Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Evaluating the Performance of the Best Job First CPU Scheduling Algorithm with a Markov Chain Model in a Multiprocessor Environment

Rupesh Sendre, Rahul Singhai, Pradeep Kumar Jatav
International Institute of Professional Studies (IIPS), Devi Ahilya University, Indore (M.P.)

Abstract

CPU scheduling addresses the challenge of selecting which job from the ready queue should be dispatched by the dispatcher to the CPU, aiming to enhance resource utilization and overall system performance. Numerous traditional CPU scheduling algorithms have been proposed, each with its own advantages and disadvantages. Among these, the Best Job First CPU scheduling algorithm, introduced by Mohammed et al. [21], stands out. This paper evaluates the performance of this algorithm using a Markov Chain model in a multiprocessor environment. The study involves modeling the transitions between jobs using Markovian principles, and assessing the system's performance in terms of resource utilization, response time, and throughput. A comparative analysis is conducted to substantiate the findings, and numerical illustrations are provided to support the simulation study.

Keywords – CPU Scheduling; Best Job First Algorithm; Multiprocessor; Markov Chain Analysis;

1. Introduction

Scheduling algorithms are used for distributing resources among users which simultaneously and asynchronously request them. These algorithms are used in various systems such as operating systems (to allocate processor time among queues and jobs), communications networks (routers handling packet traffic), disk drives (I/O scheduling), printers (print spooling), and real-time systems. The primary goals of scheduling algorithms are to minimize resource starvation, ensure fairness among resource users, and maximize processor utilization by efficiently switching between jobs. Scheduling addresses the problem of determining which outstanding requests should be allocated resources.

CPU scheduling algorithms help the dispatcher decide which ready-state process should be assigned to the CPU next, aiming to enhance resource utilization and minimize queuing delays. Various scheduling algorithms have been developed, proving effective in both uniprocessing and multiprocessing systems. Multiprocessing systems, in particular, offer improved performance in terms of response time, throughput, waiting time, and turnaround time, especially in scenarios where uniprocessing systems fall short. Therefore, it is crucial to analyze the performance of scheduling algorithms in multiprocessing environments as well.

In this study, we implement the traditional Best Job First (BJF) scheduling algorithm in a multiprocessing environment. In conventional uniprocessing systems, each job receives an equal amount of processor time, and the dispatcher ensures proportional service time for each queue based on its quantum. However, if the time quantum is too small, excessive context switching may occur, and I/O-

bound jobs might not receive adequate processor time. To address these issues, we implemented BJF scheduling for multiprocessors with a relatively large time quantum.

To demonstrate the effectiveness of Multiprocessor Best Job First (MPBJF) scheduling, we modeled the scheduler transitions across two processors using a stochastic process. Our results indicate that MPBJF scheduling achieves precise proportional processing and high performance across diverse data sets [7]. Key observations from implementing the BJF scheduling algorithm for multiprocessor systems include:

- **Accuracy**: Utilizing the Markov chain model [18-19, 24-25], MPBJF scheduling achieves accurate proportional fairness with a low error rate, regardless of the number of queues and processors in the system.
- **Efficiency and Scalability**: MPBJF scheduling runs jobs per processor and adds minimal overhead to the existing operating system scheduler, even when jobs dynamically arrive, depart, or change their time quantum.
- **Flexible User Control**: MPBJF scheduling assigns a default time quantum to each queue based on its priority and allows users to specify job time quantum to control dispatcher transitions.
- **High Performance**: MPBJF scheduling integrates well with existing scheduler schemes targeting attribute such as latency and throughput, maintaining high performance and accurate fairness [7, 17].

These features demonstrate the versatility and efficiency of MPBJF scheduling in enhancing the performance of multiprocessor systems.

2. Related Work

Designing effective and efficient ready queue processing in a multiprocessor environment has always been a significant area of research. Numerous enhancements to various CPU scheduling algorithms have been proposed to evaluate their performance. For instance, Shukla and Jain [1] estimated ready queue processing under a new CPU scheduling algorithm in a multiprocessor environment with varying time quantum. Their combined study of lottery scheduling and systematic lottery scheduling proved efficient through model-based studies with numerical illustrations. Shukla et al. [10] proposed a lottery scheduling procedure in a multiprocessor environment where process size and auxiliary information were positively correlated. Additionally, Shukla and Jain [11] showed that a size-based priority scheme for predicting ready queue time length outperformed the traditional lottery scheduling scheme in terms of confidence intervals. In another study [2], the authors provided a general estimate of ready queue processing in a multiprocessor environment and derived confidence intervals to compare the efficiency of the estimate. Tam et al. [3] suggested shared memory multiprocessors with cache sharing within a chipset, introducing multiple processing chips configured as SMP, CMP, and SMT with non-uniform data sharing operating system schedulers. Levin et al. [4] developed the DP Fair scheduling policy to address the shortcomings of greedy scheduling algorithms. Bertogna and Cirinei [5] proposed a new approach for analyzing real-time systems globally scheduled on a symmetric multiprocessor platform, demonstrating its effectiveness through mathematical formulations and numerical illustrations.

Fairness is a fundamental requirement for any CPU scheduler. Existing round-robin scheduling algorithms are often inaccurate, inefficient, and non-scalable for multiprocessors, a problem exacerbated by the trend towards larger-scale multi-core processors. Li et al. [7] introduced a distributed weighted round-robin scheduling algorithm to address these issues, achieving accurate proportional processing and high performance across diverse data sets, supported by mathematical formulations and numerical experiments. Fedorova et al. [8] described a new operating system scheduling algorithm that improves performance isolation on chip multiprocessors, ensuring applications run efficiently under specific thread allocations. They implemented this cache-fair algorithm in Solaris™ 10, showing improved performance for SPEC CPU, SPEC JBB, and TPC-C benchmarks, supported by comparative studies. Elliott and Anderson [17] explored the significant performance advantages GPUs can offer over traditional CPUs. Their survey on real-time systems integrating GPUs into multiprocessor environments presented an integrated soft real-time multiprocessor system, demonstrating higher system performance through mathematical formulations and numerical illustrations. Burns et al. [9] introduced an EDF-based task-splitting scheme for scheduling multiprocessor systems, comparing it with two other schemes and generating numerical results to maximize processor utilization. Davis and Burns [14] surveyed hard real-time scheduling algorithms and schedulability analysis techniques for homogeneous multiprocessor systems, reviewing different scheduling methods and performance metrics for comparison.

Vijayalakshmi and Padmavathi [15] compared genetic algorithms and list scheduling algorithms within a multiprocessor task scheduling environment, generating experimental results that addressed multiprocessor scheduling problems. Li and Baruah [12] proposed inter-processor migration for scheduling mixed-criticality implicit-deadline sporadic task systems on identical multiprocessor platforms, demonstrating their effectiveness through theoretical and mathematical experiments. Cheramy et al. [13] developed a simulator for comparing and understanding real-time multiprocessor scheduling policies, generating data sets for simulations and collecting experimental data. Chandra et al. [16] introduced a novel weight readjustment algorithm to translate infeasible weight assignments to feasible ones, presenting surplus fair scheduling—a proportional share CPU scheduler designed for symmetric multiprocessors. They implemented this scheduler in the Linux kernel, demonstrating its efficacy through experimental evaluation. Shukla and Ojha [20] presented a Markov chain model to study transition states and designed various scheduling schemes treated as specific cases, comparing them under a Markov chain model setup and evaluating their merits through simulation studies. The Improved Round Robin (IRR) policy reduces average waiting time, increases throughput, and maintains CPU utilization levels similar to traditional Round Robin scheduling. Sendre et al. [6] used a Markov chain model to determine the performance of the IRR algorithm, proposing efficient and useful scheduling methods supported by numerical studies.

The set of possible values of an individual random variable $X^{(n)}$ *(or X(t))* of a stochastic process $\{X^{(n)}, n \geq 1\}$, $\{X(t), t \in T\}$ is known as state space, The stochastic process $\{X^{(n)}, n = 0,1,2\ldots\}$ is called Markov chain, if, for $j, k, j1, \ldots, j(n-1) \in N$ (or any subset of *I*). Medhi conducted extensive studies on a variety of stochastic processes and their applications in different fields. He developed a Markov chain model to study uncertain rainfall phenomena and demonstrated the use of stochastic processes in queue management [22-23]. Naldi proposed and developed a Markov chain model to understand internet traffic

sharing among various operators in a competitive market [19]. Another researcher, Jain & Jain also proposed a linear data model-based study of an improved round-robin (RR) CPU scheduling algorithm, incorporating features of the shortest job first (SJF) scheduling with varying time quantum. This study used a Markov chain model with different datasets and included numerical analyses [25]. Sendre et al. also studied the use of Markov chain models on a variety of stochastic processes and their applications in different multilevel queue schedulers, designing a scheduling scheme that was compared through numerical studies [24].

Awad et al. developing efficient scheduling and allocation methods is essential for optimizing various performance metrics. A notable approach is the Load Balancing Mutation Particle Swarm Optimization (LBMPSO) model, which addresses reliability, execution time, transmission time, makespan, round trip time, transmission cost, and load balancing between tasks and virtual machines. LBMPSO enhances the reliability of the cloud computing environment by leveraging available resources and rescheduling tasks that fail to allocate initially. Comparative studies have shown that LBMPSO outperforms standard Particle Swarm Optimization (PSO), random algorithms, and the Longest Cloudlet to Fastest Processor (LCFP) algorithm in terms of reducing makespan, execution time, round trip time, and transmission cost [26]. Shyam & Nandal development of a new Round Robin scheduling algorithm has demonstrated significant improvements over existing algorithms such as traditional Round Robin (RR), Improved Round Robin (IRR), Enhanced Round Robin (ERR), Self-Adjustment Round Robin (SARR), and First-Come, First-Served (FCFS). This new scheduling method offers superior performance, highlighting its effectiveness in optimizing scheduling tasks compared to these well-known algorithms [27]. Another researcher, Bitam et al. addressing the job scheduling problem within fog computing environments, a novel bio-inspired optimization approach known as the Bees Life Algorithm (BLA) has been introduced. This method focuses on optimizing the distribution of tasks across all fog computing nodes, aiming to achieve an optimal balance between CPU execution time and the memory allocated for services used by mobile users. Empirical performance evaluations have shown that the Bees Life Algorithm outperforms traditional methods such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) regarding CPU execution time and memory allocation [28]. Another researcher, Gawali & Shinde, also proposed system incorporates LEPT preemption to preempt resource-intensive tasks effectively. This approach is enhanced through a divide-and-conquer strategy, which has been experimentally validated by comparing it with existing frameworks like BATS (Bidirectional Arrival and Time-based Scheduler) and Improved Differential Evolution Algorithm (IDEA). The evaluation focuses on metrics such as turnaround time and response time, demonstrating the superiority of the proposed system in handling these performance metrics effectively [29].

