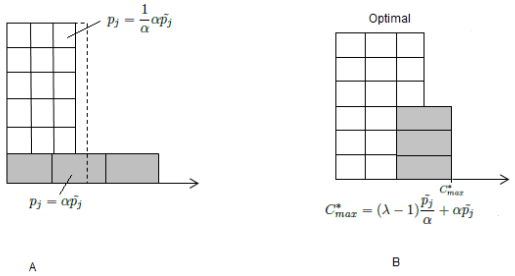

Figure 1. The figure represents the model having $|M_j| = 1$ with $\lambda = 3$ and $m = 6$. A represents worst case scenario showing all the tasks increased to α times in the most loaded machine. B represents optimal scheduling scenario.

B. Algorithm

We introduce an algorithm, **LPT-No Choice** which is based on Graham's LPT algorithm. In phase 1, The algorithm schedules the jobs to different machines in offline mode based on non-increasing order of their estimated processing times. Just like Graham's LPT a job is assigned to a machine having least load at that time. Adhering to the construction of the model, there is no choice of load balancing based on replication in phase 2. The performance of the algorithm depends on how much actual processing times of jobs vary from estimated ones. The actual processing time of a job may increase or shrink significantly from its estimated value and is known exactly only after the job is processed by a machine. So, the algorithm's performance incur the cost of estimation.

Theorem 1.2: The **LPT-No Choice** have competitive ratio of $\frac{2\alpha^2 m}{2\alpha^2+m-1}$.

Proof: The algorithm assigns the jobs to processors based on their estimated processing times using LPT in offline mode. So, the planned makespan considering estimated processing times of tasks, $\tilde{C}_{max}$ have following inequality relation with total estimated processing time, $\tilde{p_j}$ and estimated processing time of last task, l.

$$\tilde{C}_{max} \leq \frac{\sum \tilde{p}_j + (m-1)\tilde{p}_l}{m} \tag{2}$$

The actual makespan of a schedule, C_{max}, calculatedafter actual processing times of all the jobs are known,must be smaller than $\alpha\tilde{C}_{max}$. Considering this we have following inequality:

$$C_{max} \leq \alpha\tilde{C}_{max} \leq \alpha[\frac{\sum \tilde{p}_j + (m-1)\tilde{p}_l}{m}] \tag{3}$$

Considering the worst case situation the processor with $\tilde{C}_{max}$ will see its tasks increase by α and the load on the rest of the processors will by shrink $\frac{1}{\alpha}$. The argument behind this statement is that greater the value of ratio $\frac{C_{max}}{\sum p_j}$, the algorithm will give bad approximation. So increasing the load on the machine which have C_{max} and deceasing the rest of the load on other processors will reach worst case scenario. So the total actual processing time is given by the following equation.

$$\sum p_j = \frac{\tilde{p}_j - \tilde{C}_{max}}{\alpha} + \alpha\tilde{C}_{max} \tag{4}$$

Also the actual optimal makespan have following constraint

$$C^*_{max} \geq \frac{\sum p_j}{m}$$

Substituting for $\sum p_j$, we have

$$mC^*_{max} \geq \frac{\tilde{p}_j - \tilde{C}_{max}}{\alpha} + \alpha\tilde{C}_{max}$$

$$\Rightarrow mC^*_{max} \geq \frac{\tilde{p}_j - [\frac{\sum \tilde{p}_j + (m-1)\tilde{p}_l}{m}]}{\alpha} + C_{max}$$

$$\Rightarrow mC^*_{max} \geq \frac{m-1}{\alpha m}[\sum \tilde{p}_j - \tilde{p}_l] + C_{max}$$

By the property of LPT $\sum \tilde{p}_j - \tilde{p}_l \geq m(\tilde{C}_{max} - \tilde{p}_l)$, putting this we have,

$$mC^*_{max} \geq \frac{m-1}{\alpha}\left[\tilde{C}_{max} - \tilde{p}_l\right] + C_{max}$$

The Graham's LPT algorithm in normal scenario give optimal performance till two jobs are assigned to machines. But in the scenario of estimated processing times of tasks whose value can significantly vary from actual actual ones, any schedule using estimated values cannot guarantee optimality for even two jobs per machine. So, we consider the instance of at least two jobs per machine to derive performance ratio of our algorithm. For at least two jobs in machine having $\tilde{C}_(max)$, the processing time (estimated) of latest job, $\tilde{p}_l$ atmost be equal to $\tilde{C}_(max)/2$. Putting this in the above equation, we have

$$mC^*_{max} \geq \frac{m-1}{\alpha}\left[\tilde{C}_{max} - \frac{\tilde{C}_{max}}{2}\right] + C_{max}$$

