We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 3

Heuristics Techniques for Scheduling Problems with Reducing Waiting Time Variance

Satyasundara Mahapatra, Rati Ranjan Dash and Sateesh K. Pradhan

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.69224

Abstract

In real computational world, scheduling is a decision making process. This is nothing but a systematic schedule through which a large numbers of tasks are assigned to the processors. Due to the resource limitation, creation of such schedule is a real challenge. This creates the interest of developing a qualitative scheduler for the processors. These processors are either single or parallel. One of the criteria for improving the efficiency of scheduler is waiting time variance (WTV). Minimizing the WTV of a task is a NP-hard problem. Achieving the quality of service (QoS) in a single or parallel processor by minimizing the WTV is a problem of task scheduling. To enhance the performance of a single or parallel processor, it is required to develop a stable and none overlap scheduler by minimizing WTV. An automated scheduler's performance is always measured by the attributes of QoS. One of the attributes of QoS is 'Timeliness'. First, this chapter presents the importance of heuristics with five heuristic-based solutions. Then applies these heuristics on $1\|WTV$ minimization problem and three heuristics with a unique task distribution mechanism on $Q_m|prec|WTV$ minimization problem. The experimental result shows the performance of heuristic in the form of graph for consonant problems.

Keywords: task scheduling, quality of services, waiting time variance, single processor, uniform parallel processors

1. Introduction

In real world, scheduling is an approach through which a large number of tasks (jobs) are assigned to the resources (processors) that complete the task execution process in time. Due to the limitation of resources, a number of challenging issues are initiated on execution processes. Hence, huge numbers of tasks are waiting in a queue for execution. An efficient and convenient

way of ordering between the tasks and resources is the only solution to resolve these issues. Such ordering is otherwise spelled as scheduling through which the efficiency and accuracy of the task execution process is enhanced. Designing and developing a stable and secured automated scheduler for real world problems is a real challenge for enhancing the quality of services (QoS) of the scheduler. A qualitative automated scheduler's performance is always measured by the attributes of QoS. One of the attributes of QoS is 'Timeliness', which measures the time taken to execute the task and produce an output.

Numerous criteria of timeliness provide good QoS to a task execution process. These criteria are response time, waiting time, turn-around time, elapsed time etc. Delay indicates the extra waiting time taken by the task due to the time consumed by the resources in an execution process. To optimize the scheduling process, new methods with objectives are adapted and integrated as per the requirements and constraints of the issues at hand. In case of discrete alternatives, scheduling is the discipline of decision making. Available resources, imposed constraints, and time required for executions are important factors to form a schedule. These factors are concern for an individual or a group. In real computational world, a series of activities to be outlined serially with the help of these factors is a challenge. This can be described as multiobjective optimization deterministic scheduling problem. The main objectives are to minimize the makespan and not to overlap two or more activities in the same time span with same resources.

Scheduling problems typically involve for search groupings, orderings, or assignments of a discrete set of activities, which satisfy the imposed conditions or constraints. These elements are generally modeled by means of countable discrete structures known as combinatorial structure. These structures are represented through a vector of decision variables which can assume values within a finite or a countable infinite set. Within these settings, a solution for a scheduling problem is a value assignment to the variables that meet specified criteria. Such cases formulate the scheduling problem exploiting the concepts of constraint satisfaction problems or optimization problems.

In Computer Science and Engineering, multiobjective optimization deterministic scheduling problems are belonging to a broad class of combinatorial optimization problems. These combinatorial optimization problems area belongs to NP hard, moreover asymptotically getting an optimal solution in linear time is impossible. In the field of Computer Science and Engineering, mathematical optimizations determine an optimal solution which may be an extremely time consuming procedure due to their computational complexity, whereas heuristic is a technique for finding an approximate solution. In other words, a heuristic is a procedure which produces a quick solution that is good enough for solving the problem at hand. This solution may not be the best of all the actual solutions to this problem, or it may simply approximate the exact solution. But it is still valuable because finding it does not require a prohibitively long time. This is achieved by trading optimality, completeness, accuracy, and precision for speed.

The rest of the section is structured as follows. A brief review of related work of different researchers in scheduling of tasks on single processor and parallel processor with motivation is mentioned in Section 2. In Section 3, the general definition of scheduling problem is briefly discussed. As scheduling is a NP-hard problem, different approaches for solving the scheduling

problem are discussed in Section 4. The classification of deterministic scheduling problem is discussed briefly in Section 5. Different existing heuristic methods are discussed along with pseudo code in Section 6. The single processor scheduling problem with problem formulation and performance analysis of different heuristic methods is discussed in Section 7. In Section 8, the parallel processor scheduling problem with problem formulation and performance analysis of different heuristic methods is discussed. Section 9 contains a brief report on analysis of work leading to conclusion, scope for utilization of this study in different similar areas and suggestions for future research in this field.

2. Review of literature and motivation

In many manufacturing and services industries, scheduling is a decision making process that is used in a day-to-day basis. It deals with the allocation of resources to tasks over a given time period. In computational world, these resources are single processor, multi processors, parallel processors, and dedicated processors. The goal is to optimize one or more objectives such as makespan, mean flow time, mean weighted flow time, mean tardiness, mean earliness, etc. Scheduling problem is a broader class of combinatorial problem, and the purpose is to search a best way to organize task so that it is completed in the shortest possible time as depicted in Refs. [1, 2]. Importance of different types of real world scheduling problems such as single processor scheduling problem, two processor scheduling problems, parallel processor scheduling problems, job shop scheduling problems, flow shop scheduling problems, open shop scheduling problems, etc. are classified and discussed in Ref. [3] and play a significant role in research. The combinatorial problems are belonging to the real world problem. These problems are either problem of minimization or maximization. Such problems consist of a set of instances, candidate solutions for each instance, and a function that assigns to each instance and each candidate solution, a positive rational number called solution value is depicted in Ref. [4]. These problems are distinguished into three subclasses and presented in Ref. [5]. They are named as optimization problem, decision problem, and search problems. An optimization problem is defined as the answer to its instance that specifies a solution for which a value of a certain objective is at its optimum, whereas a decision problem takes only two values, either 'yes' or 'no', as an answer to the instance of the problem. Finally, the search problem simply aims at finding a valid solution, regardless of any quality criterion.

As scheduling is a decision making problem, effective algorithms are developed and designed by the researchers to solve it in due course of time. Such algorithms consist of two parts named as 'head' and 'method'. The head starts with the keyword 'algorithm' followed by a name (i.e., description for the purpose of algorithm), whereas method is used to describe the idea or logic used in the algorithm. The semantic representations are reflected with the help of layout of output, procedure or function name, variable, etc. These algorithms consist of a block of instructions used in a sequential order. Changing the instruction in algorithm changes the behavior of the algorithm is explained in Ref. [2].

Scheduling of task is an integral part of single and parallel computing. Extensive research has been conducted in this area leading to significant theoretical and practical results. New

scheduling algorithms are in demand for addressing concerns originating from the single and parallel processors. How heuristic methodology encourages the researcher to explore and pursue the creative journey through internal discovery in the field of research is presented in Ref. [6].Two heuristic task scheduling methods for single processor, called balanced spiral (BS) and verified spiral (VS), which incorporate certain proven properties of optimal task sequences for minimizing the waiting time variance is proposed in Ref. [7]. The success of stochastic algorithms is often due to their ability to effectively amplify the performance of search heuristics that is focused and discussed in Ref. [8]. A heuristic procedure to minimize the weighted completion time variance in single processor is presented in Ref. [9]. Two heuristic methods named as EC1 an EC2 are developed and proposed in Ref. [10] for solving the problem for a set of large tasks by minimizing waiting time variance in the single machine problem. A novel heuristic method named as RSS is developed and proposed in Ref. [11] for solving the problem for large set of tasks by minimizing waiting time variance in the single machine problem.