In this study, a novel approach named Harris Hawks Optimization based on Simulated Annealing (HHOSA) is introduced for job scheduling in cloud environments. The HHOSA method integrates Simulated Annealing (SA) as a local search technique within the Harris Hawks Optimization (HHO) framework to enhance solution quality and convergence speed compared to the standard HHO algorithm. Performance evaluation of HHOSA is conducted against state-of-the-art job scheduling algorithms using the CloudSim toolkit, employing both standard and synthetic workloads. The results highlight that HHOSA significantly reduces the makespan in job scheduling compared to standard HHO and other existing algorithms. Furthermore, HHOSA demonstrates faster convergence particularly in larger search spaces, making it well-suited for addressing large-scale scheduling challenges [30].

Another researcher, Gupta et al. enhanced versions of the Heterogeneous Earliest Finish Time (HEFT) algorithm are proposed to minimize the makespan of workflow submissions on virtual machines while adhering to user-defined financial constraints. The study highlights that these enhanced HEFT variants outperform the basic HEFT algorithm by achieving shorter schedule lengths for workflows executed across different virtual machines. This improvement underscores the efficacy of adapting HEFT to meet specific financial constraints while optimizing workflow scheduling in cloud computing environments [31]. Another researcher, Alsaidy et al. proposed initialization method for Particle Swarm Optimization (PSO) is introduced, utilizing heuristic algorithms such as Longest Job to Fastest Processor (LJFP) and Minimum Completion Time (MCT). These algorithms are employed to initialize the PSO framework, aiming to improve its performance in minimizing metrics such as makespan, total execution time, degree of imbalance, and total energy consumption in task scheduling scenarios. The effectiveness of the proposed LJFP-PSO and MCT-PSO algorithms is evaluated and compared against recent task scheduling methods through simulations. Results demonstrate that LJFP-PSO and MCT-PSO achieve superior performance compared to conventional PSO and other comparative algorithms, highlighting their efficacy in optimizing task scheduling processes [32]. Another researcher, Ullah et al. also focus is on enhancing makespan optimization to minimize unproductive time for machinery and tasks in an Electronics manufacturing facility. The research employs a case study methodology centered on job scheduling, emphasizing resource availability. Specifically, Johnson's algorithm and its variations for two-machine and three-machine flow shop scheduling scenarios are implemented to identify optimal scheduling sequences. The investigation evaluates idle time and makespan metrics for individual machines using task processing durations and in-out timestamps. Results indicate optimal idle times of 6.21 minutes and a makespan of 142.06 minutes for two-machine scenarios. This research offers valuable insights applicable to industries with diverse machinery and components, contributing significantly to scheduling efficiency and overall productivity improvements [33].

3. Proposed System

In a uniprocessor environment, jobs are assigned to processors based on a specific CPU scheduling algorithm, such as First-Come-First-Serve (FCFS) or preemptively based on a time-quantum in algorithms like Best Job First scheduling. However, in a multiprocessing setup, jobs are dispatched in a Best Job First fashion, but the selection of which processor receives the job is dynamic. This paper examines the performance of the Best Job First scheduling algorithm using a Markov chain model in a multiprocessing environment with two processors, P_1 and P_2, each handling a large number of randomly assigned jobs.

We make the following assumptions to model job allocation between P_1 and P_2:

- All ready jobs reside in a ready queue, and the CPU scheduler selects jobs based on the Best Job First strategy and assigns them randomly to any available processor.
- Initially, each job has an equal probability (P_{r1} and P_{r2}) of being assigned to either processor P_{r1} and P_{r2} ($\sum_{i=1}^{2} \mathrm{Pri} = 1$).
- Jobs are initially assigned P_i(i = 1, 2) with a dynamic priority and a fixed timer interrupts job execution after a predefined time-quantum.

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

- A job holds the processor until its time quantum expires. If the job is not completed, it returns to the end of the ready queue; if completed, it exits the processor.
- Processor allocation continues until the ready queue is empty.
- Processors may be assigned jobs consecutively (P_1, P_2, P_1...) or in an alternating pattern.
- If both processors are free, they move to a resting state; if both are busy, they remain in a busy state.
- If both processors are ready and new jobs arrive in the ready queue, they can be assigned to either processor, changing their state from R (ready) to either P_1 or P_2.
- When processors become available, their state changes from B (busy) to either P_1 or P_2.

This study utilizes these assumptions to analyze the efficiency of the Best Job First scheduling algorithm in a multiprocessing environment using a Markov chain model.

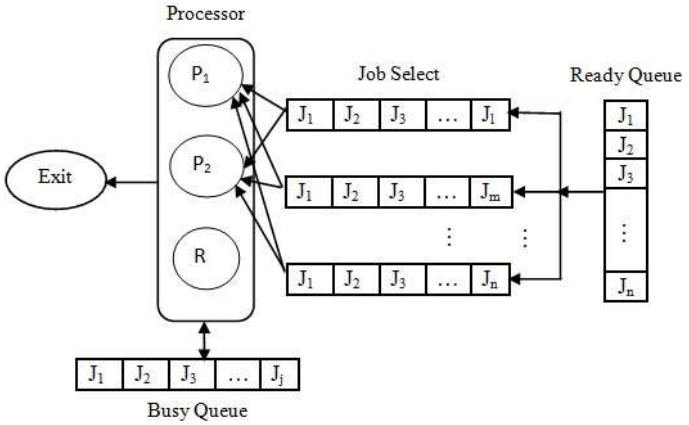


Figure 3.1: Generalized markov chain models in multiprocessor environment.

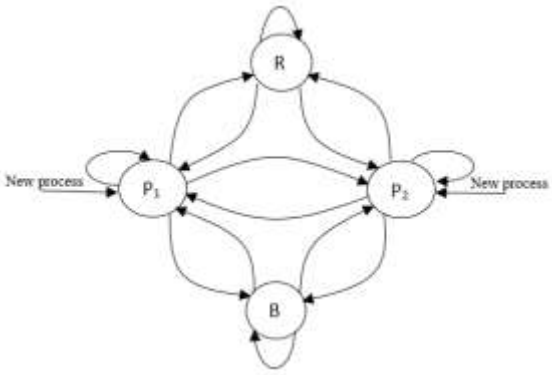


Figure 3.2: Generalized transition probability model.

Under their assumption the behavior of processors and action of scheduler can be modeled by core state discrete time Markov chain (fig. 3.2) in which transition probabilities are represented by an edge

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

connecting the circulars indicating the different chain states and the time is indicated by number of attempts.

Thus, the initial condition before the first allotment of processors are:

$P [X^{(0)} = P_1] = P_{r1}$; $P [X^{(0)} = P_2] = P_{r2}$; $P [X^{(0)} = B] = 0$; $P [X^{(0)} = R] = 1 - \sum_{i=1}^{2} Pri.$] …… eq. 1

Therefore, the one step transition probabilities matrix is as follows:

$P =$ ($X^{(n-1)}$ ↑ ↓, $X^{(n)}$ ←→)

	P_1	P_2	B	R
P_1	T_{11}	T_{12}	T_{13}	T_{14}
P_2	T_{21}	T_{22}	T_{23}	T_{24}
B	T_{31}	T_{32}	T_{33}	0
R	T_{41}	T_{42}	0	T_{44}

Let T_{ij} $(i, j = 1, 2, 3,…)$ be the unit step transition probabilities of scheduler over three states then transition probability depend on subject to condition:

$T_{14} = (1 - \sum_{i=1}^{3} T1i)$; $T_{24} = (1 - \sum_{i=1}^{3} T2i)$; $T_{34} = (1 - \sum_{i=1}^{3} T3i)$; $T_{44} = (1 - \sum_{i=1}^{3} T4i)$; & $0 \leq T_{ij} \leq 1$,

The state probabilities, after the first time quantum can be obtained by a simple relationship:

$P [X^{(1)} = P_1] = P [X^{(0)} = P_1] P [X^{(1)} = P_1 / X^{(0)} = P_1] + P [X^{(0)} = P_2] P [X^{(1)} = P_1 / X^{(0)} = P_2] + P [X^{(0)} = B] P [X^{(1)} = P_1 / X^{(0)} = B] + P [X^{(0)} = R] P [X^{(1)} = P_1 / X^{(0)} = R]$

$P [X^{(1)} = P_1] = \sum_{i=1}^{3} Pri\ Ti1$; $P [X^{(1)} = P_2] = \sum_{i=1}^{3} Pri\ Ti2$;

$P [X^{(1)} = B] = \sum_{i=1}^{3} Pri\ Ti3$; $P [X^{(1)} = R] = \sum_{i=1}^{3} Pri\ Ti4$] ……eq. 2

Similarly, state probabilities after second time quantum can be obtained by simple relationship:

$P [X^{(2)} = P_1] = P [X^{(1)} = P_1] P [X^{(2)} = P_1 / X^{(1)} = P_1] + P [X^{(1)} = P_2] P [X^{(2)} = P_1 / X^{(1)} = P_2] + P [X^{(1)} = B] P [X^{(2)} = P_1 / X^{(1)} = B] + P [X^{(1)} = R] P [X^{(2)} = P_1 / X^{(1)} = R]$

$P [X^{(2)} = P_1] = \sum_{i=1}^{4} (\sum_{j=1}^{3} Prj\ Tji)\ T_{i1}$; $P [X^{(2)} = P_2] = \sum_{i=1}^{4} (\sum_{j=1}^{3} Prj\ Tji)\ T_{i2}$;

$P [X^{(2)} = B] = \sum_{i=1}^{4} (\sum_{j=1}^{3} Prj\ Tji)\ T_{i3}$; $P [X^{(2)} = R] = \sum_{i=1}^{4} (\sum_{j=1}^{3} Prj\ Tji)\ T_{i4}$] .....eq. 3

