Flexible Flow Shop Scheduling with Stochastic Processing Times: A Decomposition-Based Approach

S.H. Choi[a] and K. Wang[b]

Abstract

Flexible flow shop scheduling problems are NP-hard and tend to become more complex when stochastic uncertainties are taken into consideration. Although some methods have been developed to address such problems, they remain inherently difficult to solve by any single approach. This paper presents a novel decomposition-based approach (DBA), which combines both the Shortest Processing Time (SPT) and the Genetic Algorithm (GA), to minimizing the makespan of a flexible flow shop (FFS) with stochastic processing times. In the proposed DBA, a neighbouring K-means clustering algorithm is developed to firstly group the machines of an FFS into an appropriate number of machine clusters, based on their stochastic nature. Two optimal back propagation networks (BPN), corresponding to the scenarios of simultaneous and non-simultaneous job arrivals, are then selectively adopted to assign either SPT or GA to each machine cluster for sub-schedule generation. Finally, an overall schedule is generated by integrating the sub-schedules of machine clusters. Computation results show that the DBA outperforms SPT and GA alone for FFS scheduling with stochastic processing times.

Keywords: flexible flow shop; scheduling; stochastic processing times; decomposition; neighbouring K-means clustering; back propagation network

[a] Department of Industrial and Manufacturing Systems Engineering,
The University of Hong Kong, Pokfulam Road, Hong Kong.
Email: shchoi@hku.hk

[b] Department of Management Science and Engineering,
Economics and Management School, Wuhan University, Wuhan, China.
Email: kai.wang@whu.edu.cn

1. Introduction

Production scheduling is a decision-making process to allocate limited resources, such as machines, material handling equipment, operators, and tools, to tasks or jobs to achieve certain objectives (Pinedo, 1995). Research efforts on production scheduling generally consider a static environment with a fixed number of jobs, deterministic processing times, and no unexpected events that would influence job processing when the schedule is executed. Real manufacturing is, however, dynamic and subject to a wide range of stochastic uncertainties, such as machine breakdown, stochastic processing times, rush order, etc. Therefore, production scheduling under uncertainty has indeed attracted much attention in recent years.

Three fundamental ways (Vidal, 2004; Aytug et al., 2005), including the completely reactive approach, the robust approach, and the predictive-reactive approach, have been employed to tackle production scheduling under uncertainty. The completely reactive approach can change decisions during execution when necessary. Although its computation cost is low, it uses only local information to generate a schedule which may not be globally optimal in nature. The robust approach constructs solutions by modelling uncertainties (O'Donovan et al., 1999; Liu et al., 2007) or optimizing the performance under different scenarios (Kouvelis et al., 2000). Such an approach can be viewed as a form of under-capacity scheduling in order to maintain the robustness under different scenarios. The predictive-reactive approach is a two-step process. First, a predictive schedule is generated over the time horizon considered. This schedule is then rescheduled during execution in response to unexpected disruptions.

Since each of these three approaches has its own strength and weakness, some early studies on comparative analysis of approaches in dynamic environments have been conducted. It was found that the completely reactive approach and the predictive-reactive approach were complementary. Lawrence and Sewell (1997) studied the static and dynamic applications of heuristic approach to job shop scheduling problems with uncertain processing times. Experiment results indicated that the predictive-reactive approaches based on global information were highly likely to perform better than the completely reactive approaches in an environment with little uncertainty. However, with increasing level of uncertainty, the global information might become invalid. As a result, the predictive-reactive approaches tend to give poorer results than the completely reactive approaches. Sabuncuoglu and Bayiz (2000) tested the reactive scheduling approaches under machine breakdown in a classical job shop system. They showed that online scheduling rules degraded less than offline scheduling algorithms in a stochastic environment. This conclusion was consistent with that of Lawrence and Sewell’s (1997).

In order to handle a complex environment, it is imperative to take advantage of mixing these three approaches to deal with uncertainty. Matsuura et al. (1993) developed a predictive approach on a periodic basis, called switching. The system switched to a dispatching rule for the remaining operations when the deviation between the predictive schedule and the realized one exceeded a certain level. A search of available literature reveals that few research works have been reported on solving complex scheduling problems by combined approaches.