Several meta-heuristics have been inspired by nature in due course of time. Two well-known robust metaheuristic methods, including genetic algorithm (GA), simulated annealing (SA), were improved and presented in Ref. [12] to tackle large-scale problems. A MAX-MIN Ant System, which makes use of a separate local search routine, is proposed in Ref. [13] for tackling a typical university course timetabling problem. An ant algorithm based on a multiagent system inspired by the observation of some real ant colony behavior exploiting the stigmergic communication paradigm is discussed in Ref. [14]. An agent-based parallel genetic algorithm for job shop scheduling problem is proposed in Ref. [15]. A genetic algorithm (GA) has been developed in Ref. [16] for minimizing the average residence time to produce a set of batches in function of batch order in a multipurpose-multiproduct batch plant. Multi objective genetic algorithm to find a balance point in respect of a solution of the Pareto front is presented in Ref. [17]. A decomposition heuristics algorithm based on multibottleneck processors for large-scale job shop scheduling problems is proposed in Ref. [18]. A new heuristic based on adaptive memory programming and a simulated annealing algorithm is presented in Ref. [19].

To enhance the property of different heuristic methods for parallel processing in uniform processors, a unique task allocation scheme named as PUM is developed and presented in Ref. [20]. One exact algorithm and one approximation algorithm are proposed in Ref. [21] to minimize the completion time variance. A heuristic algorithm to solve preemptive scheduling problem of dependent tasks on parallel identical processors is proposed in Ref. [22]. A new heuristic algorithm for scheduling metatasks in heterogeneous computing system is presented in Ref. [23]. Heuristic algorithms are proposed to solve a number of independent tasks on multiple number of identical parallel processors problem so as to minimize the waiting time variance [24].

In computing systems, while working with large data files on a Web server, often the response time to a user's request is strongly dependent on the time required to access or retrieve the data files referenced by the user. Especially in online systems, it is often desirable to provide uniform response to user's requests, i.e., minimize the variance of response time by minimizing the variance of access time. The variance of completion time and variance of waiting time performance measures are analyzed [25] for the single processor sequencing problem. These

http://dx.doi.org/10.5772/intechopen.69224

measures are compared and contrasted to the performance measures of mean completion time and mean waiting time. It was shown that the sequence that minimizes the variance of waiting times is antithetical to the sequence that minimizes the variance of flow times, which motivate to take waiting time variance as the performance parameter.

Another motivation is to find out the effectiveness of the methods used for calculation of WTV in parallel processor by efficient task allocation scheme, which will be able to generate a schedule with less time as far as possible.

3. Scheduling problems

The deterministic scheduling problems are part of a much broader class of combinatorial optimization problems. To analyze these problems, the peculiarities of the problem must be studied. The time required for solving those scheduling problems is seriously limited, so that only low-order polynomial time algorithms may be used. Thus, the examination of the complexity of these problems should be the basis for analysis of scheduling problems and algorithms, which is shown in **Figure 1** as a problem solving cycle for deterministic scheduling problem.

The deterministic scheduling problems can be defined as a combination of a set of tasks '$\mathcal{T}$', a set of processors '$\mathcal{P}$', and a set of additional resources '$\mathcal{R}$'. Scheduling means to assign processors from $\mathcal{P}$ and possibly, resources from $\mathcal{R}$ to tasks from $\mathcal{T}$ in order to complete all tasks under the imposed constraints. There are two general constraints arise in classical scheduling theory. They are, each task is to be processed by at most one processor at a time, and each processor is capable of processing at most one task at a time. The processors may be either parallel (i.e., performing the same functions) or dedicated (i.e., specialized for the execution of certain tasks). The parallel processors are distinguished as identical, uniform, and unrelated

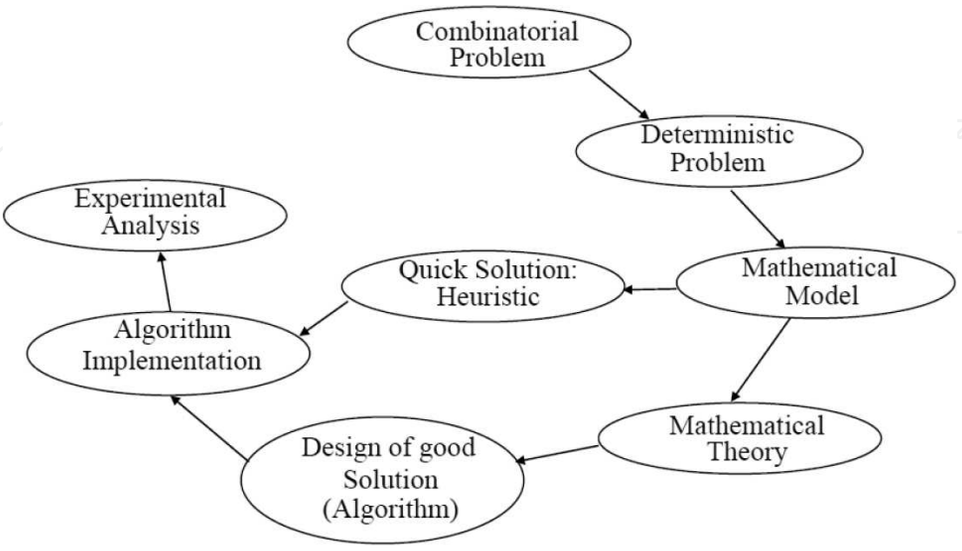

Figure 1. Problem solving cycle for deterministic scheduling problem.

depending on their speeds. In identical, all processors have equal task processing speeds. Similarly, the uniform processors have different speed and unrelated processors depend on the particular task.

The dedicated processors are distinguished as task shop, flow shop, and open shop. In task shop, each task has its own predetermined route to follow with a set of processors. But a distinction is made between task shops in which each task visits each processor at most once and task shops in which a task may visit each processor more than once. On the contrary in flow shop, a set of processor are placed in series. Each task has to be processed on every processor exactly once. All tasks have to follow the same route, i.e., they have to be processed first on processor 1, then on processor 2, and so on. In case of open shop, a set of tasks must be processed for given amounts of time at each of a given set of processors, in an arbitrary order. The idea is to determine the time at which each task is to be processed at each processor. In such systems, it is assumed that the buffers between processors have unlimited capacity and a task after completion on one processor may wait before its processing starts on the next one. However, buffers of zero capacity tasks cannot wait between two consecutive processors are termed as no-wait property.

The classical deterministic scheduling problem can be stated as follows. There are a set of n tasks simultaneously available for being processed on a set of m processors. Let all tasks available for processing at time zero. Each task j, $j \in \mathcal{T} = \{1, 2, \ldots, n\}$, passes through the processors 1, 2, …, m in that order and requires an uninterrupted processing time pt_{ij} on processor i, $i \in \mathcal{P} = \{1, 2, \ldots, m\}$. The scheduling objective is to minimize makespan. Makespan or maximum completion time is the time interval between starting the first task on a processor and the completion of the last processor and denoted by C_{max}. Let $\mathcal{T}_j$ be the set of subtasks scheduled on processor i. Then, the completion time on processor i can be computed as $C_i = \Sigma_{j \in \mathcal{P}_i} pt_{ij}$. Hence, maximum completion time, i.e., makespan can be calculated as $C_{max} = max_{i \in \mathcal{P}} C_i$. To minimize the makespan of a deterministic scheduling problem, apriori knowledge on different procedure of scheduling schemes is required and discussed in the next section.