Using equation 3,

$$mC^*_{max} \geq \frac{m-1}{2\alpha}\left[\frac{C_{max}}{\alpha}\right] + C_{max}$$

$$mC^*_{max} \geq \left[\frac{m-1}{2\alpha^2} + 1\right] C_{max}$$

$$\frac{C_{max}}{C^*_{max}} \leq \frac{2\alpha^2 m}{2\alpha^2 + m - 1}$$

V. Model 2: replication is done everywhere

In this problem model we consider no restriction on task assignment. The tasks are allowed to be replicated everywhere i.e. i.e. $\forall j, |M_j| = |M|$. We introduce **LPT-No Restriction** algorithm which is also a two phase algorithm. In the first phase no restriction is put into task assignment. It is assumed that each task can be replicated on any machine. In the second phase we simply use the Longest Processing Time algorithm (LPT) using estimated processing times of tasks in non-increasing order as input.

Lemma 2.1: LPT-No Restriction give approximation ratio of $\frac{2}{\alpha^2}p_l$ when there are at least two tasks in the machine to which the last task, l is scheduled.

Proof: As there is at least one task j before l in the machine to which l is assigned, we have

$$C^*_{max} \geq p_l + p_j$$

As actual processing time of a task must be greater than $\frac{1}{\alpha}$ times of its estimated value, we have

$$C^*_{max} \geq \frac{1}{\alpha}\tilde{p}_l + \frac{1}{\alpha}\tilde{p}_j$$

As j is scheduled before l using LPT on estimated values of processing times, $\tilde{p}_j \geq \tilde{p}_l$ holds true for tasks l and j. Using this, we have

$$C^*_{max} \geq \frac{2}{\alpha}\tilde{p}_l$$

$$\Rightarrow C^*_{max} \geq \frac{2}{\alpha^2} p_l \tag{5}$$

Theorem 2.1: $\frac{C_{max}}{C^*_{max}} \leq 1 + (\frac{m-1}{m})\frac{\alpha^2}{2}$

Proof: The optimal makespan, C^*_{max} must be at least equal to the average load on the m machines. We have

$$C^*_{max} \geq \frac{\sum p_j}{m} \tag{6}$$

By the property of LPT the load on each machine i is greater than the load on the machine which reach C_{max} before the last task l is scheduled. So for each machine i, $C_{max} \leq \sum_{j \in E_i} p_j + p_l$ holds true. Summing for all the machines we have

$$mC_{max} \leq \sum p_j + (m-1)p_l$$

$$C_{max} \leq \frac{\sum p_j}{m} + \frac{(m-1)}{m} p_l \tag{7}$$

Using (6) and (7), we have

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{m-1}{m}\left(\frac{p_l}{C^*_{max}}\right)$$

As $C^*_{max} \geq \frac{2}{\alpha^2} p_l$, We have

$$\frac{C_{max}}{C^*_{max}} \leq 1 + \left(\frac{m-1}{m}\right)\frac{\alpha^2}{2}$$

Hence the theorem follows.

VI. Model 3: Replication in groups

Figure 2 shows the constrution of two phases in model 3. In this model the first phase is in offline mode and each task is pre-assigned to a particular group of processors. In the second phase the tasks are scheduled within the group they are assigned to in first phase. We have a set J of n jobs. There are k groups and size of each group is equal and have m/k processors within each group. We have considered task allocation in a group such that each task can be assigned to only one group, i.e. $\forall j$, $|M_j| = m/k$. In phase 2 each task is scheduled to a particular processor within the group it was allocated in phase 1.

We propose an two phase algorithm, **LS-Group** which is based on Graham's List Scheduling algorithm. In phase 1, using LS in offline mode the algorithm pre-assign the jobs in different groups. In phase 2, the algorithm chooses online LS to schedule tasks to processors within each group.

Theorem 3.1: When the number of groups is k the approximation ratio is $\frac{k\alpha^2}{\alpha^2+k-1}[1 + \frac{k-1}{m}] + \frac{m-k}{m}$

Proof: We have k groups of m/k machines. We have restriction that each task can be assigned to only one of these groups. In Phase 1, each task is assigned to different groups. In Phase 2, tasks are scheduled online within repective group.