In this paper, we aim to minimize the makespan of a flexible flow shop (FFS) scheduling problem with stochastic processing times. The FFS scheduling problem has been proven NP-hard in nature which is particularly difficult to solve (Garey, 1979; Gupta, 1988). Consideration of stochastic processing times further aggravates its complexity. Enlightened by the works of Matsuura’s (1993) and Lawrence’s (1997), we propose a decomposition-based approach (DBA) that combines two complementary approaches, namely the completely reactive approach and the predictive-reactive approach, to deal with stochastic processing times.

In the proposed DBA, a neighbouring K-means clustering algorithm firstly groups the machines of an FFS into several machine clusters based on their stochastic nature. Then, either the predictive-reactive approach or the completely reactive approach, determined by the process of approach assignment, is employed to generate a sub-schedule for each machine cluster. Finally, these sub-schedules are integrated to give an overall one. Thus, by combining two different scheduling approaches, DBA explores a new direction for future research in the field of scheduling under uncertainty.

The remaining part of this paper is organized as follows. The next section briefly reviews the literature on FFS scheduling under uncertainty. Section 3 is devoted to problem description. Section 4 describes the framework of DBA and explains it in details. To evaluate the effectiveness of DBA, simulation is conducted and the computation results are analyzed in Section 5. Finally, conclusions are summarized and some directions of future work are discussed in Section 6.

2. Literature review

Ever since the flexible flow shop (FFS) scheduling problem was identified in 1970’s

(Arthanari and Ramamurthy, 1971), it has attracted considerable attention during the past decades (Wang, 2005). An FFS, also called a hybrid flow shop (HFS), consists of a series of production stages, each of which has several functionally identical machines operating in parallel. All the jobs released to an FFS have to visit all the stages in the same order.

Corresponding to the approaches to production scheduling under uncertainty, the study of FFS scheduling under uncertainty can be classified into the same three categories, namely the completely reactive approach, the robust approach, and the predictive-reactive approach.

The completely reactive approach is characterized by its capability of real-time decision making. In this approach, no schedule is generated in advance, and decisions are made locally and can be changed during execution when necessary. The dispatching rule is a typical completely reactive approach, in which jobs are selected by sorting them according to predefined criteria. As the dispatching rule can find a reasonably good solution in a relatively short time, it plays a significant role in solving scheduling problems with dynamic nature.

Hunsucker and Shah (1994) compared dispatching rules in a constrained multiprocessor flow shop, and concluded that the Shortest Processing Time (SPT) algorithm was superior for the makespan and mean flow time criteria. Similarly, Rajendran and Holthaus (1999) studied the performance of dispatching rules in dynamic flow shops and job shops with stochastic job arrivals and stochastic processing times. The performance of a variety of dispatching rules was evaluated with respect to criteria related to flow time and tardiness of jobs. Experiment results implied that no single dispatching rule dominated in all criteria. In order to minimize the mean and the standard deviation of flow time, Andres et al. (2006) compared several dispatching rules in a dynamic FFS, in which jobs were assumed to arrive continuously and setup times were sequence-dependent. They found that the dispatching rules had relatively little impact on the mean and the standard deviation of flow time, compared to the number of simultaneous jobs and the configuration of batch size.

Although dispatching rules tend to be simple and fast, they cannot optimize the overall performance of a system. Therefore, the research focus of dispatching rules has been shifted from a single dispatching rule to a set of dispatching rules.

Tang et al. (2005) examined the dynamic FFS scheduling problem to minimize the average flow time, the average tardy time, and the percentage of tardy jobs. The arrival times of jobs were unknown in advance and job arrivals were assumed to follow a Poisson process. He developed a neural network approach to generating schedules. Experiment results indicated that the neural network approach consistently performed better than a single traditional dispatching rule, although it did not always perform best. Singh et al. (2007) introduced a multi-criteria methodology by swapping dispatching rules in a shop with dynamic nature. The swapping of dispatching rules was determined by the worst performance criteria in the performance measures. It was evaluated in the presence of machine breakdown and had been demonstrated to improve the system performance.

The robust approach is another possible way to address the FFS scheduling problem under uncertainty. It aims to generate a schedule to minimize the effect of disruptions when the schedule is implemented.