4. Approaches to scheduling problems

From the literature review, it was observed that there exists a large class of combinatorial optimization problems for which most probably no polynomial optimization algorithms are available. These are the problems whose decision counterparts are NP complete. Hence, in such cases, the optimization problems are NP hard. A comprehensive study on NP completeness, NP hardness, polynomial time transformation, etc. helps the researchers in analyzing the multiobjective scheduling problem. It also helps the researchers to solve those problems by using polynomial time algorithm. The usefulness of the algorithm depends on the order of its worst-case complexity function and on the particular application. It was found that sometimes, the worst-case complexity function is not low enough, although still polynomial, a mean complexity function of the algorithm may be sufficient. On the other hand, if the decision

http://dx.doi.org/10.5772/intechopen.69224

version of the analyzed problem is NP complete, then there are several approaches taken into consideration to make the problem NP hard. These approaches are discussed below.

First, constraints like allowing preemptions, assuming unit-length tasks, and assuming certain types of precedence graphs are relaxed by imposing on the original problem and then solving the relaxed problem. The solution of the latter may be a good approximation to the solution of the original problem. In case of computer application, the relaxation method is justified when parallel processors share a common primary memory. Moreover, such a relaxation is also advantageous from the viewpoint of certain optimality criteria.

Second, in the process of solving NP hard scheduling problems, the use of approximation algorithms tends to find an optimal schedule but does not always succeed. It is a useful heuristic for finding near optimal solutions, when the optimal solution is not required [5]. The necessary condition for these algorithms to be applicable in practice is that their worst-case complexity function is bounded from above by a low-order polynomial in the input length. So that approximation algorithm often give raise to heuristic that return solution much closer to optimal than indicated by their performance guarantee and bring the researchers to study of heuristics and allowed to prove how well the heuristic performs on all instances [5]. Their sufficiency follows from an evaluation of the difference between the value of a solution they produce and the value of an optimal solution. This evaluation may concern the worst case or a mean behavior. However, for some combinatorial problems, it can be proved that there is no hope of finding an approximation algorithm of certain accuracy.

Analysis of the worst-case behavior of an approximation algorithm may be complimented by an analysis of its mean behavior. This can be done in two ways. The first consists in assuming that the parameters of instances of the considered problem are drawn from a certain distribution, and then the mean performance of algorithm is analyzed. Distinguish between the absolute error and the relative error asymptotic optimality results in the stronger (absolute) sense are quite rare. On the other hand, asymptotic optimality in the relative sense is often easier to establish. It is rather obvious that the mean performance can be much better than the worst-case behavior, thus justifying the use of a given approximation algorithm. A main obstacle is the difficulty of proofs of the mean performance for realistic distribution functions. Thus, the second way of evaluating the mean behavior of approximation algorithms, consisting of experimental studies, is still used very often in real world problems.

The third and the last way of dealing with hard scheduling problems is to use exact enumerative algorithms whose worst-case complexity function is exponential in the input length. Such problems are not NP hard in strong sense. These problems are possible to solve by pseudo-polynomial optimization algorithm whose worst-case complexity function is bounded from above by a polynomial in the input length and in the maximum number appearing in the instance of the problem. For reasonably small numbers, such an algorithm may behave quite well in practice, and it can be used even in computer applications.

The above discussion is summarized in a schematic way in **Figure 2**. It is observed that finding an exact algorithm for a large-scale task scheduling problem is not easy. Hence, local optimum

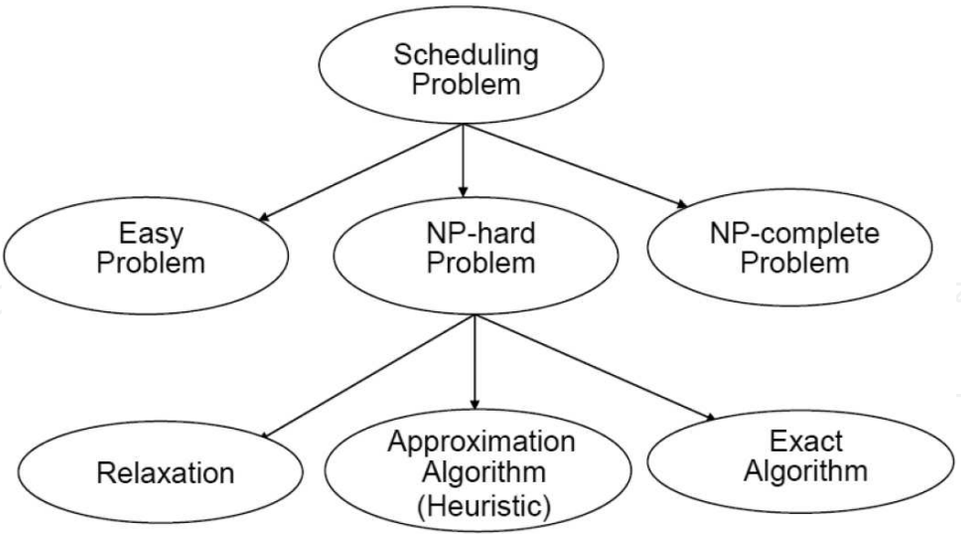

Figure 2. Approaches to scheduling problem.

algorithm as heuristic is always better to develop and to be used. Knowledge of classification for these scheduling problems serves as a basis for developing heuristic algorithms, which is discussed in next session.

5. Classification of deterministic scheduling problems

A scheduling problem is described by a triplet $(\alpha|\beta|\gamma)$ and shows the possible classification under the each parameter of the triplet [26]. A detailed nature of triplet is explained in appendix A. The symbol α is represented for processor environment and contains only one entry that is classified into two types, named as single processor and multiple processors. Single processor again is classified into three categories. They are named as single processor, parallel processor, and dedicated processor.

Parallel processors are classified as per their behavior of the parallelisms into three types. They are named as identical parallel processors, uniform parallel processors, and unrelated parallel processors denoted by the symbol P, Q, and R, respectively. Similarly dedicated processors are classified into three categories named as flow shop processors, task shop processors, and open shop processors denoted by the symbol F, J, and O, respectively.

The symbol β is represented for different types of tasks and resource constraints. It may contain no entry at all, a single entry, or multiple entries. The possible entries in the β field are preemption (*pmtn*) used for the interruption of task and re-start in latter, resource (*res*) used for identification of particular type of resource, precedence (*prec*) required for the completion of one or more tasks before another task is allowed to start its processing, ready time (r_j) represents the task j starting time for processing, delivery time (q_j) represents the time spent for delivery the task j after its processing, processing time (pt_j) represents the processing time of task j on a processor, deadline $(\tilde{d})$ are imposed on the performance of a task set, maximal number of tasks ($n_j \leq k$) describes the maximal number of sub-tasks (n_j) constituting a task (k) in

Completion time	C_j
Flow time	$F_j = C_j - r_j$
Lateness	$L_j = C_j - d_j$
Tardiness	$D_j = max\{C_j - d_j, 0\}$
Earliness	$E_j = max\{d_j - C_j, 0\}$
Tardy task unit	$U_j = \begin{cases} 1 & if\ C_j > d_j \\ 0 & Othewise \end{cases}$

Table 1. Objective functions.

case of task shop systems, and no-wait (*no – wait*) describes a no-wait property in the case of scheduling on dedicated processors.