Let us consider that C_{max} comes from group $G1$. Also, taking the property of List Scheduling that the load difference between any two groups cannot be greater than the largest task. So, for any group $Gl \neq G1$, We have

$$|\sum_{i \in G1} \tilde{p}_i - \sum_{i \in Gl} \tilde{p}_i| \leq max_{i \in T} \tilde{p}_i$$

for all, $l = 2, 3, ..., k$

Adding for all values of l, We have

$$|(k-1)\sum_{i \in G1} \tilde{p}_i - \sum_{l=2}^{k} \sum_{i \in Gl} \tilde{p}_i| \leq (k-1) max_{i \in T} \tilde{p}_i$$

$$\Rightarrow \sum_{l=2}^{k} \sum_{i \in Gl} \tilde{p}_i \geq (k-1)[\sum_{i \in G1} \tilde{p}_i - max_{i \in T} \tilde{p}_i]$$

As the actual processing time of tasks can vary within a factor α and $\frac{1}{\alpha}$ of their estimated processing time, the following inequality holds

$$\alpha \sum_{l=2}^{k} \sum_{i \in Gl} p_i \geq (k-1)[\frac{1}{\alpha} \sum_{i \in G1} p_i - \alpha max_{i \in T} p_i]$$

$$\sum_{l=2}^{k} \sum_{i \in Gl} p_i \geq (k-1)[\frac{1}{\alpha^2} \sum_{i \in G1} p_i - max_{i \in T} p_i] \tag{8}$$

In phase 2, We are applying LS on online mode. We assume that C_{max} comes from $G1$. Using LS property we can write,

$$C_{max} \leq \frac{\sum_{i \in G1} p_i}{m/k} + \frac{m/k - 1}{m/k} p_{max} \tag{9}$$

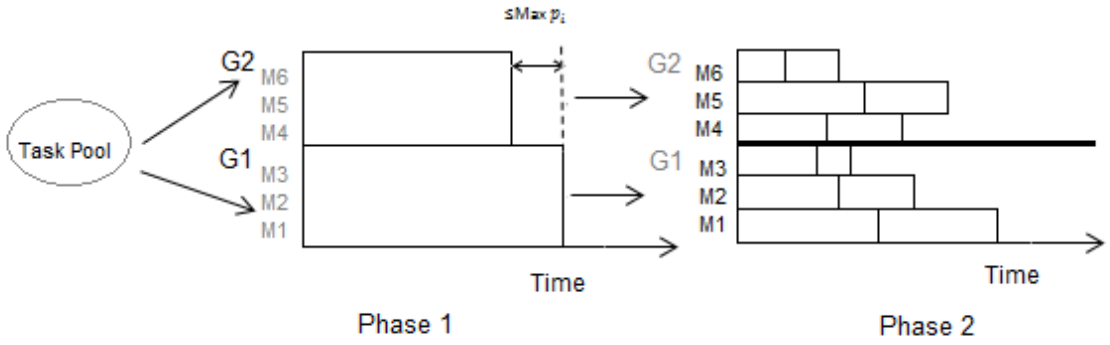

Figure 2. Replication in groups with $m = 6$, $k = 2$. The left fig. shows tasks assignment to groups in phase 1; The right fig shows each task assigned to a machine within a group

Also, C^*_{max} must be greater than the average of the loads on machines.

$$C^*_{max} \geq \frac{\sum_{i \in T} p_i}{m}$$

$$\Rightarrow C^*_{max} \geq \frac{\sum_{i \in G1} p_i + \sum_{l=2}^{k} \sum_{i \in Gl} p_i}{m}$$

from (11), we derive

$$\Rightarrow C^*_{max} \geq \frac{\sum_{i \in G1} p_i + (k-1)\left[\frac{1}{\alpha^2} \sum_{i \in G1} p_i - max_{i \in T} p_i\right]}{m}$$

$$\Rightarrow m\alpha^2 C^*_{max} + \alpha^2(k-1) max_{i \in T} p_i \geq \alpha^2 \sum_{i \in G1} p_i + (k-1) \sum_{i \in G1} p_i$$

$$\Rightarrow \frac{\alpha^2}{\alpha^2 + k - 1} \left[m C^*_{max} + (k-1) max_{i \in T} p_i\right] \geq \sum_{i \in G1} p_i \quad (10)$$

Using (12) and (13), We have

$$C_{max} \leq \frac{k\alpha^2}{\alpha^2 + k - 1} \left[C^*_{max} + \frac{(k-1)}{m} max_{i \in T} p_i\right] + \frac{m/k - 1}{m/k} p_{max}$$