Wang et al. (2005a) presented a class of hypothesis-test-based genetic algorithms to address flow shop scheduling problems with stochastic processing times. In the proposed algorithm, solutions were generated by the Genetic Algorithm (GA) and evaluated by multiple independent simulations. In order to improve the effectiveness of exploring the search space, the hypothesis test was performed to discard the solutions with no significant difference. They demonstrated the effectiveness of the proposed algorithm by comparison with the traditional GA. For the same problem, Wang et al. (2005b) also proposed a genetic ordinal optimization algorithm to combine ordinal optimization and optimal computing budget allocation. Such combination could achieve not only a better solution quality, but also the robustness of solutions to stochastic optimization problems.

Gholami et al. (2009) proposed a heuristic to solve FFS scheduling problems with sequence-dependent setups and stochastic machine breakdown. This method employed the random key genetic algorithm to identify the optimal solution. A simulator, using event-driven policy and right-shift heuristic approach, was incorporated into the genetic algorithm to evaluate the expected makespans. The robustness of the algorithm was analyzed using the Taguchi parameter design. The number of jobs, the number of stages, the mean-time-between-failure, and the population size were found to have significant impact on the robustness of the algorithm. Furthermore, the mean-time-to-repair was an adjustment factor that had significant impact on the mean makespan.

In response to unexpected disruptions, the predictive-reactive approach is by far the most studied to reschedule dynamic manufacturing systems. Two issues, when and how to react to disruptions, have to be addressed.

For the first issue, three policies, namely periodic, event driven, and hybrid, have been introduced in the literature (Church and Uzsoy, 1992; Vieira et al., 2003). The periodic policy updates the schedule for a fixed interval based on the status of the shop. For the event driven policy, rescheduling is trigged by the disruptions instead of by time intervals. A hybrid policy reschedules periodically as well as when a disruption arises. Vieira et al. (2000) investigated the performance of three rescheduling strategies for parallel machine systems with machine failure. They found that although all the three strategies exhibited similar performance, the hybrid strategy decreased flow time slightly for it performed rescheduling more often.

To address the second issue, the most common rescheduling methods include the completed scheduling, the right-shift schedule repair, and the partial schedule repair (Abumaizar and Svestka, 1997; Sabuncuoglu and Bayiz, 2000; Vieira, et al., 2003). The completed scheduling regenerates a completely new schedule for all the unprocessed operations. The right-shift schedule repair postpones the remaining operations by the amount of time needed to make the schedule feasible. The partial schedule repair only reschedules the operations that are affected by the disruption. Akturk and Gorgulu (1999) proposed a match-up scheduling approach to scheduling a modified flow shop. They rescheduled the initial schedule between the disruption and match-up time when machine breakdown occurred.

Although the completed scheduling can construct a better solution in theory, it is rarely applied in practice due to its high computation burden and increasing scheduling instability (Sabuncuoglu and Kizilisik, 2003; Liu, et al., 2007). Conversely, the right-shift schedule repair yields the least scheduling instability with the lowest computation effort, while the partial schedule repair is a moderate one in this regard.

It can be said that the techniques reported in available literature on FFS scheduling under uncertainty were mostly based on a single approach, yielding some initial yet limited performance. This paper therefore attempts to further address this issue with an integrated approach that combines and takes advantage of the completely reactive approach with the predictive-reactive approach.

3. Problem description

Machines of an FFS are arranged into stages in series. At each stage, a number of functionally identical machines operate in parallel, and a job has to be processed on one of these machines. In a real manufacturing environment, the processing times might be highly uncertain due to quality problems, equipment downtime, tool wear, and operator availability (Lawrence and Sewell, 1997).

The stochastic processing time can be described as the sum of the expected processing time E(P) and the standard deviation σ. The coefficient of processing time variation (CPTV), defined as $CPTV = \sigma / E(P)$, can be used as an indicator to processing time uncertainty; it equals 0 when the processing times are deterministic, and increases with the uncertainty. The

CPTV is applied to model stochastic processing times in this study.

In order to simplify a typical FFS scheduling problem with stochastic processing times, the following assumptions are made: (1) Preemption is not allowed for job processing; (2) Each machine can process at most one operation at a time; (3) All jobs are released simultaneously at the first stage; (4) All machines are available when jobs are released to the FFS; (5) There is no travel time between machines; (6) There is no setup time for job processing; (7) Infinite buffers exist for machines; (8) For the same job, the expected processing time at any parallel machine at a stage is identical; (9) The actual processing time of a job on a machine is uncertain, and it can be longer or shorter than the expected one; (10) As parallel machines in a stage are functionally identical, they lead to the same CPTV when processing any jobs, but the CPTV may be different for the machines at other stages; (11) The actual processing time of a job on any machines at a stage follows the gamma distribution. Its mean value is the expected processing time, E(P), and the standard deviation is $\sigma = CPTV \times E\left(P\right)$; (12) Except for stochastic processing times, there are no other types of uncertainties to disturb job processing.