The symbol γ is represented as objective function for minimizing the different performance measure (i.e., optimality criteria) of scheduling and contains single entry only. These measures are depicted in Ref. [26], and the parameter required for computing these objective function of a task j is calculated and given in **Table 1**.

As it has been observed from different research articles that a good number of objectives are available for minimizing the different performance measure of scheduling, the ultimate objective is minimizing the makespan. To fulfill the aforementioned objective under different constraints, several methods have been developed which therefore gives raise to various classes of schedules.

6. Heuristic methods for scheduling problems

From the literatures, it is observed that a number of task ordering methods are developed and improved in due course of time. These methods either belong to exact or heuristic or meta-heuristic methods. In the process of searching a best or improved method with desired objective, all possible solutions are tested one by one. This process is viable only for small size of problems but very challenging, complicated, and time consuming as the size of problem increases. Therefore, to reorder the tasks of large problems, heuristic methods are developed for obtaining optimum solution. The solutions obtained by the heuristic methods are optimum or near optimum in nature by using less number of computer resources and computational time. Calculation of CTV and WTV are the two objectives for these types of heuristics. For minimizing the WTV, Elion & Chowdhary, verified spiral (VS), balanced spiral (BS), and Rati-Satya-Sateesh (RSS) heuristic methods are discussed below.

6.1. Eilon and Chowdhary (EC1 & EC2)

EC1 and EC2 are two types of heuristics, designed and presented in Ref. [10] for an n-task WTV problem. Here, 'n' numbers of tasks are scheduled on the basis of V shape property of optimal sequence. In case of EC1, the largest processing time task is removed from the job queue and placed at last position of the schedule. The second largest processing time task is

removed from the job queue and placed at the first position of the schedule. Similarly, the third largest processing time task is removed from the job queue and placed at the last but one position of the schedule. The fourth largest processing time task is removed from the job queue and placed in second position of the schedule. This process continues until the job queue is empty. This method places the jobs in a spiral front and rear manner. EC2 heuristic, the modified version of EC1, produces the task schedule by incorporating the Schrage's conjecture with EC1. The Schrage's conjecture states that there exists an optimal sequence in which the largest job is scheduled at last position, the second longest is at first position, the third and fourth longest are last-but-one position and last-but-two position, respectively, in the sequence [27].

6.2. Verified spiral (VS)

Verified spiral (VS) presented in Ref. [7] is an improved version of EC1. This method incorporates Schrage's conjecture and Hall & Kubiak's proof [28] about the placement of first three largest processing time tasks. For the remaining task on the task queue, a modified spiral placement method is implemented. This method removes the next task from the task queue and place either after the front task or before the rear task of the sequence on the basis of which position produces a small WTV with the existing tasks.

6.3. Balanced spiral (BS)

The balanced spiral (BS) method discussed in Ref. [7] is developed to reduce the computational cost of VS method. This method is otherwise known as observation method, as it balance the left (L) and right (R) optimal sequence to get optimum or near optimum sequence after placing the processing time of each tasks in sequence one by one until the task queue is empty.

6.4. Rati-Satya-Sateesh (RSS)

In our locality, the fishmongers are those who sell a whole unit of fish. Sometimes a large fish has to be distributed equally to two or more customers. These fishmongers are so skilled that they can equally distribute the cut pieces of the same fish among the customers during the time of cutting. It reduces the post measurement for equality, which generally found almost equal. This distribution mechanism to serve the customers used in this method is named as RSS, presented in Ref. [20]. This method allocates the tasks in the sequence with minimum computational cost and time.

The effectiveness of the above discussed methods is presented in the next two sections by using single processor and parallel processors with an objective WTV.

7. Single processor scheduling

In the task scheduling problem, '*n*' number of tasks has to be processed by a single processor with some processing objectives, order, and constraints. Discovering an optimized schedule, which minimizes the WTV of the tasks, is the aspiration of the problem. Due to nonavailability

of the processor in real time, a task has to wait for processing, as the processor is processing another task and may also due to the precedence process constraint.

In the process of searching, an effective and optimized sequence of tasks, it needs to calculate all possible combination of tasks (factorial n). It consumes much time and resources to give an optimum sequence. Different heuristics and meta-heuristics methods are required to develop by reducing the number of calculations for handling many concurrent tasks in computer and in network systems. To achieve this service stability on an individual recourse, it is required to minimize the WTV, which is the objective of the task scheduling problem on single processor.

7.1. Problem formulation

The above mentioned problem can typically describe as an allocation of tasks to a processor by considering the concept that once a task get into the processor for processing, it did not leave from the processor until the processing time of that task was over. The decision whether the task *"j"* (i.e., the task number) is scheduled to the processor successfully, then *"k"* the allocation variable is 1 (one) or 0 (zero) otherwise, which can be represented by an integer. These decision variable depends upon the position of task in the task sequence, which is represented by s_{kj} for $k \in \mathcal{L} = \{1, 2, \ldots, n\}$ and $j \in \mathcal{T} = \{1, 2, \ldots, n\}$. The task to be scheduled first is placed at first position, thus processed first; the task to be scheduled second is placed at second position, thus processed second, and so on. Then, the waiting time for task j at position k is represented as wt_{kj} and the processing time of task j is represented by pt_j. The WTV of tasks in a complete sequence is obtained as follows in Eq. (S.1).

$$WTV = \frac{1}{n-1}\sum\nolimits_{j \in \mathcal{L}_k}\left(wt_{kj} - \frac{1}{n}\sum\nolimits_{j \in \mathcal{L}_k} wt_{kj}\right)^2 \tag{S.1}$$

The objective is to minimize the variance of waiting time of n number of tasks can be found by Eq. (S.2).

$$Minimize\ 1||WTV \tag{S.2}$$

subject to:

$$\sum\nolimits_{k \in \mathcal{T}_j} s_{kj} = 1 \tag{S.3}$$

$$\sum\nolimits_{j \in \mathcal{L}_k} s_{kj} = 1 \tag{S.4}$$

$$s_{kj} = 0\ or\ 1 \quad \forall k \in \mathcal{L},\ j \in \mathcal{T} \tag{S.5}$$

$$wt_{kj} = 0 \quad \forall k = 1\ ,\ j \in \mathcal{T} \tag{S.6}$$

$$wt_{kj} \geq wt_{k-1\,j} + \sum\nolimits_{j \in \mathcal{L}_k} s_{k-1\,j} * pt_j \quad \forall k \in \mathcal{L}\ ,\ j \in \mathcal{T} \tag{S.7}$$

The constraint that each position of the sequence is used exactly once by a task is described in Eq. (S.3). Each task is assigned to a position in the sequence is exactly described once in

Eq. (S.4). The integer constraint for decision variable is described in Eq. (S.5). The waiting time for the first task is described in Eq. (S.6), and the waiting time wt_{kj} of the task at position k ($k >= 2$) is described in recursive Eq. (S.7).

7.2. Problems for testing and performance analysis

This section presents the effectiveness of five heuristic algorithms discussed in section 6 by generating the test cases with the help of three probability distributions namely normal distribution, Poisson distribution, and exponential distribution. At first, a small set of test cases have been selected which are same as used in Refs. [7, 10] to find the effectiveness of the algorithms. To increase the number of testing cases, another three large sets of data are also generated randomly of 5 through 500 numbers of tasks. These large data sets are generated with the help of normal, Poisson, and exponential distribution, respectively.