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

The generalized expressions for n time quantum are:

$$\left.\begin{aligned} P[X^{(n)} = P_1] &= \sum_{m=1}^{4} \cdots\cdots \sum_{l=1}^{4} \sum_{k=1}^{4} \sum_{i=1}^{4} \sum_{j=1}^{3} Prj\ Tji\ Tik\ Tkl\ \ldots\ldots Tm1; \\ P[X^{(n)} = P_2] &= \sum_{m=1}^{4} \cdots\cdots \sum_{l=1}^{4} \sum_{k=1}^{4} \sum_{i=1}^{4} \sum_{j=1}^{3} Prj\ Tji\ Tik\ Tkl\ \ldots\ldots Tm2; \\ P[X^{(n)} = B] &= \sum_{m=1}^{4} \cdots\cdots \sum_{l=1}^{4} \sum_{k=1}^{4} \sum_{i=1}^{4} \sum_{j=1}^{3} Prj\ Tji\ Tik\ Tkl\ \ldots\ldots Tm3; \\ P[X^{(n)} = R] &= \sum_{m=1}^{4} \cdots\cdots \sum_{l=1}^{4} \sum_{k=1}^{4} \sum_{i=1}^{4} \sum_{j=1}^{3} Prj\ Tji\ Tik\ Tkl\ \ldots\ldots Tm4 \end{aligned}\right\} \quad \text{.....eq. 4}$$

4. BJF CPU Scheduling Schemes Under Multiprocessing Environment

In this section, we have discussed few scheduling schemes that may be produced from above mentioned generalized MPBJF scheduling model by imposing some restrictions and condition. The three schemes realized are as follows:

4.1 Scheme - I: When job may be assigned to either the first processor P_1 or second processor P_2 only. New job joins from ready queue from tail and the oldest job is dispatched to any of two processor P_1 or P_2 for execution. Similarly, the other jobs are selected from ready queue in FCFS order is assigned randomly to either processor P_1 or P_2. Thus, the assignment of processor for jobs is random.

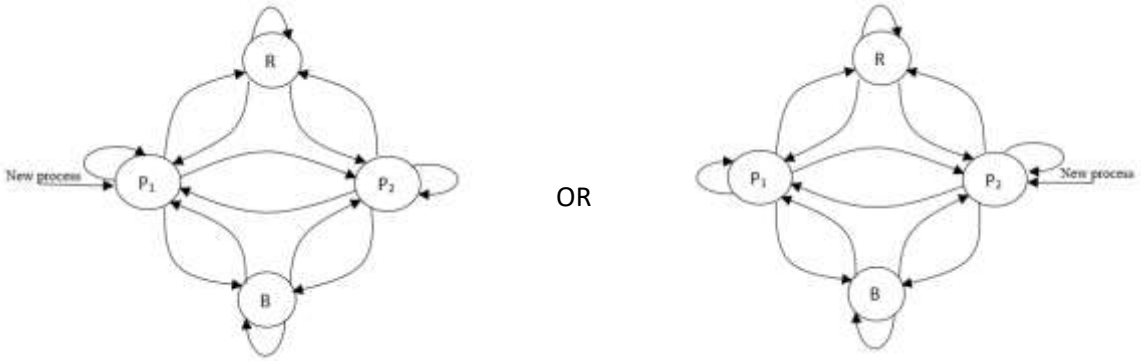


Figure 4.1: Restricted transition probability diagram

Thus, the initial probabilities under scheme-I are:

$P[X^{(0)} = P_1] = 1$; $P[X^{(0)} = P_2] = 0$; $P[X^{(0)} = B] = 0$; $P[X^{(0)} = R] = 0$

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Unit step transaction probability matrix for $X^{(n)}$ under scheme-I is:

$P = \begin{matrix} \\ X^{(n-1)} \\ \\ \end{matrix} \begin{matrix} \longleftarrow X^{(n)} \longrightarrow \\ \end{matrix}$

	P_1	P_2	B	R
P_1	T_{11}	T_{12}	T_{13}	T_{14}
P_2	T_{21}	T_{22}	T_{23}	T_{24}
B	T_{31}	T_{32}	T_{33}	0
R	T_{41}	T_{42}	0	T_{44}

By using eq. 2 the state probabilities after the first time quantum are:

$P [X^{(1)} = P_1] = T_{11}$; $P [X^{(1)} = P_2] = T_{12}$; $P [X^{(1)} = B] = T_{13}$; $P [X^{(1)} = R] = T_{14}$

By using eq. 3 the state probabilities after the second time quantum are:

$P [X^{(2)} = P_1] = P [X^{(1)} = P_1] P [X^{(2)} = P_1 / X^{(1)} = P_1] + P [X^{(1)} = P_2] P [X^{(2)} = P_1 / X^{(1)} = P_2] + P [X^{(1)} = B] P [X^{(2)} = P_1 / X^{(1)} = B] + P [X^{(1)} = R] P [X^{(2)} = P_1 / X^{(1)} = R]$

$P [X^{(2)} = P_1] = T_{11}T_{11} + T_{12} T_{21} + T_{13} T_{31} + T_{14} T_{41}$

$P [X^{(2)} = P_2] = P [X^{(1)} = P_1] P [X^{(2)} = P_2 / X^{(1)} = P_1] + P [X^{(1)} = P_2] P [X^{(2)} = P_2 / X^{(1)} = P_2] + P [X^{(1)} = B] P [X^{(2)} = P_2 / X^{(1)} = B] + P [X^{(1)} = R] P [X^{(2)} = P_2 / X^{(1)} = R]$

$P [X^{(2)} = P_2] = T_{12} T_{21} + T_{22}T_{22} + T_{23} T_{32} + T_{24} T_{42}$

$P [X^{(2)} = B] = P [X^{(1)} = P_1] P [X^{(2)} = B / X^{(1)} = P_1] + P [X^{(1)} = P_2] P [X^{(2)} = B / X^{(1)} = P_2] + P [X^{(1)} = B] P [X^{(2)} = B / X^{(1)} = B] + P [X^{(1)} = R] P [X^{(2)} = B / X^{(1)} = R]$

$P [X^{(2)} = B] = T_{13} T_{31} + T_{23} T_{32} + T_{33}T_{33}$

$P [X^{(2)} = R] = P [X^{(1)} = P_1] P [X^{(2)} = R / X^{(1)} = P_1] + P [X^{(1)} = P_2] P [X^{(2)} = R / X^{(1)} = P_2] + P [X^{(1)} = B] P [X^{(2)} = R / X^{(1)} = B] + P [X^{(1)} = R] P [X^{(2)} = R / X^{(1)} = R]$

$P [X^{(2)} = R] = T_{14} T_{41} + T_{24} T_{42} + T_{44}T_{44}$

Similarly, third time quantum are:

$P [X^{(3)} = P_1] = (T_{11}T_{11} + T_{12} T_{21} + T_{13} T_{31} + T_{14} T_{41}) T_{11} + (T_{12} T_{21} + T_{22}T_{22} + T_{23} T_{32} + T_{24} T_{42}) T_{21} + (T_{13} T_{31} + T_{23} T_{32} + T_{33} T_{33}) T_{31} + (T_{14} T_{41} + T_{24} T_{42} + T_{44} T_{44}) T_{41}$

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

$P[X^{(3)} = P_2] = (T_{11}T_{11} + T_{12}T_{21} + T_{13}T_{31} + T_{14}T_{41})T_{12} + (T_{12}T_{21} + T_{22}T_{22} + T_{23}T_{32} + T_{24}T_{42})T_{22} + (T_{13}T_{31} + T_{23}T_{32} + T_{33}T_{33})T_{32} + (T_{14}T_{41} + T_{24}T_{42} + T_{44}T_{44})T_{42}$

$P[X^{(3)} = B] = (T_{11}T_{11} + T_{12}T_{21} + T_{13}T_{31} + T_{14}T_{41})T_{13} + (T_{12}T_{21} + T_{22}T_{22} + T_{23}T_{32} + T_{24}T_{42})T_{23} + (T_{13}T_{31} + T_{23}T_{32} + T_{33}T_{33})T_{33}$

$P[X^{(3)} = R] = (T_{11}T_{11} + T_{12}T_{21} + T_{13}T_{31} + T_{14}T_{41})T_{14} + (T_{12}T_{21} + T_{22}T_{22} + T_{23}T_{32} + T_{24}T_{42})T_{24} + (T_{14}T_{41} + T_{24}T_{42} + T_{44}T_{44})T_{44}$

Similarly, we can find fourth, fifth and so on time quantum.

4.2 Scheme - II: When processors assigned in alternative manner (i.e. P_1, P_2, P_1, …) The following transition are restricted in this scheme:

- A new job can only enter to first processor P1 only.
- Transition from processor P1 to P1 or P2 to P2 are restricted.

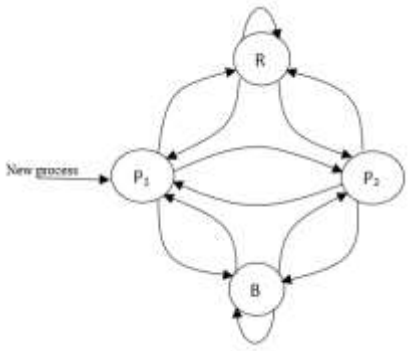


Figure 4.2: Restricted transition probability diagram.