The FFS scheduling objective under consideration is to determine the processing sequence of operations on each machine such that the makespan, which is equivalent to the completion time of the last job to leave the FFS, is minimized without violating any of the assumptions above. This FFS scheduling problem can also be described as follows:

$$\min\{\max[C_{tj}]\} \tag{1}$$

Subject to the following constraints:

$$C_{1j} = P_{1j},\ if \quad \sum_{i=1}^{m_1} U_{1ij} > 0 \tag{2}$$

$$C_{1j_2} = \sum_{i=1}^{m_1} \sum_{j_2=1}^{n} \left(B_{1ij_1j_2} \times C_{1j_1} \right) + P_{1j_2},\ if \quad \sum_{i=1}^{m_1} U_{1ij_2} = 0 \tag{3}$$

$$C_{kj} = C_{(k\text{-}1)j} + P_{kj}, \quad if \quad k > 1 \ \ \& \ \sum_{i=1}^{m_k} U_{kij} > 0 \tag{4}$$

$$\begin{gathered} C_{kj_2} = \max\{ \sum_{i=1}^{m_k} \sum_{j_2=1}^{n} \left(B_{kij_1j_2} \times C_{kj_1} \right),\ C_{(k\text{-}1)j_2} \ \} + P_{kj_2}, \\ if \quad k > 1 \ \ \& \ \sum_{i=1}^{m_k} U_{kij_2} = 0 \end{gathered} \tag{5}$$

$$ST_{kij} \geq 0 \tag{6}$$

$$E\left(P_{ki_1j}\right) = E\left(P_{ki_2j}\right) = E\left(P_{kj}\right), \quad \left(i_1, i_2\right) \in M_k \tag{7}$$

$$ST_{(k+1)i_1 j} - ST_{ki_2 j} \geq P_{ki_2 j} \tag{8}$$

$$[(ST_{kij_1} - ST_{kij_2}) \geq P_{kij_2}] \quad or \quad [(ST_{kij_2} - ST_{kij_1}) \geq P_{kij_1}] \tag{9}$$

Where

k:	stage index, $1 \leq k \leq t$
m_k:	number of parallel machines at stage k
M_k:	set of parallel machines at stage k
i, i_1, i_2:	machine index, $1 \leq i, i_1, i_2 \leq m_k$
j, j_1, j_2:	job index, $1 \leq j, j_1, j_2 \leq n$
C_{kj}:	completion time of Job j at stage k
$B_{kij_1j_2}$:	Boolean variable, 1 if Job j_2 is scheduled immediately after Job j_1 on machine i at stage k, and 0 otherwise
U_{kij}:	Boolean variable, 1 if Job j is the first job on machine i at stage k, and 0 otherwise
$E(P_{kj})$:	expected processing time of Job j at stage k
$E(P_{kij})$:	expected processing time of Job j on machine i at stage k
P_{kj}:	stochastic processing time of Job j at stage k
P_{kij}:	stochastic processing time of Job j on machine i at stage k
ST_{kij}:	start time of Job j on machine i at stage k

For the first stage, constraints (2) and (3) give the completion time of the first job and that of each subsequent job on the machines, respectively. Similarly for all other stages, constraints (4) and (5) determine the completion time of the first job and that of each subsequent job on the machines, respectively. While constraint (6) ensures non-negative start time of job processing, constraint (7) stipulates that each of the parallel machines at a stage takes equal time to process the same job. Lastly, constraint (8) requires the processing sequence of each stage to satisfy the processing time, and constraint (9) guarantees that each machine can process only one job at a time.

4. The proposed decomposition-based approach (DBA)

4.1 The framework of DBA

The predictive approaches are likely to perform better than the completely reactive approaches in a low stochastic environment, but they may lead to poor result in one with a high stochastic nature (Lawrence and Sewell, 1997; Sabuncuoglu and Bayiz, 2000). Based on this observation, a decomposition-based approach (DBA) is proposed to provide better performance for FFS scheduling problems in any stochastic environment.