To measure the performance of the heuristics presented in Section 6, at first for optimality, all possible sequences are generated by placing the tasks randomly for each problem of small data set. Each generated possible sequence is considered as one sub example of all possible optimal sequences. For example, there are 120 numbers of task sequences (e.g., 5!) are generated for 5 numbers of tasks. Similarly, there are 720 numbers of task sequences (e.g., 6!) are generated as there are six tasks so on. But the above discussed five heuristic methods generate only one task sequence for each test case of small data set. The basic aim is to calculate WTV for the test cases, which satisfy the V-shaped optimal property.

Figure 3 shows the WTV performance of five heuristic methods is as good as the performance of optimal methods for small size test cases. It was also observed that the RSS method gives optimum or near optimum WTV results as compared with optimum generated WTV value.

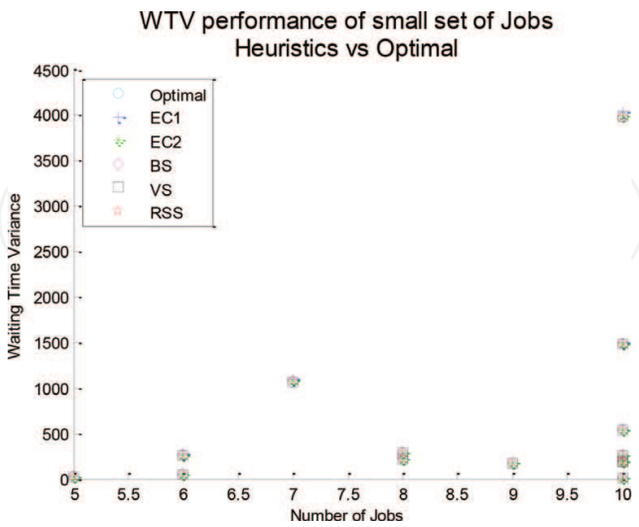

Figure 3. WTV performance of between heuristics vs optimal for small set of jobs.

http://dx.doi.org/10.5772/intechopen.69224

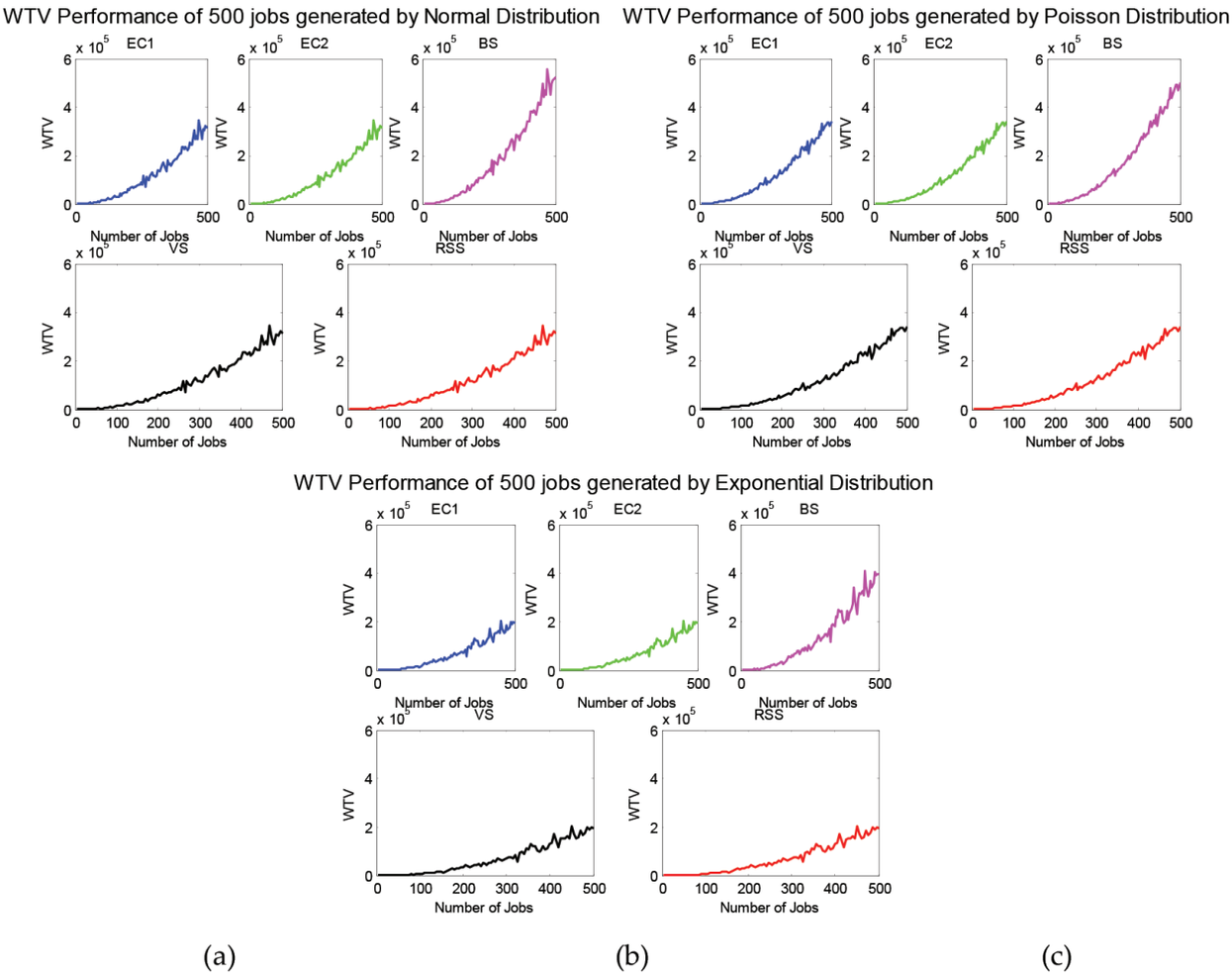

Figure 4. Performance of WTV with respect to heuristics methods for large set of data (i.e., processing time) generated by normal, Poisson, and exponential distribution.

The WTV performance of EC1, EC2, VS, BS, and RSS heuristic methods for all the test cases of large data set is shown in **Figure 4**. The computational result depicted that the WTV obtained by RSS method seems to be near optimum in comparison with other four methods for different numbers of tasks generated by three distribution methods discussed above.

For single processor scheduling problem, the computational cost is treated as computational average time. It is observed that all heuristic methods used sorting mechanism before the generation of tasks sequence except optimal method. Quick sort is an efficient sorting mechanism that takes *O (n log n)* computational cost. It is also observed that the sequence generated by VS method takes much larger computational cost than BS and RSS method as the calculation of WTV is made multiple times. The sequence generated by BS method also takes larger computational cost than RSS method as the calculation of total processing time is made multiple times. Hence, by applying the concept of cutting a large fish into small pieces and distributed among the customers uniformly by a fishmonger generate an optimum or near optimum sequence by minimizing WTV in very less computational cost is a major achievement.

8. Parallel processor scheduling

Parallel processing is one of the arising concepts that used to schedule a batch of *'n'* numbers of tasks to be processed by *'m'* numbers of parallel processors [24]. This section presents a parallel scheduling algorithm as a solution to the problem $Q_m|prec|WTV$ with an effect of minimization of mean WVT. This approach is a heuristic based and the tasks are allocated dynamically in the task sequence by keeping variance as a controlling parameter. The tasks are placed in the individual sequence with the help of heuristic algorithms, so that the dynamic heuristic methods take extremely less computational cost. These algorithms are tested on randomly generated problems of varying numbers of tasks and processors as parameters. The effectiveness with respect to mean WTV is done by comparing the result among the discussed heuristic methods. The findings are shown in graphic form for corresponding problems.