Thus, the initial probabilities under scheme-II are:

$P[X^{(0)} = P_1] = 1$; $P[X^{(0)} = P_2] = 0$; $P[X^{(0)} = B] = 0$; $P[X^{(0)} = R] = 0$

Unit step transaction probability matrix for $X^{(n)}$ under scheme-II is:

$P =$ $X^{(n-1)}$ ↕ , $X^{(n)}$ →

	P_1	P_2	B	R
P_1	0	T_{12}	T_{13}	T_{14}
P_2	T_{21}	0	T_{23}	T_{24}
B	T_{31}	T_{32}	T_{33}	0
R	T_{41}	T_{42}	0	T_{44}

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

By using eq. 2 the state probabilities after the first time quantum are:

$P[X^{(2)} = P_1] = P[X^{(1)} = P_1] P[X^{(2)} = P_1 / X^{(1)} = P_1] + P[X^{(1)} = P_2] P[X^{(2)} = P_1 / X^{(1)} = P_2] + P[X^{(1)} = B] P[X^{(2)} = P_1 / X^{(1)} = B] + P[X^{(1)} = R] P[X^{(2)} = P_1 / X^{(1)} = R]$

$P[X^{(2)} = P_1] = T_{12} T_{21} + T_{13} T_{31} + T_{14} T_{41}$

$P[X^{(2)} = P_2] = P[X^{(1)} = P_1] P[X^{(2)} = P_2 / X^{(1)} = P_1] + P[X^{(1)} = P_2] P[X^{(2)} = P_2 / X^{(1)} = P_2] + P[X^{(1)} = B] P[X^{(2)} = P_2 / X^{(1)} = B] + P[X^{(1)} = R] P[X^{(2)} = P_2 / X^{(1)} = R]$

$P[X^{(2)} = P_2] = T_{12} T_{21} + T_{23} T_{32} + T_{24} T_{42}$

$P[X^{(2)} = B] = P[X^{(1)} = P_1] P[X^{(2)} = B / X^{(1)} = P_1] + P[X^{(1)} = P_2] P[X^{(2)} = B / X^{(1)} = P_2] + P[X^{(1)} = B] P[X^{(2)} = B / X^{(1)} = B] + P[X^{(1)} = R] P[X^{(2)} = B / X^{(1)} = R]$

$P[X^{(2)} = B] = T_{13} T_{31} + T_{23} T_{32} + T_{33} T_{33}$

$P[X^{(2)} = R] = P[X^{(1)} = P_1] P[X^{(2)} = R / X^{(1)} = P_1] + P[X^{(1)} = P_2] P[X^{(2)} = R / X^{(1)} = P_2] + P[X^{(1)} = B] P[X^{(2)} = R / X^{(1)} = B] + P[X^{(1)} = R] P[X^{(2)} = R / X^{(1)} = R]$

$P[X^{(2)} = R] = T_{14} T_{41} + T_{24} T_{42} + T_{44} T_{44}$

Similarly, third time quantum are:

$P[X^{(3)} = P_1] = (T_{12} T_{21} + T_{23} T_{32} + T_{24} T_{42}) T_{21} + (T_{13} T_{31} + T_{23} T_{32} + T_{33} T_{33}) T_{31} + (T_{14} T_{41} + T_{24} T_{42} + T_{44} T_{44}) T_{41}$

$P[X^{(3)} = P_2] = (T_{12} T_{21} + T_{13} T_{31} + T_{14} T_{41}) T_{12} + (T_{13} T_{31} + T_{23} T_{32} + T_{33} T_{33}) T_{32} + (T_{14} T_{41} + T_{24} T_{42} + T_{44} T_{44}) T_{42}$

$P[X^{(3)} = B] = (T_{12} T_{21} + T_{13} T_{31} + T_{14} T_{41}) T_{13} + (T_{12} T_{21} + T_{23} T_{32} + T_{24} T_{42}) T_{23} + (T_{13} T_{31} + T_{23} T_{32} + T_{33} T_{33}) T_{33}$

$P[X^{(3)} = R] = (T_{12} T_{21} + T_{13} T_{31} + T_{14} T_{41}) T_{14} + (T_{12} T_{21} + T_{23} T_{32} + T_{24} T_{42}) T_{24} + (T_{14} T_{41} + T_{24} T_{42} + T_{44} T_{44}) T_{44}$

Similarly, we can find fourth, fifth and so on time quantum.

4.3 Scheme - III: When some restriction is applied to control transition –

Transition from processors P_1 or P_2 to R is restricted as it is assuming that no processor can move to resting state until there exists at least one job in ready queue and if any of the processor become free then operating systems immediately assigns few jobs that are currently assign to another processor.

- Transition from state R to R or B to B is restricted.

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

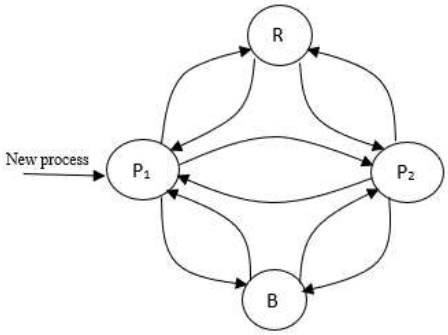


Figure 4.3: Restricted transition probability diagram.

Thus, the initial probabilities under scheme-III are:

$P[X^{(0)} = P_1] = 1$; $P[X^{(0)} = P_2] = 0$; $P[X^{(0)} = B] = 0$; $P[X^{(0)} = R] = 0$

Unit step transaction probability matrix for $X^{(n)}$ under scheme-3 is:

$P =$ ($X^{(n-1)}$ ↕, $X^{(n)}$ ↔)

	P_1	P_2	B	R
P_1	0	T_{12}	T_{13}	T_{14}
P_2	T_{21}	0	T_{23}	T_{24}
B	T_{31}	T_{32}	0	0
R	T_{41}	T_{42}	0	0

By using eq. 2 the state probabilities after the first time quantum are:

$P[X^{(1)} = P_1] = 0$; $P[X^{(1)} = P_2] = T_{12}$; $P[X^{(1)} = B] = T_{13}$; $P[X^{(1)} = R] = T_{14}$

By using eq. 3 the state probabilities after the second time quantum are:

$$P[X^{(2)} = P_1] = P[X^{(1)} = P_1]\, P[X^{(2)} = P_1 / X^{(1)} = P_1] + P[X^{(1)} = P_2]\, P[X^{(2)} = P_1 / X^{(1)} = P_2] + P[X^{(1)} = B]\, P[X^{(2)} = P_1 / X^{(1)} = B] + P[X^{(1)} = R]\, P[X^{(2)} = P_1 / X^{(1)} = R]$$

$$P[X^{(2)} = P_1] = T_{12} T_{21} + T_{13} T_{31} + T_{14} T_{41}$$

$$P[X^{(2)} = P_2] = P[X^{(1)} = P_1]\, P[X^{(2)} = P_2 / X^{(1)} = P_1] + P[X^{(1)} = P_2]\, P[X^{(2)} = P_2 / X^{(1)} = P_2] + P[X^{(1)} = B]\, P[X^{(2)} = P_2 / X^{(1)} = B] + P[X^{(1)} = R]\, P[X^{(2)} = P_2 / X^{(1)} = R]$$

$$P[X^{(2)} = P_2] = T_{12} T_{21} + T_{23} T_{32} + T_{24} T_{42}$$

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

$P[X^{(2)}=B]=P[X^{(1)}=P_1]\,P[X^{(2)}=B/X^{(1)}=P_1]+P[X^{(1)}=P_2]\,P[X^{(2)}=B/X^{(1)}=P_2]+P[X^{(1)}=B]\,P[X^{(2)}=B/X^{(1)}=B]+P[X^{(1)}=R]\,P[X^{(2)}=B/X^{(1)}=R]$

$P[X^{(2)}=B]=T_{13}T_{31}+T_{23}T_{32}$

$P[X^{(2)}=R]=P[X^{(1)}=P_1]\,P[X^{(2)}=R/X^{(1)}=P_1]+P[X^{(1)}=P_2]\,P[X^{(2)}=R/X^{(1)}=P_2]+P[X^{(1)}=B]\,P[X^{(2)}=R/X^{(1)}=B]+P[X^{(1)}=R]\,P[X^{(2)}=R/X^{(1)}=R]$

$P[X^{(2)}=R]=T_{14}T_{41}+T_{24}T_{42}$

Similarly, third time quantum are:

$P[X^{(3)}=P_1]=(T_{12}T_{21}+T_{23}T_{32}+T_{24}T_{42})T_{21}+(T_{13}T_{31}+T_{23}T_{32})T_{31}+(T_{14}T_{41}+T_{24}T_{42})T_{41}$

$P[X^{(3)}=P_2]=(T_{12}T_{21}+T_{13}T_{31}+T_{14}T_{41})T_{12}+(T_{13}T_{31}+T_{23}T_{32})T_{32}+(T_{14}T_{41}+T_{24}T_{42})T_{42}$

$P[X^{(3)}=B]=(T_{12}T_{21}+T_{13}T_{31}+T_{14}T_{41})T_{13}+(T_{12}T_{21}+T_{23}T_{32}+T_{24}T_{42})T_{23}$

$P[X^{(3)}=R]=(T_{12}T_{21}+T_{13}T_{31}+T_{14}T_{41})T_{14}+(T_{12}T_{21}+T_{23}T_{32}+T_{24}T_{42})T_{24}$

Similarly, we can find fourth, fifth and so on time quantum.

5. Simulation Study with Numerical Analysis Using Data Sets

In order to analyze three schemes mentioned in section 4.1, 4.2 and 4.3 under Markov chain model with varying quantum probability (random and linear) transition elements using different data sets are as follows:

5.1 Data set – I

Scheme I: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Random

P = $X^{(n-1)}$ \ $X^{(n)}$

	P_1	P_2	B	R
P_1	0.12	0.34	0.18	0.36
P_2	0.21	0.28	0.30	0.21
B	0.35	0.31	0.34	0
R	0.28	0.32	0	0.40

Linear

P = $X^{(n-1)}$ \ $X^{(n)}$

	P_1	P_2	B	R
P_1	0.10	0.10	0.10	0.70
P_2	0.10	0.10	0.10	0.70
B	0.10	0.10	0.80	0
R	0.10	0.10	0	0.80

Table 5.1.1: The transition probabilities $P[X^{(n)} = P_i]$ for random and linear cases:

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0.12	0.34	0.18	0.36	0.1	0.1	0.1	0.7
n = 2	0.25	0.307	0.185	0.259	0.1	0.1	0.1	0.7
n = 3	0.232	0.311	0.2	0.258	0.1	0.1	0.1	0.7
n = 4	0.235	0.311	0.203	0.252	0.1	0.1	0.1	0.7
n = 5	0.235	0.311	0.205	0.251	0.1	0.1	0.1	0.7
n = 6	0.236	0.311	0.205	0.25	0.1	0.1	0.1	0.7
n = 7	0.235	0.311	0.205	0.25	0.1	0.1	0.1	0.7
n = 8	0.235	0.311	0.205	0.25	0.1	0.1	0.1	0.7
n = 9	0.235	0.311	0.205	0.25	0.1	0.1	0.1	0.7
n = 10	0.235	0.311	0.205	0.25	0.1	0.1	0.1	0.7