As shown in Figure 1, the DBA framework consists of two modules. An original FFS scheduling problem is firstly broken down into sub-problems. A sub-problem, termed as a cluster scheduling problem, aims to schedule the machines within a machine cluster and abides by the same assumptions for the FFS scheduling problem. After generation of cluster scheduling problems, either the predictive-reactive approach or the completely reactive approach is selectively assigned to solve these problems. The solutions are subsequently integrated to provide an overall schedule. The major feature of DBA is its decomposition strategy – combining and taking advantage of different approaches to generate a better result when scheduling in any stochastic environment.

[Insert Figure 1 here]

Figure 2 shows a typical decomposition result of an FFS with 7 stages and 3 parallel machines at each stage. Geometric figures with the same shape represent parallel machines in a stage. The FFS is decomposed into three machine clusters. One of the two approaches, the predictive-reactive approach or the completely reactive approach, is selected to schedule the machines in each machine cluster.

[Insert Figure 2 here]

4.1.1 Generation of cluster scheduling problems

Cluster scheduling problems are generated by decomposing an FFS with a clustering algorithm into machine clusters, each of which contains a number of machines sharing a similar stochastic nature during job processing. As the actual processing times of jobs on a machine may be uncertain, the processing time uncertainty is used to describe the stochastic nature of a machine.

Clustering is the classification of objects into different groups, such that the objects in each group would share some common traits. Quite a few algorithms, such as K-means, fuzzy C-means, and self-organization maps etc, have been proposed to perform classification. Among these clustering algorithms, K-means (MacQueen, 1967) is relatively simple and widely used in data mining, in that it groups *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. For this reason, K-means is further developed to form a neighbouring K-means clustering algorithm to decompose an FFS.

After decomposition of an FFS, machines in the same machine cluster share a similar stochastic nature and can be scheduled by the same approach. A machine cluster with a low stochastic nature is solved by the predictive-reactive approach, while one with a high stochastic nature is scheduled by the completely reactive approach.

4.1.2 Solving cluster scheduling problems and solution integration

The cluster scheduling problems, produced by decomposition of an FFS, need to be solved with appropriate approaches. Due to their better performance and being widely used, GA and SPT are identified as the predictive-reactive approach and the completely reactive approach, respectively. GA tends to give poorer result than SPT when scheduling machines with a high stochastic nature.

In order to assign an appropriate approach to a machine cluster, it is essential to establish an effective model to estimate the makespan difference (MDSG) of the schedules generated by SPT and GA. Artificial neural networks (ANNs) have been widely used in various areas due to its capability of identifying complex nonlinear relationships between input and output. The back propagation network (BPN) is a commonly used ANN structure and has been successfully applied for system modelling, prediction, and classification (Lin and Hwang, 1999). It is therefore adopted to estimate the MDSG for each machine cluster, and the positive or negative sign of the MDSG determines the approach to be assigned to the machine cluster.

After approach assignment above, the sub-schedule for each of the machine clusters can be generated by either GA or SPT, and subsequently integrated into an overall schedule.

4.2 Details of the DBA

4.2.1 FFS decomposition

To decompose an FFS scheduling problem, the machines of an FFS are grouped into a few machine clusters in which machines share a similar stochastic nature. The stochastic nature of a machine results from the uncertainties which occur during job processing and when a schedule deviates from the planned one. Since the CPTV represents processing time uncertainty, it is adopted to describe the stochastic nature of machines. As assumed in Section 3 that the CPTV is identical for all parallel machines at the same stage, a stochastic vector U_i

can be formed to describe the stochastic nature of parallel machines at Stage i, giving

$$U_i = [CPTV_i] \tag{10}$$

where $CPTV_i$ is the CPTV of the parallel machines at Stage i. A machine with a larger CPTV indicates a high stochastic nature of job processing.

Based on the stochastic vector U_i, we can measure the difference of the stochastic nature (DSN) between two stages of parallel machines by computing the distance of their stochastic vectors. As the Euclidean distance is one of the most commonly used methods to measure the distance between a pair of data, it serves to define the DSN between two stages of parallel machines, which is calculated as follows:

$$D(U_i,U_j) = \|U_i - U_j\|_2 = \sqrt{(CPTV_i - CPTV_j)^2} \tag{11}$$

where U_i and U_j are the stochastic vectors of parallel machines at Stages i and j, respectively, and $D(U_i,U_j)$ is the DSN between U_i and U_j. For the parallel machines at Stages i and j, the larger the $D(U_i,U_j)$, the more is their dissimilarity and the less likely of their being in the same machine cluster.