8.1. Problem formulation for task scheduling problem

The uniform parallel processors $i \in \mathcal{P} = \{1, 2, \ldots, m\}$ are having different speeds $s \in \mathcal{S} = \{1, 2, \ldots, m\}$ with the relation $s_1 < s_2 < s_3 < \ldots < s_m$. This means that the first processor is the slowest processor with low processing cost and the last processor is the fastest processor with high processing cost. For a given task, the processing times on the uniform parallel processor is in the ratios listed as $1/s_1 : 1/s_2 : 1/s_3 : \ldots : 1/s_m$. The processors are continuously available, and they are never kept idle while work is waiting. The processors are assigned by the maximum processing time capacity of a task, so that allotments of tasks are assigned on the basis of the processing time. Thus, low processing time tasks are assigned to slowest processors and highest processing time tasks are assigned to fastest processors. The designed uniform scheduling problem is based on the allocation of n, numbers of independent tasks $j \in \mathcal{T} = \{1, 2, \ldots, n\}$ as per the processing time at location $k \in \mathcal{L} = \{1, 2, \ldots, n\}$ on a set of m numbers of uniform parallel processors $i \in \mathcal{P} = \{1, 2, \ldots, m\}$.

The problem is formulated under five numbers of assumptions. At first, the starting time of individual processors are assumed to initialize at time 0 (zero). In other words, all the tasks for each processor are ready to begin for processing at the same time, i.e., 0 (zero). Second, each processor is available deliberately prior to a condition that once the processor given a task to process, it cannot be preempted until the task's processing time is completed on that processor. Third, once a task is allocated to any one processor, it cannot be laid away to other processor under any circumstance. Fourth, the number of tasks must be greater than the number of processors, i.e., $n > m$, as the problem with n ≤ m is irrelevant. Fifth, all the allocated processors will be waiting according to the order of allocation, i.e., after the previously allocated task has been finished the present task can be started.

From the literature, it was observed that number dominant properties on WTV problem has been discovered and depicted by the researchers. To start, first for any scheduling sequence *R*, *CTV* of *R* is equal to *WTV* of *R*′, where *R*′ is the antithetical schedule of *R* [25]. Second, the scheduling sequence that minimizes *WTV* is antithetical to the scheduling sequence that minimizes *CTV* [25]. Third, *CTV* remains unchanged when reversing the order of the last $n1$ tasks [25]. Fourth, for *CTV* minimization problems, an optimal scheduling sequence is of the

form of (n, $n-2$, …, $n-1$), i.e., the largest task is arranged at the first position, the second longest task is arranged at the last position, and the third longest task is arranged at the second position [28]. Fifth, the optimal sequence for a WTV minimization problem is V shaped [10]. Sixth, $P_m \parallel CTV$ problem is NP complete in the strong sense when 'm' is arbitrary [24]. Seventh, $P_m \parallel CTV$ Problem is NP complete in the ordinary sense when 'm' is fixed [24].

Minimization of WTV as a performance measure for task scheduling problem has been discussed in Section 7 for achieving the service stability between the tasks in single processor. The parallel processor is nothing but multiple numbers of single processors with same speed or multiple numbers of single processors with different speed are working simultaneously for achieving the concurrency. Hence, to come up with an optimized schedule, which minimize the WTV is the aspiration of the task scheduling problem in parallel environment. The WTV developed (S.1) in Section 7 will be utilized for the development of the WTV on parallel processors. The WTV of tasks in a complete sequence for the parallel processor is obtained as follows in Eq. (P.1).

$$WTV = \frac{1}{m}\sum_{i\in\mathcal{P}}\left(\frac{1}{n_i-1}\sum_{j\in\mathcal{L}_{ki}}\left(wt_{kij}-\frac{1}{n_i}\sum_{j\in\mathcal{L}_k}wt_{kij}\right)^2\right) \tag{P.1}$$

The objective is to find an optimum or near optimum schedule with pseudo-polynomial time of $Q_m|prec|WTV$ problem by minimizing the variance of waiting time for n number of tasks on m number of uniform parallel processors by Eq. (P.2).

$$Minimize\ (Q_m|prec|WTV) \tag{P.2}$$

subject to:

$$\Sigma_{j\in\mathcal{P}i}\, s_{kij} = 1 \qquad \forall\, k\in\mathcal{L} \tag{P.3}$$

$$\Sigma_{i\in\mathcal{T}_j} s_{kij} = 1 \qquad \forall\, k\in\mathcal{L} \tag{P.4}$$

$$\Sigma_{j\in\mathcal{P}_i} wt_{kij} = 0 \qquad \forall\, k=1 \tag{P.5}$$

$$wt_{kij} = wt_{k-1ij} + \Sigma_{j\in\mathcal{P}_i} s_{kij} * pt_j \qquad \forall\, k\in\mathcal{L},\ j\in\mathcal{T},\ i\in\mathcal{P} \tag{P.6}$$

$$\Sigma_{j\in\mathcal{P}_i} q_{kij} = 1\ \forall\, k\in\mathcal{L} \tag{P.7}$$

$$q_{kij} * N + wt_{k+1ij} \geq wt_{kij} + \Sigma_{j\in\mathcal{P}_i} s_{kij} * pt_j \qquad \forall\, k\in\mathcal{L},\ j\in\mathcal{T},\ i\in\mathcal{P} \tag{P.8}$$

$$s_{kij}\in\{0,1\} \qquad \forall\, k\in\mathcal{L},\ j\in\mathcal{T},\ i\in\mathcal{P} \tag{P.9}$$

$$q_{kij}\in\{0,1\} \qquad \forall\, k\in\mathcal{L},\ j\in\mathcal{T},\ i\in\mathcal{P} \tag{P.10}$$

$$C_{kij}\geq 0 \qquad \forall\, k\in\mathcal{L},\ j\in\mathcal{T},\ i\in\mathcal{P} \tag{P.11}$$

$$wt_{kij}\geq 0 \qquad \forall\, k\in\mathcal{L},\ j\in\mathcal{T},\ i\in\mathcal{P} \tag{P.12}$$

where N is large number.

Each task is assigned to a position is exactly once in any one of the processor sequence is described in Eq. (P.3). Each position of any one process or sequence is used exactly once by a task is described in Eq. (P.4). The waiting time for first task of the individual processor sequence is described in Eq. (P.5). The waiting time of all other allotted tasks for the individual processor except first one is described in Eq. (P.6). Eqs. (P.7) and (P.8) state that if two tasks are on the same processor, then one must be scheduled after the other; otherwise, the values of wt_{kij} and wt_{k+1ij} will not be related. Eqs. (P.9) and (P.10) indicate that the introduced decision variables are binary in nature. Eqs. (P.11) and (P.12) represent that the value of completion time and waiting time must be greater than zero.

8.2. Task allocation methods for uniform parallel processors

The uniform parallel processors are identified by their different speeds. The processors are arranged in chronological order, such that the first processor is the slowest processor with low processing cost and the last processor is the fastest processor with high processing cost. The scheduling problem ($Q_m|prec|WTV$) discussed above is a combinatorial problem. Therefore, usage of a heuristic is inevitable to obtain solution in polynomial time. The challenge is to distribute the tasks in an efficient manner among the processors. A unique task allocation method named as PUM is presented in Ref. [20] for the allocation of tasks among the processors.

Uniform parallel processors consist of a bank of single processors with different speed, and the computational cost is depending on the speed of the processors. It is most important to allocate the task in such a way that the computational cost must be maintained. Hence, the unique task allocation scheme named as PUM is combined with the heuristic methods namely VS, BS, and RSS is also discussed in Ref. [20]. The efficiency of the three heuristic methods with the unique task allocation scheme for uniform parallel processors is tested with a large number of test cases discussed in the next section.