Scheme II: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Random

$P = \begin{matrix} \\ X^{(n-1)} \end{matrix} \overset{X^{(n)}}{\longrightarrow}$

	P_1	P_2	B	R
P_1	0	0.34	0.18	0.48
P_2	0.21	0	0.30	0.49
B	0.35	0.31	0.34	0
R	0.28	0.32	0	0.40

Linear

$P = \begin{matrix} \\ X^{(n-1)} \end{matrix} \overset{X^{(n)}}{\longrightarrow}$

	P_1	P_2	B	R
P_1	0	0.10	0.10	0.80
P_2	0.10	0	0.10	0.80
B	0.10	0.10	0.80	0
R	0.10	0.10	0	0.80

Table 5.1.2: The transition probabilities P [$X^{(n)} = P_i$] for random and linear cases:

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0	0.34	0.18	0.48	0	0.1	0.1	0.8
n = 2	0.269	0.209	0.163	0.359	0.1	0.09	0.09	0.72
n = 3	0.201	0.257	0.167	0.375	0.09	0.091	0.091	0.728
n = 4	0.217	0.24	0.17	0.372	0.091	0.091	0.091	0.727
n = 5	0.214	0.246	0.169	0.371	0.091	0.091	0.091	0.727
n = 6	0.215	0.244	0.17	0.372	0.091	0.091	0.091	0.727
n = 7	0.215	0.245	0.17	0.372	0.091	0.091	0.091	0.727
n = 8	0.215	0.245	0.17	0.372	0.091	0.091	0.091	0.727
n = 9	0.215	0.245	0.17	0.372	0.091	0.091	0.091	0.727
n = 10	0.215	0.245	0.17	0.372	0.091	0.091	0.091	0.727

Scheme III: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Random

$P =$ $X^{(n-1)}$ ↓ ; $X^{(n)}$ →

	P_1	P_2	B	R
P_1	0	0.34	0.18	0.48
P_2	0.21	0	0.30	0.49
B	0.35	0.65	0	0
R	0.28	0.72	0	0

Linear

$P =$ $X^{(n-1)}$ ↓ ; $X^{(n)}$ →

	P_1	P_2	B	R
P_1	0	0.10	0.10	0.80
P_2	0.10	0	0.10	0.80
B	0.10	0.90	0	0
R	0.10	0.90	0	0

Table 5.1.3: The transition probabilities P [$X^{(n)} = P_i$] for random and linear cases:

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0	0.34	0.18	0.48	0	0.1	0.1	0.8
n = 2	0.269	0.463	0.102	0.167	0.1	0.81	0.01	0.08
n = 3	0.18	0.278	0.187	0.356	0.09	0.091	0.091	0.728
n = 4	0.224	0.439	0.116	0.223	0.091	0.746	0.018	0.145
n = 5	0.195	0.312	0.172	0.323	0.091	0.156	0.084	0.67
n = 6	0.216	0.411	0.129	0.246	0.091	0.688	0.025	0.198
n = 7	0.2	0.334	0.162	0.305	0.091	0.21	0.078	0.623
n = 8	0.212	0.393	0.136	0.26	0.091	0.64	0.03	0.241
n = 9	0.203	0.348	0.156	0.294	0.091	0.253	0.073	0.585
n = 10	0.21	0.382	0.141	0.268	0.091	0.601	0.034	0.275

5.2 Data set – II

Scheme I: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Random

$P =$ $X^{(n-1)}$ ↓ ; $X^{(n)}$ →

	P_1	P_2	B	R
P_1	0.33	0.24	0.21	0.22
P_2	0.30	0.28	0.27	0.15
B	0.38	0.34	0.28	0
R	0.23	0.36	0	0.41

Linear

$P =$ $X^{(n-1)}$ ↓ ; $X^{(n)}$ →

	P_1	P_2	B	R
P_1	0.20	0.20	0.20	0.40
P_2	0.20	0.20	0.20	0.40
B	0.20	0.20	0.60	0
R	0.20	0.20	0	0.60

Table 5.2.1: The transition probabilities P [$X^{(n)} = P_i$] for random and linear cases:

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0.33	0.24	0.21	0.22	0.2	0.2	0.2	0.4
n = 2	0.311	0.297	0.193	0.199	0.2	0.2	0.2	0.4
n = 3	0.311	0.295	0.2	0.195	0.2	0.2	0.2	0.4
n = 4	0.312	0.295	0.201	0.193	0.2	0.2	0.2	0.4
n = 5	0.312	0.295	0.201	0.192	0.2	0.2	0.2	0.4
n = 6	0.312	0.295	0.201	0.192	0.2	0.2	0.2	0.4
n = 7	0.312	0.295	0.201	0.192	0.2	0.2	0.2	0.4
n = 8	0.312	0.295	0.201	0.192	0.2	0.2	0.2	0.4
n = 9	0.312	0.295	0.201	0.192	0.2	0.2	0.2	0.4
n = 10	0.312	0.295	0.201	0.192	0.2	0.2	0.2	0.4

Scheme II: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Random

$P = X^{(n-1)} \downarrow \quad X^{(n)} \rightarrow$

	P_1	P_2	B	R
P_1	0	0.24	0.21	0.55
P_2	0.30	0	0.27	0.43
B	0.38	0.34	0.28	0
R	0.23	0.36	0	0.41

Linear

$P = X^{(n-1)} \downarrow \quad X^{(n)} \rightarrow$

	P_1	P_2	B	R
P_1	0	0.20	0.20	0.60
P_2	0.20	0	0.20	0.60
B	0.20	0.20	0.60	0
R	0.20	0.20	0	0.60

Table 5.2.2: The transition probabilities P $[X^{(n)} = P_i]$ for random and linear cases:

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0	0.24	0.21	0.55	0	0.2	0.2	0.6
n = 2	0.278	0.269	0.124	0.329	0.2	0.16	0.16	0.48
n = 3	0.203	0.227	0.166	0.403	0.16	0.168	0.168	0.504
n = 4	0.224	0.25	0.15	0.374	0.168	0.166	0.166	0.499
n = 5	0.218	0.239	0.157	0.384	0.166	0.167	0.166	0.5
n = 6	0.22	0.244	0.154	0.38	0.167	0.166	0.166	0.5
n = 7	0.219	0.242	0.155	0.382	0.166	0.167	0.166	0.5
n = 8	0.219	0.243	0.155	0.381	0.166	0.167	0.166	0.5
n = 9	0.219	0.242	0.155	0.381	0.166	0.167	0.166	0.5
n = 10	0.219	0.242	0.155	0.381	0.166	0.167	0.166	0.5

Scheme III: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Random

$P = X^{(n-1)} \downarrow \quad X^{(n)} \rightarrow$

	P_1	P_2	B	R
P_1	0	0.24	0.21	0.55
P_2	0.30	0	0.27	0.43
B	0.38	0.62	0	0
R	0.23	0.77	0	0

Linear

$P = X^{(n-1)} \downarrow \quad X^{(n)} \rightarrow$

	P_1	P_2	B	R
P_1	0	0.20	0.20	0.60
P_2	0.20	0	0.20	0.60
B	0.20	0.80	0	0
R	0.20	0.80	0	0

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Table 5.2.3: The transition probabilities P [$X^{(n)} = P_i$]for random and linear cases:

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0	0.24	0.21	0.55	0	0.2	0.2	0.6
n = 2	0.278	0.554	0.065	0.103	0.2	0.64	0.04	0.12
n = 3	0.215	0.186	0.208	0.391	0.16	0.168	0.168	0.504
n = 4	0.225	0.482	0.095	0.198	0.168	0.57	0.066	0.197
n = 5	0.226	0.265	0.177	0.331	0.167	0.244	0.148	0.443
n = 6	0.223	0.419	0.119	0.238	0.167	0.506	0.082	0.247
n = 7	0.226	0.311	0.16	0.303	0.167	0.297	0.135	0.404
n = 8	0.224	0.387	0.131	0.258	0.167	0.465	0.093	0.278
n = 9	0.225	0.334	0.152	0.29	0.167	0.33	0.126	0.379
n = 10	0.225	0.372	0.137	0.267	0.167	0.437	0.1	0.298

5.3 Data set – III

Scheme I: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Random

$$P = \begin{matrix} \\ X^{(n-1)} \end{matrix} \overset{X^{(n)}}{\longrightarrow} \begin{array}{c|cccc} & P_1 & P_2 & B & R \\ P_1 & 0.27 & 0.35 & 0.18 & 0.20 \\ P_2 & 0.40 & 0.25 & 0.18 & 0.17 \\ B & 0.45 & 0.32 & 0.23 & 0 \\ R & 0.38 & 0.40 & 0 & 0.22 \end{array}$$

Linear

$$P = \begin{matrix} \\ X^{(n-1)} \end{matrix} \overset{X^{(n)}}{\longrightarrow} \begin{array}{c|cccc} & P_1 & P_2 & B & R \\ P_1 & 0.30 & 0.30 & 0.30 & 0.10 \\ P_2 & 0.30 & 0.30 & 0.30 & 0.10 \\ B & 0.30 & 0.30 & 0.40 & 0 \\ R & 0.30 & 0.30 & 0 & 0.40 \end{array}$$

Table 5.3.1: The transition probabilities P [$X^{(n)} = P_i$] for random and linear cases:

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0.27	0.35	0.18	0.2	0.3	0.3	0.3	0.1
n = 2	0.37	0.32	0.153	0.158	0.3	0.3	0.3	0.1
n = 3	0.357	0.322	0.159	0.163	0.3	0.3	0.3	0.1
n = 4	0.359	0.322	0.159	0.162	0.3	0.3	0.3	0.1
n = 5	0.359	0.322	0.159	0.162	0.3	0.3	0.3	0.1
n = 6	0.359	0.322	0.159	0.162	0.3	0.3	0.3	0.1
n = 7	0.359	0.322	0.159	0.162	0.3	0.3	0.3	0.1
n = 8	0.359	0.322	0.159	0.162	0.3	0.3	0.3	0.1
n = 9	0.359	0.322	0.159	0.162	0.3	0.3	0.3	0.1
n = 10	0.359	0.322	0.159	0.162	0.3	0.3	0.3	0.1

Scheme II: Let initial probabilities are: $Pr_1 = 1$; $Pr_2 = 0$; $Pr_3 = 0$ and $Pr_4 = 0$