As $C^*_{max} \geq max_{i \in T} p_i \geq p_{max}$, we have

$$C_{max} \leq \frac{k\alpha^2}{\alpha^2 + k - 1} \left[C^*_{max} + \frac{k-1}{m} C^*_{max}\right] + \frac{m-k}{m} C^*_{max}$$

So we have approximation ratio,

$$\frac{C_{max}}{C^*_{max}} \leq \frac{k\alpha^2}{\alpha^2 + k - 1} \left[1 + \frac{k-1}{m}\right] + \frac{m-k}{m}$$

Corollary 3.1: When the number of groups is 2 the approximation ratio is $1 + \frac{2}{1+\alpha^2}(\alpha^2 - \frac{1}{m})$

VII. Summary

The Table I summarizes the results in term of approximation ratio given by the three models.

The figure **??** shows ratio- replication graphs for each of the models for different values of α. We vary the value of α while keeping the number of machines fixed, $m = 210$. For replication everywhere scenario the number of replication is full and is always equal to total number of machines. For no replication scenario the approximation ratio is fixed for each value of α. For the model where replication is allowed within the group the approximation ratio decreases significantly for small replication. We observe that after a significant number of replications the ratio is not much improved further.

bibtex test [1]

References

[1] M. Gatto and P. Widmayer. On the robustness of graham's algorithm for online scheduling. In *Proc of WADS*, 2007.

Replication	Approximation ratio
$\|M_j\| = 1$	$\frac{C_{max}}{C^*_{max}} \leq \frac{2\alpha^2 m}{2\alpha^2+m-1}$ [Th. 1.2] No approximation better than α^2 [Th. 1.1]
$\|M_j\| = \|M\|$	$\frac{C_{max}}{C^*_{max}} \leq 1 + (\frac{m-1}{m})\frac{\alpha^2}{2}$ [Th. 2.1]
$\|M_j\| = m/k$	$\frac{C_{max}}{C^*_{max}} \leq \frac{k\alpha^2}{\alpha^2+k-1}\left[1 + \frac{k-1}{m}\right] + \frac{m-k}{m}$ [Th. 3.1] $\frac{C_{max}}{C^*_{max}} \leq 1 + \frac{2}{1+\alpha^2}\left(\alpha^2 - \frac{1}{m}\right)$ when $k = 2$ [Col. 3.1]

Table I
SUMMARY TABLE

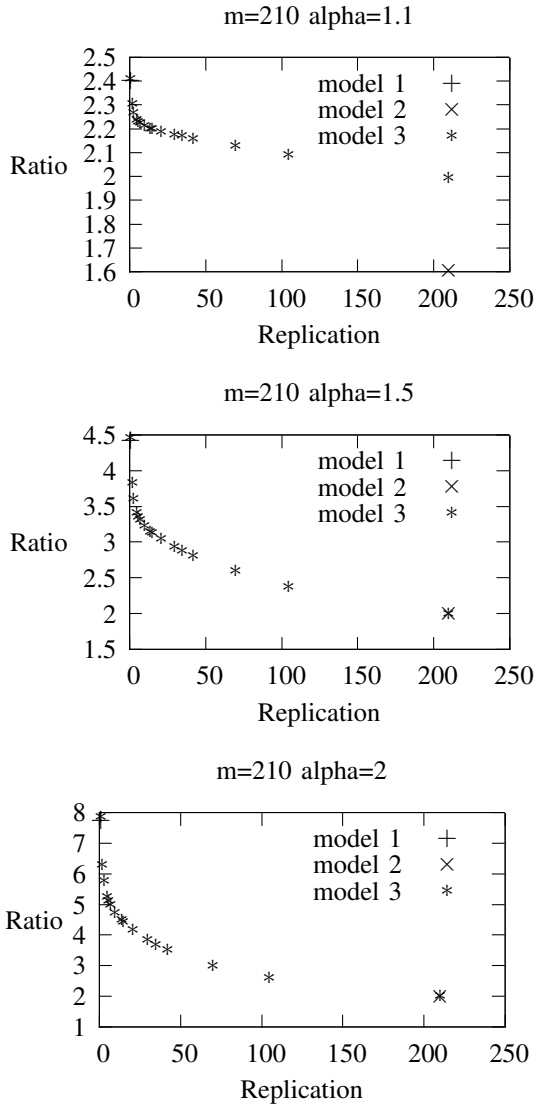

Figure 3. Ratio-Replication graph with m=210 and value of α as 1.1, 1.5 and 2