8.3. Problems for testing and performance analysis

To find the effectiveness of these heuristic methods, test cases are randomly generated with the help of four probability distributions. At first with the help of normal distribution, 901 numbers of test cases are generated randomly in combination of 5 and 6 numbers of uniform parallel processors for each case of 100 through 1000 numbers of tasks. The test cases are followed by the same number of tasks and processors with the help of Poisson distribution, exponential distribution, and uniform distribution. The performance analysis of the heuristic methods with unique task allocation scheme is discussed below.

For analysis, mean WTV is taken as the measure of performance. Performance of measure of three heuristic methods named as VS, BS, and RSS is analyzed by using a unique task allocation scheme named as PUM. This enhances the performance of heuristic methods for parallel processing in uniform processors. The allocation scheme in combination with heuristic algorithms is tested with a large number of test cases starting from 100 to 1000 tasks separately. The results analysis for normal distribution on uniform parallel processor is presented in **Figure 5**,

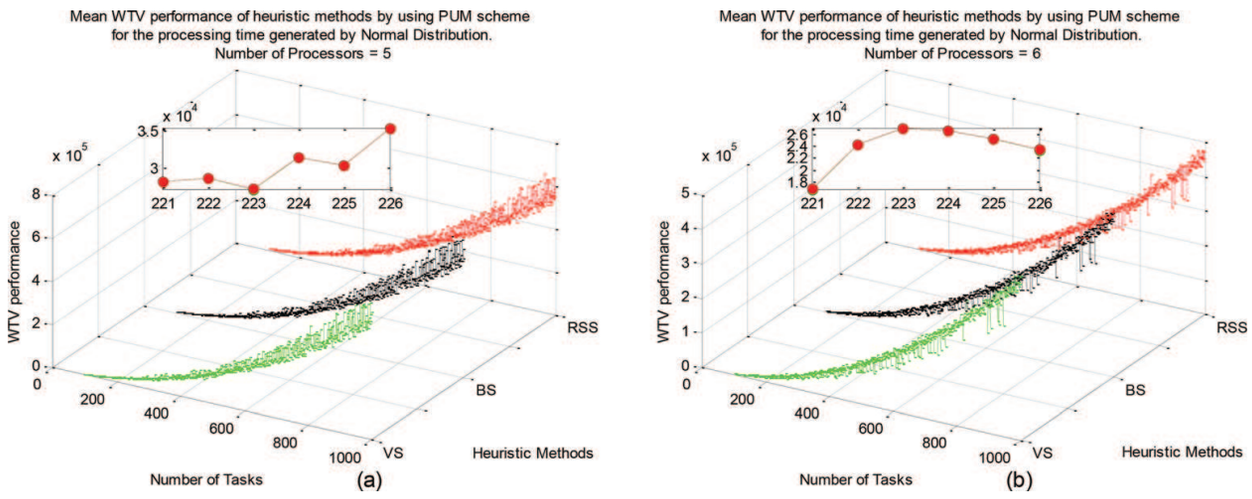

Figure 5. Comparison of mean WTV with respect to heuristics methods by using PUM allocation scheme for the processing time generated by normal distribution.

which consists of two subfigures (a) and (b). The mean WTV obtained by the three heuristic algorithms with the help of unique task allocation scheme is shown in each subfigure. The task allocation schemes are implemented on each test case generated by normal distributions. The subfigures (a) and (b) represent mean WTV performance for 5 and 6 numbers of uniform parallel processors, respectively. The three heuristic methods are represented in each subfigure (a) and (b) by three distinct colors. Green color represents VS method, black color represents BS method, and red color represents RSS method. An enlarged view of mean WTV performance of heuristic methods from total task numbers 221 to 226 is presented in each subfigure. The computational result shows that the mean WTV obtained by RSS methods in combination of PUM is apparently same in comparison with other two heuristic methods.

Similarly, the processing time for all the test cases is generated with the help of Poisson, exponential, and uniform distribution, respectively. It is also observed that mean WTV obtained by RSS methods in combination of PUM are apparently same in comparison with other two heuristic methods as presented in Ref. [20].

Developing an efficient task allocation scheme and execute it with the heuristic methods for uniform parallel processors is NP hard. To overcome it in uniform parallel processor, an efficient task allocation scheme is required along with the heuristic methods. The average time required for finding sequence by computing the heuristics in uniform parallel processor is represented as computational cost. From the above discussed heuristic methods with PUM allocation scheme, it is found that the VS method requires at least four tasks to commence, and all the heuristic methods discussed in Section 6 need a sorting procedure after the PUM allocation process is over and before the starting of heuristic process. Quick sort is an efficient sorting mechanism that takes $O\ (n \log n)$ computational cost. Hence, it is used to sort the tasks before implementation of heuristics. From the performance analysis, it is observed that the computational cost of VS method is much larger then BS and RSS method, as the calculation of WTV is made multiple times, and the computational cost of BS method is also larger than the

RSS method, as the calculation of total processing time is made multiple times. It is therefore revealed that the computational cost of RSS method is the least.

9. Conclusion and future scope

This work is motivated from the various criteria of timeliness that provide services to the users of computer and network systems including response time, waiting time, turn-around time, elapsed time etc. To provide uniform response to the users, i.e., to minimize the variance of response time by minimizing the variance of access time is the problem of task scheduling by minimizing WTV as a measure in single processor and extend to parallel processors. In other words, a step has been taken for developing a scheduling procedure that minimizes the WTV of the individual task.

In task scheduling problems, a lot of works are done on the area of completion time rather than waiting time. Variance as a parameter is introduced by the researcher to minimize the CTV by distributing the task processing time in such a way that the uniformity among the task is obtained (i.e., QoS). For obtaining the uniformity in the scheduling problems, variance of completion time is more effective rather than the completion time. It was also found that the sequence that minimizes the variance of completion time is antithetical to the sequence that minimizes the variance of waiting time. But it was found from the literature that a large number of works are done on CTV, and in case of WTV, it is few.

The aim of this work is to analyze, study the peculiarity behavior, and develop efficient heuristic methods for solving different classes of scheduling problems. As the addressed problems are NP hard, the alternative of using heuristic methods has been proven to be good one, whereas the exact solution always gives optimum solution by taking maximum time for both single processor as well as parallel processors for a large set of tasks.

In these respects at first, basic elements of classical deterministic scheduling problem, different aspects related to scheduling problem and algorithms, and classification of scheduling problems are presented. Second, different methods for solving scheduling problems, complexity of scheduling problem, and basic knowledge on different schedule class are discussed. At last, an overview on different objective classification criteria for both single processor and parallel processors was presented.

Using the aforementioned background, a mixed integer programming model with two scheduling problems was addressed:

- A single processor scheduling problem *Minimize* (1||*WTV*) for minimizing WTV was stated and solved in section 7 by using five heuristic methods namely as EC1, EC2, VS, BS, and RSS.
- The processing time of tasks are generated randomly by three probability distributions namely normal distribution, Poisson distribution, and exponential distribution.
- Performances of five heuristic methods are analyzed. It was observed that RSS method gives optimum or near optimum results than other heuristic methods

- From the comparative result, it was also observed that the obtained WTV of the sequence generated with the help of heuristic methods are always satisfying the V-shaped optimality property.
- It was also observed that RSS method gives results with minimum computational cost than other heuristic methods.
- A uniform parallel processor scheduling problem *Minimize* ($Q_m|prec|WTV$) for minimizing WTV was proposed and solved in Section 8 by using a RSS method in combination with a unique proposed task allocation scheme named as PUM.
- A unique task allocation scheme was developed for allocating the task to individual processor.
- The processing time of tasks are generated randomly by four probability distributions namely normal distribution, Poisson distribution, exponential distribution, and uniform distribution.
- Performance of measure of three heuristic methods namely as VS, BS, and RSS are analyzed by using a unique task allocation scheme named as PUM.
- The experimental results are compared and observed that RSS method with PUM allocation scheme reveals the best solution with minimal computational cost.