Although $D(U_i,U_j)$ is useful for traditional clustering, it is not sufficient for decomposing an FFS into machine clusters. In this study, a machine cluster is a group of machines which share a similar stochastic nature. As illustrated in Figure 2 in Section 4, a machine cluster exhibits two basic characteristics, namely:

- Parallel machines at a stage are allocated to the same machine cluster since they have the same CPTV during job processing;
- Stages in the same machine cluster are in series.

Such two characteristics make the decomposition of an FFS different from the traditional clustering problem. For the traditional clustering problem, data objects can be clustered based only on their similarity distance and no other rules would need to be followed. Therefore, the traditional K-means clustering algorithm might not be able to group the machines of an FFS into appropriate machine clusters simply based on $D(U_i,U_j)$.

Indeed, there are two problems in applying the traditional K-means clustering algorithm to decompose an FFS, namely (1) how to group the machines into machine clusters, in addition to considering $D(U_i,U_j)$, and (2) how to choose a suitable machine cluster number. To address these two problems, a neighbouring K-means clustering algorithm, involving both a machine allocation algorithm and weighted cluster validity indices, is proposed.

For the first problem, a machine allocation algorithm is developed to allocate machines to each machine cluster. As the parallel machines at the same stage are assumed to have the same CPTV, each machine cluster centre is a set of parallel machines at a stage represented by the stage ID of the parallel machines. The set of machine cluster centres is defined as C = $\{S_1, \dots, S_k, \dots, S_n\}$, where S_k is the stage ID of the kth machine cluster centre. The machine cluster centres of set C are sequenced in ascending order of their stage IDs. Based on the concept of machine cluster centre, the machine allocation algorithm, which can group machines in accordance with the characteristics of a machine cluster, is outlined in Table 1.

[Insert Table 1 here]

The machine allocation algorithm aims to allocate the parallel machines between two neighbouring machine cluster centres to one of them. In order to identify the optimal allocation, each possible allocation is measured by the total DSN, which is defined as the sum of all the DSNs from stages of parallel machines to their allocated machine cluster centres, and the allocation result with the minimal total DSN is the optimal one.

For the second problem, we propose an approach based on weighted cluster validity indices, which measure the intra-distances and the inter-distances between machine clusters, for choosing an appropriate machine cluster number. Traditionally, neither a small cluster number nor a large one can offer a satisfactory classification of the data objects. Recently, cluster validity indices (CVIs) have attracted much attention as an approach to determining the optimal cluster number. A validity index indicates how well the clustering algorithm classifies a given data set. In order to evaluate the clusters, most CVIs are defined by combining the intra-cluster distances and inter-cluster distances. The intra-distance measures the distances of objects within a cluster to represent its compactness. The inter-distance computes the distance between two different clusters; it is an indicator of cluster separability. Therefore, a good clustering algorithm should have small intra-cluster distances and large inter-cluster distances.

Kim and Ramakrishna (2005) classified the CVIs into two categories, namely the ratio type and the summation type. The ratio type is the ratio of the intra-cluster distance (IntraDis) to the inter-cluster distance (InterDis) or vice versa. Dunn (Dunn, 1973) and DB (Davies and Bouldin, 1979) are two typical approaches of the ratio type, which can be formulated as $Validity = InterDis / IntraDis$ and as $Validity = IntraDis / InterDis$, respectively. The summation type is the weighted sum of the intra-cluster distance and the inter-cluster distance. It can be described as $Validity = IntraDis + \lambda \times InterDis$. Examples of this kind include Vsv (Kim et al., 2001) and DVI (Shen et al., 2005). Any one of these four types of CVIs in Table 2 can be applied to evaluate the decomposition result of an FFS. A small value of DB, Vsv and DVI indicates a good clustering, while Dunn prefers a large value for a good clustering.

[Insert Table 2 here]

Since we aim to schedule neighbouring machine clusters by different approaches, a good clustering algorithm should encourage large inter-cluster distances between neighbouring machine clusters rather than that between non-neighbouring machine clusters. For this purpose, we give the weight W_{ij} to the inter-distance between two machine clusters, which is described as:

$$W_{ij} = \frac{1}{\left(F_i + F_j\right)} \tag{12}$$

where F_i is the first stage of ith machine cluster. Taking Figure 2 as an example, F_1, F_2, and F_3 equal 1, 3, and 6, respectively.