Consider data set of random and linear probabilities matrix are follows:

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Random

$X^{(n)}$

P = $X^{(n-1)}$

	P_1	P_2	B	R
P_1	0	0.35	0.18	0.47
P_2	0.40	0	0.18	0.42
B	0.45	0.32	0.23	0
R	0.38	0.40	0	0.22

Linear

$X^{(n)}$

P = $X^{(n-1)}$

	P_1	P_2	B	R
P_1	0	0.30	0.30	0.40
P_2	0.30	0	0.30	0.40
B	0.30	0.30	0.40	0
R	0.30	0.30	0	0.40

Table 5.3.2: The transition probabilities P [$X^{(n)} = P_i$] for random and linear cases:

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0	0.35	0.18	0.47	0	0.3	0.3	0.4
n = 2	0.4	0.246	0.104	0.25	0.3	0.21	0.21	0.28
n = 3	0.24	0.273	0.14	0.346	0.21	0.237	0.237	0.316
n = 4	0.304	0.267	0.125	0.304	0.237	0.229	0.229	0.305
n = 5	0.279	0.268	0.132	0.322	0.229	0.231	0.231	0.308
n = 6	0.289	0.269	0.129	0.314	0.231	0.23	0.23	0.307
n = 7	0.285	0.268	0.13	0.318	0.23	0.23	0.23	0.307
n = 8	0.286	0.269	0.129	0.316	0.23	0.23	0.23	0.307
n = 9	0.287	0.268	0.129	0.317	0.23	0.23	0.23	0.307
n = 10	0.287	0.268	0.129	0.317	0.23	0.23	0.23	0.307

Scheme III: Let initial probabilities are: Pr_1= 1; Pr_2= 0; Pr_3= 0 and Pr_4= 0

Consider data set of random and linear probabilities matrix are follows:

Random

$X^{(n)}$

P = $X^{(n-1)}$

	P_1	P_2	B	R
P_1	0	0.35	0.18	0.47
P_2	0.40	0	0.18	0.42
B	0.45	0.55	0	0
R	0.38	0.62	0	0

Linear

$X^{(n)}$

P = $X^{(n-1)}$

	P_1	P_2	B	R
P_1	0	0.30	0.30	0.40
P_2	0.30	0	0.30	0.40
B	0.30	0.70	0	0
R	0.30	0.70	0	0

Table 5.3.3: The transition probabilities P [$X^{(n)} = P_i$] for random and linear cases:

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Quantum No.	Random				Linear			
	P_1	P_2	B	R	P_1	P_2	B	R
n = 1	0	0.35	0.18	0.47	0	0.3	0.3	0.4
n = 2	0.4	0.39	0.063	0.147	0.3	0.49	0.09	0.12
n = 3	0.24	0.266	0.142	0.352	0.21	0.237	0.237	0.316
n = 4	0.304	0.38	0.091	0.225	0.237	0.45	0.134	0.179
n = 5	0.278	0.3	0.123	0.302	0.229	0.29	0.206	0.274
n = 6	0.29	0.352	0.104	0.257	0.231	0.405	0.156	0.208
n = 7	0.285	0.318	0.116	0.284	0.231	0.324	0.191	0.254
n = 8	0.287	0.34	0.108	0.268	0.231	0.381	0.167	0.222
n = 9	0.286	0.326	0.113	0.278	0.231	0.342	0.184	0.245
n = 10	0.287	0.334	0.11	0.271	0.231	0.37	0.172	0.229

6. Graphical Analysis

Graphical analysis is performed under above mentioned three schemes in section 4.1, 4.2 and 4.3 with different data sets in section 5.1, 5.2 and 5.3 considering random and linear probability matrix to put various quantum values. So, this analytical discussion on graphs about the variation P $[X^{(n)} = Pi]$ over three data sets are as follows:

6.1 Data set – I:

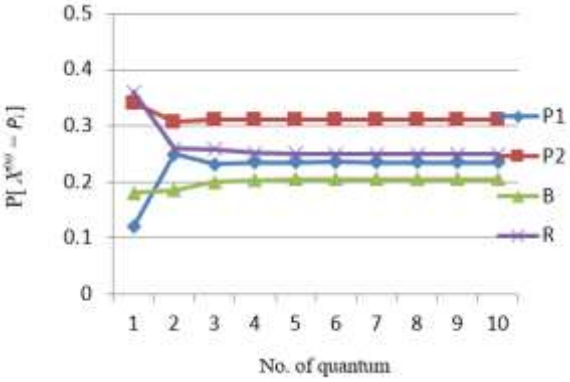


Figure 6.1.1: Scheme – I, random probability.

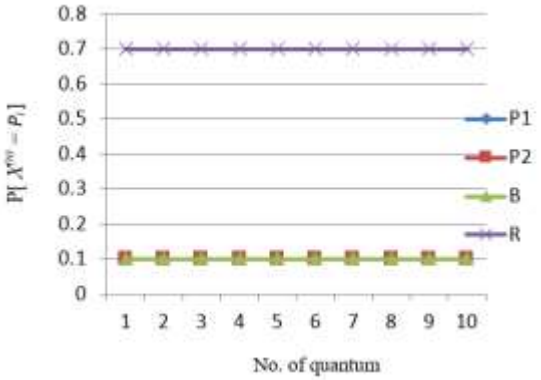
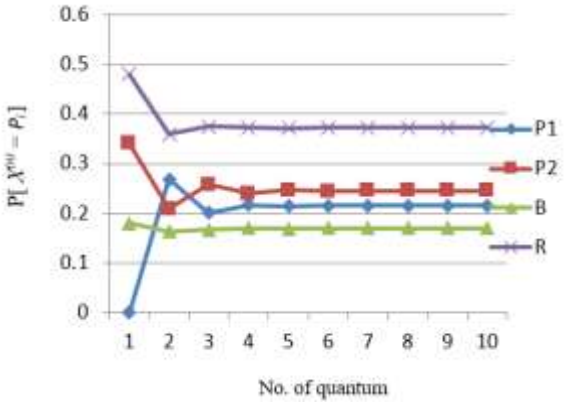
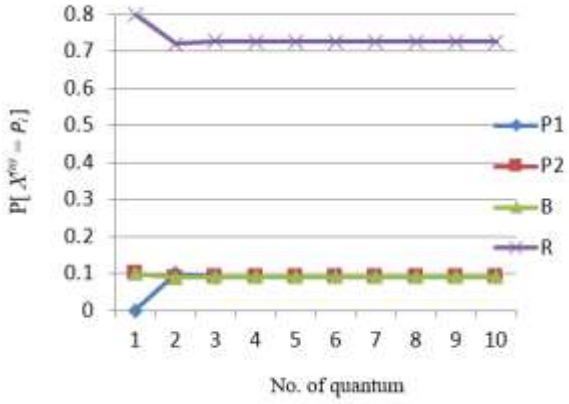


Figure 6.1.4: Scheme – I, linear probability.





Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

Figure 6.1.2: Scheme – II, random probability.

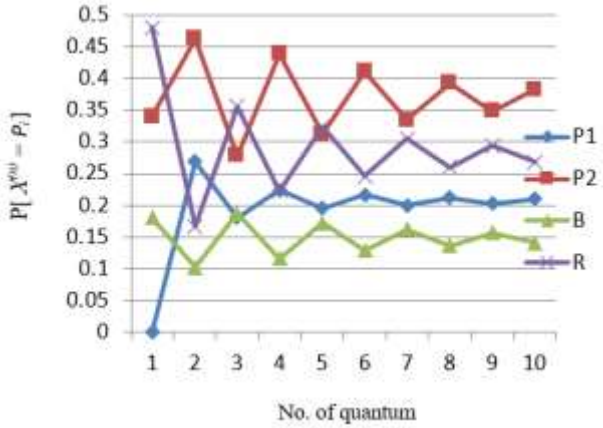


Figure 6.1.3: Scheme – III, random probability.

Figure 6.1.5: Scheme – II, linear probability.

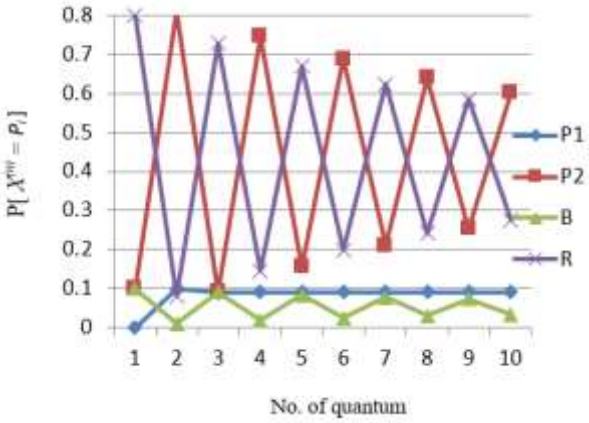


Figure 6.1.6: Scheme – III, linear probability.

Remark: In data set – I, we have observed that the graphical analysis reveals striking similarities across the graphs. Specifically, the dispatcher spends a significantly higher amount of time in the resting state R compared to other transition states. A notable observation in the context of multiprocessor job scheduling is highlighted in figures 6.1.1 and 6.1.3, where the random probability for processor P_2 state is slightly higher than that for the resting state. This suggests an improvement in the dispatcher's performance, indicating a proportional increase in the likelihood of jobs assigned to processor P_2 being executed more frequently than those assigned to processor P_1. This observation underscores the effectiveness of the scheduling strategy in optimizing processor utilization and job execution efficiency.

6.2 Data set – II:

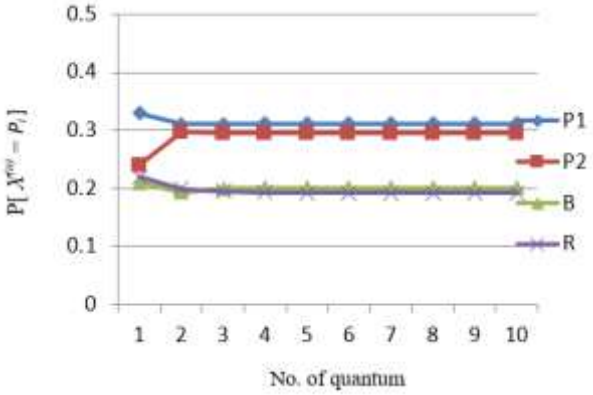


Figure 6.2.1: Scheme – I, random probability.