Therefore, it is concluded that in case of single processor, the computational cost of RSS heuristic method is less than the other four heuristic methods. In case of uniform parallel processor, the RSS method with PUM allocation scheme reveals the optimum or near optimum solution with minimal computational cost.

Often new computer systems and new performance measures used to evaluate a system lead to new directions in scheduling. The environment of scheduling is changing time to time depending on resource availability, interruptions, and nature of changed demand. New scheduling is to be prepared in between an old unprocessed schedule. This give rise to change in constraints and resources. This has to be rescheduled with changed objectives.

In future, keeping WTV as the measure of performance the following works will be carried out for finding the suitability and effectiveness of the heuristic methods and task allocation schemes proposed in this work.

- To apply the proposed work for available multiobjective scheduling problems.
- To apply the proposed work in order to investigate the field of tasks and resources allocation in project like project management scheduling, broadcast scheduling, etc.
- To find out the effect of these proposed work in dynamic scheduling.
- Exploration of more efficient scheduler with better effective scheduling methods.
- Use of stochastic scheduling problems in real life environment.
- Suitability of techniques with cloud computing which is a kind of grid with virtual services and service oriented architecture (SOA).

Author details

Satyasundara Mahapatra[1]*, Rati Ranjan Dash[2] and Sateesh K. Pradhan[3]

*Address all correspondence to: satyasundara123@gmail.com

1 Indian Institute of Science and Information Technology, Bhubaneswar, India

2 College of Engineering and Technology, Bhubaneswar, India

3 Utkal University, Bhubaneswar, India

References

[1] Graham RL. Combinatorial scheduling theory. In: Steen LA, editors. Mathematics Today. Vol. 3. Berlin: Springer; 1978. pp. 183–211.

[2] Graham RL. The combinatorial mathematics of scheduling. Scientific American. 1978;**238**: 124–132.

[3] Katona GOH. Combinatorial search problems. In: Srivastava JN, editors. A Survey of Combinatorial Theory. Vol. 28. Amsterdam: North-Holland Publ. Co; 1973.

[4] Garey MR, Johnson DS. Computers and Intractability, A Guide to the Theory of NP-Completeness. New York, USA: W.H. Freeman and Company; 2000.

[5] Williamson DP, Shmoys DB. The Design of Approximation Algorithm. Vol. 3. Cambridge, UK: Cambridge University Press; 2010.

[6] Djuraskovic I, Arthur N. Heuristic inquiry: A personal journey of acculturation and identity reconstruction. The Qualitative Report. 2010;**1**(6): 1569–1593.

[7] Nong Ye, Li X, Farley T, Xu X. Job scheduling methods for reducing waiting time variance. Computer & Operations Research (Elsevier). 2007;**18**:3069–3083.

[8] Vincent AC, Stephen F Smith. Heuristic Selection for Stochastic Search Optimization: Modeling Solution Quality by Extreme Value Theory. 10th International Conference, CP, Toronto, Canada, Proceedings, Vol. 3. 2004. pp. 197–211.

[9] Nessah R, Chu C. A lower bound for weighted completion time variance. European Journal of Operational Research. 2010;**10**(3):1221–1226.

[10] Eilon S, Chowdhury IG. Minimizing waiting time variance in the single machine problem. Management Science. 1977;**34**(6):567–575.

[11] Mahapatra S, Dash RR, Pradhan SK. An approach to solve single machine job scheduling problem using heuristic algorithm. International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS), ISSN (Online): 2279-0055. 2015;**11**(2):157–163.

[12] Poursalik K, Miri-Nargesi S. Meta-heuristic approaches for a new modeling of single machine scheduling problem. Scientific Khyber. 2013;**55**(2):107–117.

[13] Socha K, Knowles J, Sampels M. A MAX-MIN ant system for the university timetabling problem. In: Dorigo M, Di Caro G, Sampels M (editors). Ant Algorithms: Third International Workshop, ANTS 2002. Lecture Notes in Computer Science. Vol. 16; 2002. pp.1–13.

[14] Dorigo M, Bonabeau E, Theraulaz G. Ant algorithms and stigmergy. Future Generation Computer. 2000;**16**(8):851–871.

[15] Asadzadeh L, Zamanifar K. An agent-based parallel approach for the job shop scheduling problem with genetic algorithms. Mathematical and Computer Modelling. 2010;**52** (11–12): 1957–1965.

[16] Baudet P,Azzaro C, Pibouleau L, Domenech S. A genetic algorithm for batch chemical plant scheduling. Proc. Int. Congress of Chemical and Process Engineering. 1996:pp. 25–30.

[17] Cardon A, Galinho T, Vacher JP. A multi-objective genetic algorithm in job shop scheduling problem to refine an agents architecture. In Proceedings of EUROGEN'99. Jyvaskyla, Finland. University of Jyvsaskyla; 1999.

[18] Zhai Y, Liu C, Chu W, Guo R, Liu C. A decomposition heuristics based on multi-bottleneck machines for large-scale job shop scheduling problems. Journal of Industrial Engineering and Management (JIEM). 2014;**177**(5):1397–1414.

[19] El-Bouri A, Azizi A, Zolfaghari S. A comparative study of a new heuristic based on adaptive memory programming and simulated annealing: The case of job shop scheduling. European Journal of Operational Research. 2007;**155**:1894–1910.

[20] Mahapatra S, Dash RR, Pradhan SK. A heuristic for scheduling of uniform parallel processors. 2nd International Conference on Computational Intelligence and Networks (CINE), KIIT University, Bhubaneswar, Odisha. IEEE Xplore Digital Library. 11 Jan 2016:78–83.

[21] Cai X, Cheng TCE. Multi-machine scheduling with variance minimization. Discrete Applied Mathematics. 1998;**128**(1–3):55–70.

[22] Jozefowska J, Mika M, Rozycki R, Waligora G, Weglarz J. An almost optimal heuristic for preemptive C_{max} scheduling of dependent tasks on identical parallel processors. Annals of Operation Research. 2004;**129**(129):205–216.

[23] Rafsanjani MK, Bardsiri AK. A new heuristic approach for scheduling independent tasks on heterogeneous computing systems. International Journal of Machine Learning and Computing. 2011;**2**(4):371–376.

[24] Xu X, Ye N. Minimization of job waiting time variance on identical parallel machines. IEEE transactions on Systems, Man, and Cybernetics–Part C: Applications and Reviews. 2007;**37**(5):917–927.

[25] Merten AG, Muller ME. Variance minimization in single machine sequencing problems. Management Science. 1972;**38**(9):518–528.

[26] Pinedo M. Scheduling theory, algorithms and systems. Englewood Cliffs, NJ: Prentice-Hall; 1995.

[27] Schrage L. Minimizing the time-in-system variance for a finite jobset. Management Science. 1975;**207**(5):540–543.

[28] Hall NG, Kubiak W. Proof of a conjecture of Schrage about the completion time variance problem. Operations Research Letters. 1991;**27**:467–472.

IntechOpen