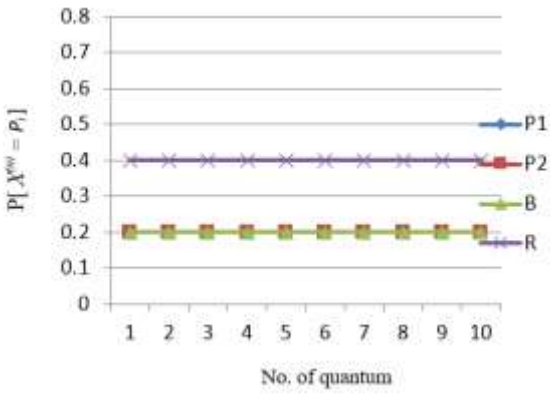


Figure 6.2.4: Scheme – I, linear probability.

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

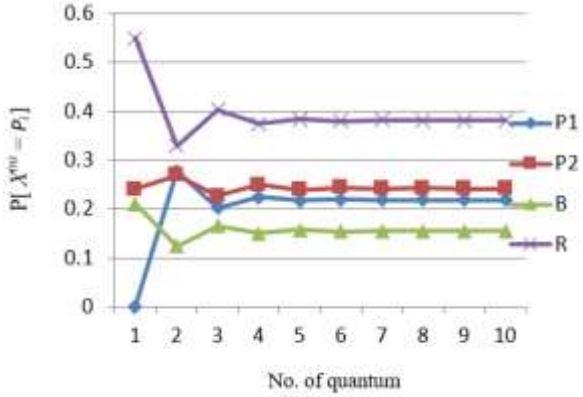


Figure 6.2.2: Scheme – II, random probability.

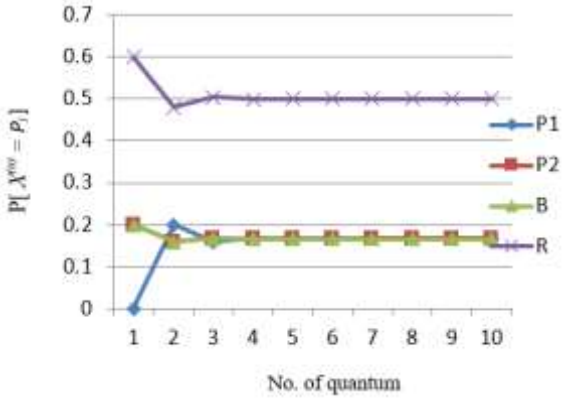


Figure 6.2.5: Scheme – II, linear probability.

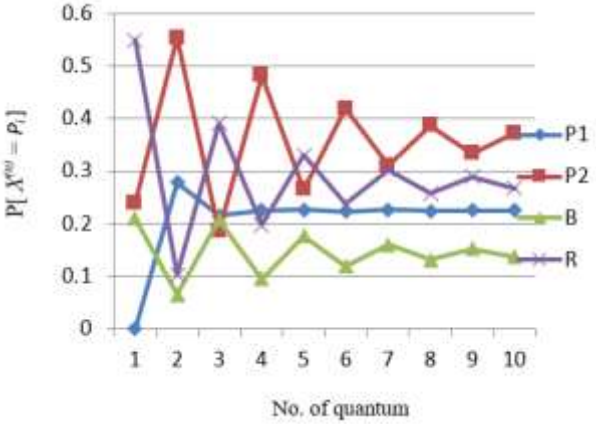


Figure 6.2.3: Scheme – III, random probability.

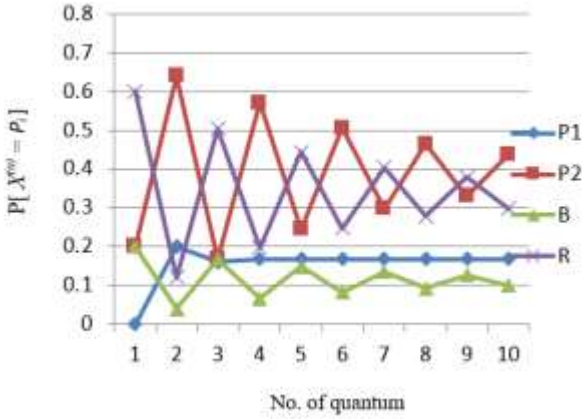


Figure 6.2.6: Scheme – III, linear probability.

Remark: In data set – II, it is evident that the processor states P_1 and P_2 exhibit stable patterns when the number of quantum cycles n >= 5 but up to n = 5 it reflects changing in graphical patterns. An interesting finding is that, across all datasets, the probability of the resting state R remains consistent on average (with exceptions noted in figures 6.2.4 and 6.2.5), while the probability of the busy state B is relatively low. This indicates efficient utilization of processor states and optimal processor utilization overall. The observation suggests that a less restrictive scheduling scheme contributes to enhanced utilization of processor time, leading to improved system performance.

6.3 Data set – III:

Industrial Engineering Journal

ISSN: 0970-2555

Volume : 52, Issue 3, March : 2023

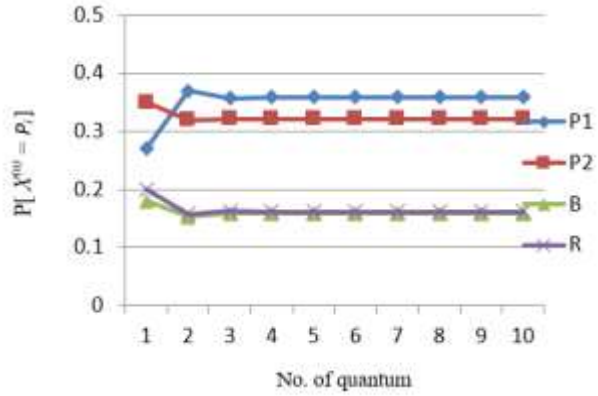


Figure 6.3.1: Scheme – I, random probability.

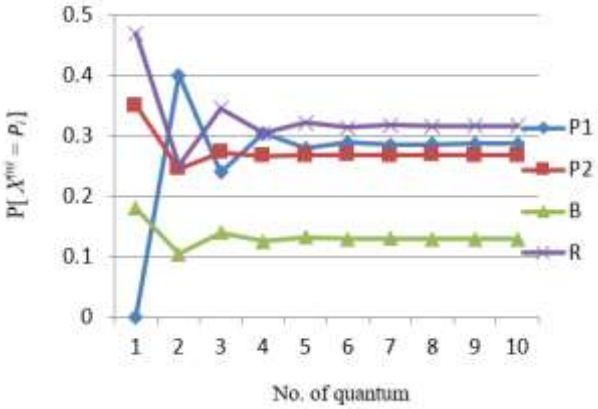


Figure 6.3.2: Scheme – II, random probability.

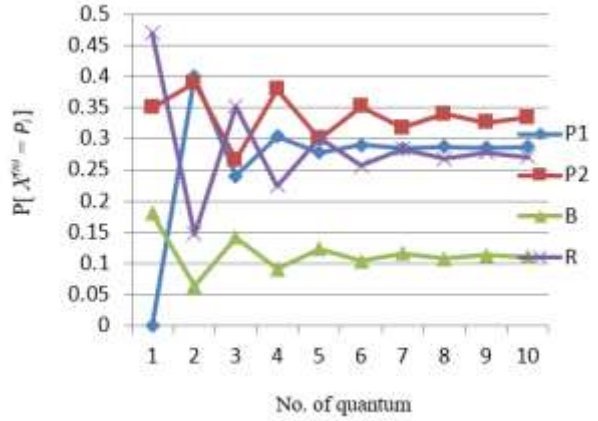


Figure 6.3.3: Scheme – III, random probability.

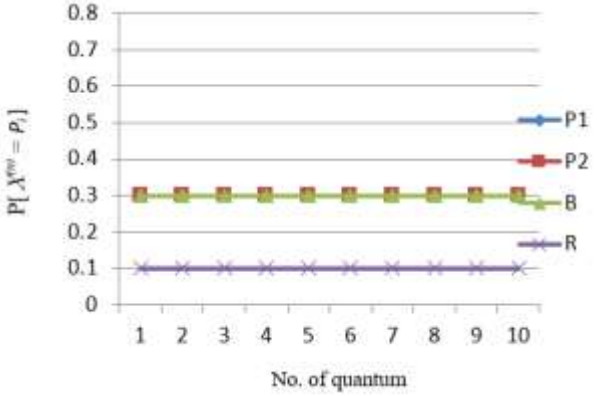


Figure 6.3.4: Scheme – I, linear probability.

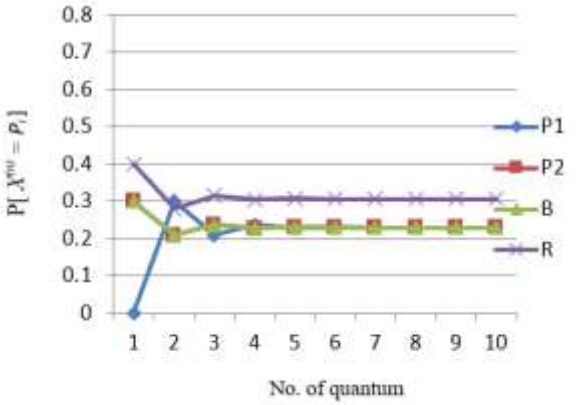


Figure 6.3.5: Scheme – II, linear probability.

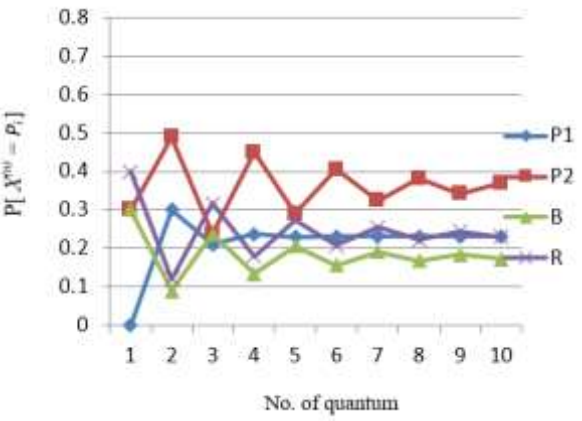


Figure 6.3.6: Scheme – III, linear probability.

Remark: In data set – III, our analysis reveals several key observations. Firstly, the random probability distribution of processor states P_1 and P_2 consistently shows higher values compared to the linear probability states across different quantum numbers. This suggests an improvement in performance efficiency for the Multiprocessor Best Job First (MPBJF) scheduling method used in these datasets.

Conversely, when considering linear probability, the graphs exhibit more variability, indicating less predictability in the dispatcher's performance. This variability translates to a lower likelihood of jobs being executed within processor states. A significant finding from dataset - III is that random probability distributions provide greater opportunities for job processing in a multiprocessor environment compared to linear probability distributions. This observation underscores the effectiveness of random probability in enhancing job execution rates and optimizing system performance under MPBJF scheduling.

7. Conclusion

Efficient and optimal utilization of processors is crucial for any operating system scheduler. Traditional scheduling algorithms have often faced challenges such as inefficiency, high overhead, or compatibility issues with existing scheduler schemes. Therefore, it is imperative for operating systems to adopt CPU scheduling algorithms that are efficient, accurate, and high-performing. This paper presents a comprehensive analysis and comparison of three variants of the Multiprocessor Best Job First (MPBJF) CPU scheduling algorithm using a Markov chain model. We utilize varying probability matrices across multiple datasets, each imposing restrictions on state transition probabilities. MPBJF scheduling seamlessly integrates with existing schedulers by prioritizing jobs on a per-processor basis, offering a practical solution for contemporary operating systems.

We conducted experimental and numerical evaluations to assess the performance of MPBJF scheduling. Through diverse datasets comprising random and linear probability distributions, our experiments demonstrate that MPBJF achieves efficient, accurate, and high-performance results compared to traditional uniprocessing systems. Our formal graphical analyses highlight the stability and effectiveness of MPBJF scheduling schemes, particularly scheme-I and scheme-III when applied to random datasets. These findings underscore the algorithm's potential to enhance system performance and are highly recommended for implementation in practical environments. Moreover, our study suggests that higher transition probabilities contribute to better utilization of processors. Therefore, we recommend that designers of multiprocessor systems consider this principle when developing quantum-based CPU scheduling algorithms. By adopting such strategies, system designers can significantly improve overall system performance and efficiency.

8. Future Scope

Looking forward, the pursuit of efficient and optimal processor utilization remains critical for advancing operating system schedulers. Historical scheduling algorithms have often fallen short due to inefficiencies, high overhead, or incompatibility with existing schemes. Thus, there is an ongoing need for operating systems to continually refine and adopt CPU scheduling algorithms that are not only efficient and accurate but also high-performing. Looking ahead, there are several avenues for future research and development in this domain. One promising direction involves exploring further optimizations in state transition probabilities to enhance MPBJF's performance under diverse operational conditions. Additionally, investigating the integration of machine learning techniques or adaptive algorithms could potentially offer more dynamic and responsive scheduling solutions. Furthermore, our study suggests that higher transition probabilities tend to optimize processor utilization. Therefore, future multiprocessor system designs should prioritize these insights when developing quantum-based

CPU scheduling algorithms. By embracing these advancements, system designers can effectively improve overall system efficiency and performance in increasingly complex computing environments.

9. References

[1] D. Shukla & A. Jain, "Estimation of ready queue processing time under SL-Scheduling scheme in Multiprocessor Environment", IJCSS, Vol. 4, Issue 1, pp. 74-81, 2010.

[2] D. Shukla, A. Jain & K. Verma, "Estimation of Ready Queue Processing Time using Transformed Factor-Type Estimator in Multiprocessor Environment", IJCA, Vol. 79, No. 16, pp. 40-48, 2013.

[3] D. Tam, R. Azimi& M. Stumm, "Thread Clustering: Sharing-Aware Scheduling on SMP-CMP-SMT Multiprocessors", ACM, pp. 47-58, 2007.

[4] G. Levin, S. Funk, C. Sadowski, I. Pye& S. Brandt, "DP-Fair: A Simple Model for Understanding Optimal Multiprocessor Scheduling", 22nd Euromicro Conference on RTS, IEEE pro., pp. 3-13, 2010.

[5] M. Bertogna & M. Cirinei, "Response-Time Analysis for Globally Scheduled Symmetric Multiprocessor Platforms", , pp. , .

[6] R. Sendre, R. Singhai & S. Jain, "Markov Chain Analysis of Improved Round Robin CPU Scheduling Algorithm", IJREAM, Vol. 04, Issue 03, pp. 728-737, 2018.

[7] T. Li, D. Baumberger & S. Hahn, "Efficient and Scalable Multiprocessor Fair Scheduling Using Distributed Weighted Round-Robin", PPoPP'09 ACM, pp. 1-10, 2009.

[8] A. Fedorova, M. Seltzer & M.D. Smith, "Improving Performance Isolation on Chip Multiprocessors via an Operating System Scheduler", 16th International Conference PACT 2007, IEEE Computer Society, pp. 2007.

[9] A. Burns, R.I. Davis, P. Wang & F. Zhang,"Partitioned EDF Scheduling for Multiprocessors using a C=D Scheme", 18th International Conference on RTNS, France, pp. 169-178, 2010.

[10] D. Shukla, A. Jain & A. Chowdhary, "Estimation of ready queue processing time under Usual Lottery Scheduling scheme in Multiprocessor Environment", JACSM, Vol. 11, No. 11, pp. 58-63, 2011.

[11] D. Shukla & A. Jain, "Analysis of ready queue processing time under PPS-LS and SRS-LS scheme in Multiprocessor Environment", GESJ: Computer Science and Telecommunication, Vol. 33, No. 1, pp. 54-61, 2012.

[12] H. Li & S. Baruah, "Global Mixed-Criticality Scheduling on Multiprocessors", 24th Euromicro Conference on RTS, IEEE pro., pp. 166-175, 2012.

[13] M. Cheramy, P.E. Hladik & A.M. Deplanche, "SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithm",5th International Workshop on WATERS, 2014.

[14] R.I. Davis & A. Burns, "A Survey of Hard Real-Time Scheduling for Multiprocessor Systems", ACM Computing Surveys, 2010.

[15] S.R. Vijayalakshmi & G. Padmavathi, "A Performance Study of GA and LHS in Multiprocessor Job Scheduling", IJCSI, Vol. 7, Issue 1, No. 1, pp. 37-42, 2010.

[16] A. Chandra, M. Adler, P. Goyal & P. Shenoy, "Surplus Fair Scheduling: A Proportional-Share CPU Scheduling Algorithm for Symmetric Multiprocessors", Work Supported NSF, CCR, CDA.

[17] G.A. Elliott & J.H. Anderson,"Globally Scheduled Real-Time Multiprocessor Systems with GPUs", Work Supported NSF, ARO, AFOSR and AFRL, pp. 1-54, 2009.

[18] D. Shukla, S. Jain, R. Singhai & R. Agrawal, "Markov Chain Model for the Analysis of Round-Robin Scheduling Scheme", IJANA, Vol. 1, No. 1, pp. 1-7, 2009.

[19] M. Naldi, "Internet Access Traffic Sharing in a Multi-user Environment", Computer Networks, Vol. 38, pp. 809-824, 2002.

[20] D. Shukla & S. Ojha, "Deadlock index analysis of multi-level queue scheduling in operating system using data model approach", GESJ: CST, No. 6(29), pp. 93-110, 2010.

[21] Al-Husainy A.F M, "Best Job First CPU Scheduling Algorithm", ITJ, Vol. 6, No. 2, pp. 288-293, 2007.

[22] J. Medhi, "Stochastic Jobs", Ed. 4, Wiley Limited (Fourth Reprint), New Delhi, 1991(a). -

[23] J. Medhi, "Stochastic Models in Queuing Theory", Academic Press Professional, Inc, San Diego, CA, 1991(b).

[24] R. Sendre & R. Singhai, "Stochastic Process to Analyze Behavior of Improved Round Robin CPU Scheduling Algorithm", IJMIE, Vol. 8, Issue 9, pp. 401-418, 2018.

[25] S. Jain & S. Jain, "Analysis of Multi Level Feedback Queue Scheduling using Markov Chain Model with Data Model Approach", IJANA, Vol. 7, No. 6, pp. 2915-2924, 2016.

[26] A.I. Awad, N.A. EL-Hefnawy & H.M. Abdel_Kader, "Enhanced Particle Swarm Optimization For Task Scheduling In Cloud Computing Environments", Elsevier, ICCMIT-2015, Vol. 65, pp. 920-929, 2015.

[27] R. Shyam & S.K. Nandal, "Improved Mean Round Robin with Shortest Job First Scheduling", IJARCSSE, Vol. 04, Issue 07, pp. 170-179, 2014.

[28] S.Bitam, S. Zeadally & A. Mellouk, "Fog Computing Job Scheduling Optimization Based on Bees Swarm", Taylor & Francis, EIS-2018, Vol. 12, No. 04, pp. 373-397, 2018.

[29] M.B. Gawali, & S.K. Shinde, "Task Scheduling and Resource Allocation in Cloud Computing using a Heuristic Approach", Springer, JCCASA-2018, Vol. 07, No. 04, pp. 02-16, 2018.

[30] I. Attiya, M.A. Elaziz, & S. Xiong, "Job Scheduling in Cloud Computing Using a Modified Harris Hawks Optimization and Simulated Annealing Algorithm", CIN-2020, Vol.2020, pp. 01-17, 2020.

[31] S. Gupta, S. Iyer, G. Agarwal, P. Manoharan, A.D. Algarni, G. Addehim & K. Raahemifar "Efficient Prioritization and Processor Selection Schemes for HEFT Algorithm: A Makespan Optimizer for Task Scheduling in Cloud Environment", Electronics-2022, pp. 01-15, 2022.

[32] S.A. Alsaidy, A.D. Abbood & M.A. Sahid "Heuristic Initialization of PSO Task Scheduling Algorithm in Cloud Computing", Elsevier, Vol. 34, pp. 2370-2382, 2022.

[33] Md.R. Ullah, S. Molla, I.Md. Siddique, A.A. Siddique & Md.M. Abedin "Utilization of Johnson's Algorithm for Enhancing Scheduling Efficiency and Identifying the Best Operation Sequence: An Illustrative Scenario", JRAP, Vol. 08, Issue 03, pp. 11-20, 2023.