Based on the weight W_{ij}, the weighted inter-distance between machine clusters i and j can be computed as $WeightedInterDis\left(C_i, C_j\right) = W_{ij} \times InterDis\left(C_i, C_j\right)$. Accordingly, integrated with the weighted inter-distance, four weighted Dunn, DB, Vsv and DVI (denoted by W-Dunn, W-DB, W-Vsv and W-DVI, respectively) are proposed in Table 2. The weighted Dunn and DB can be described as $Validity = IntraDis / WeightedInterDis$ and as $Validity = WeightedInterDis / IntraDis$ respectively, while the weighted Vsv and DVI can be formulated as $Validity = IntraDis + \lambda \times WeightedInterDis$.

In Table 2, $InterDis\left(C_i, C_j\right)$ represents the inter-distance between cluster i and j; $IntraDis\left(C_k\right)$ denotes the intra-distance of cluster i; C_i and c_i represent the cluster i and its cluster centre, respectively; nc is the number of clusters; n_k is the number of data in cluster k, while N is the total number of data; K is the pre-defined upper bound number of the clusters.

Having addressed the two problems in applying the traditional K-means clustering algorithm, we now propose a neighbouring K-means clustering algorithm in Table 3, which incorporates the machine allocation algorithm and the weighted CVI, to decompose an FFS into machine clusters without predefining the machine cluster number.

[Insert Table 3 here]

The neighbouring K-means clustering algorithm involves two steps: decomposition of an FFS and evaluation of the decomposition results. Firstly, an FFS is decomposed into different number of machine clusters by the K-means clustering algorithm, in which each machine is assigned to its nearest machine cluster centre by the machine allocation algorithm, rather than simply on their distance. The optimal decomposition result is obtained by searching for the optimal weighted CVI, which means its minimum value for the weighted DB, Vsv, or DVI, and its maximum value for the weighted Dunn.

4.2.2 The back propagation network for approach assignment

After decomposition of an FFS, the machine clusters can be scheduled either by SPT or by GA. The assigned approach for each machine cluster is determined by the makespan difference of the schedules generated by SPT and GA (MDSG), giving

$$MDSG = \left(M_{SPT\_S} - M_{GA\_S}\right) \big/ M_{GA} \tag{13}$$

where M_{SPT_S} and M_{GA_S} are the makespans with stochastic processing times generated by SPT and by GA, respectively, while M_{GA} is the makespan with deterministic processing times generated by GA.

A positive MDSG reveals that the makespan by SPT is longer than that by GA, implying the machines are with a low stochastic nature and are more suitable to schedule by GA. On the other hand, a negative MDSG indicates that the makespan by SPT is shorter than that by GA, implying the machines are with a high stochastic nature and are more suitable to schedule by SPT.

In order to accurately estimate the MDSG for each machine cluster, the back propagation network (BPN) is adopted in this study. Under the assumptions we made in Section 3 for the FFS scheduling problem, jobs are released simultaneously at the first stage. At the subsequent stages, jobs are allocated by the first-in-first-out (FIFO) rule and may not arrive simultaneously. Therefore, two scenarios have to be considered when establishing models to predict the MDSG. The first scenario assumes the jobs to be released simultaneously, while the other one allows the jobs arrive non-simultaneously.

Accordingly, two types of BPNs, each corresponds to a scenario, are needed to be established. Their architectures are identical, as illustrated in Figure 3. The details of BPN establishment for each scenario are as follows:

- Inputs: four parameters, namely CPTV, stage size, job size, and parallel machine size, are found to affect the performance of MDSG significantly according to the experiment results in Section 5. The first input is the mean of the CPTVs for all the machines in a machine cluster. All the four inputs are normalised in the range of [0, 1] for input into a BPN.
- Number of single hidden layers: Generally one hidden layer is capable of approximating any function with a finite number of discontinuities (Ripley, 1996; Chang et al., 2008). Therefore, a BPN only consists of one hidden layer.
- Number of hidden neurons: 2-20. In engineering applications, the optimal numbers of hidden neurons vary with the problems to be solved. There is no concrete rule to find the optimal number. If inadequate hidden neurons are adopted, it may introduce a greater risk of modelling the complex data poorly. If too many hidden neurons are used, the network may fit the training data well but would perform poorly on new and unseen data. For these reasons, the number of hidden neurons is determined experimentally by trial and error. As the number of hidden neurons is usually not recommended to be more than twice the input layer size (Priddy and Keller, 2005), it is intentionally chosen from the interval [2, 20] in this study. For each scenario, the BPNs with different number of hidden neurons are generated and evaluated by the minimum mean square error (MSE), and the one that corresponds to the number of hidden neurons that give rise to the least minimum MSE is termed the optimal BPN.
- Output: MDSG. For the same FFS scheduling problem with stochastic processing times, GA and SPT are used to obtain the makespan by the simulation, respectively. The MDSG is subsequently computed by Equation (13).
- Number of epochs per replication: 10000.
- Number of replications: 100. The performance of a BPN is sensitive to the initial network conditions. Therefore, for a specific number of hidden neurons, 100 BPNs with different initial conditions will be trained and evaluated respectively. Among these BPNs, only the one with minimum MSE is kept for the purpose to further identify the optimal BPN.
- Training examples: For each scenario, training examples are generated from experimental FFS scheduling problems (to be presented in Section 5.2.1) to establish the BPNs.

Both the number of epochs per replication and the number of replications are selected empirically to ensure generation of BPNs with satisfactory performance within an acceptable training time.

[Insert Figure 3 here]

After training, validating and testing the BPNs with different number of hidden neurons, the optimal one for each scenario can be identified and used to determine the MDSG. The optimal BPN generated in the scenario of simultaneous job arrivals is used to compute the MDSG of the first machine cluster, while the optimal BPN established in the scenario of non-simultaneous job arrivals is adopted to estimate the MDSG of all other machine clusters. If the MDSG of a machine cluster is predicted to be positive, GA is allocated to address the scheduling problem of the machine cluster. Otherwise, SPT is used to generate the schedule for the machine cluster.

However, the neighbouring K-means clustering algorithm cannot avoid the possibility that two neighbouring machine clusters are to be solved by the same approach. Therefore, it is reasonable to conduct a cluster merging process (CMP) to integrate neighbouring machine clusters, if necessary, using the following steps: (1) Identify the two neighbouring machine clusters which are to be solved by the same approach; (2) Merge the two neighbouring machine clusters and determine the approach for the new machine cluster by optimal BPNs; (3) Repeat Steps 1 and 2 until any two neighbouring machine clusters are allocated with different approaches.

Integrating with CMP, the complete process of approach assignment is summarized as follows: (1) Generate the training examples for both scenarios; (2) Establish the BPNs with different number of hidden neurons for both scenarios, and identify the optimal one for each scenario; (3) Estimate the MDSG for each machine cluster by optimal BPNs; (4) Assign either GA or SPT to each machine cluster according to the positive or negative sign of its estimated MDSG, respectively; (5) Conduct CMP.

4.2.3 Cluster scheduling

After decomposition and approach assignment, schedules are generated by either GA or SPT for all machine clusters and then integrated into an overall solution.

SPT performs better with low computation cost when the machines in a machine cluster have a high stochastic nature. It consists of the following two main steps: (1) Determine the job sequence based on the SPT rule for the first stage; (2) Allocate the finished job from the previous stage to the current stage by the FIFO rule until all the jobs are processed at each stage.

GA is used prior to the dispatching rules when scheduling a machine cluster with a low stochastic nature. Similar to the GA proposed by Jungwattanakit (2008), the overall structure of the GA in this study is briefly described as follows: (1) Coding: The job sequence is widely used as the chromosome for the FFS scheduling problem. Integer coding scheme is adopted for chromosome representation. For example, job sequence [2, 3, 5, 1, 4, 9, 8, 6, 7, 10] is a chromosome with ten jobs in an FFS; (2) Objective function evaluation: For the purpose of minimizing the makespan, the fitness function is formulated as $fitness = C_{max}$, where C_{max} is the maximum completion time of jobs at the last stage in an FFS. (3) Selection strategy: Roulette wheel selection is applied to reproduce the next generation; (4) Crossover and mutation operations: Order preserved crossover (OPX) and shift move mutation (SM) are adopted; (5) Crossover and mutation rates: The crossover rate and the mutation rate are analyzed by setting different values on the same FFS scheduling problem. A crossover rate of 0.8 and a mutation rate of 0.2 are found to give good performance in this study; (6) Termination criterion: The algorithm continues until 200 generations have been examined. This value is chosen empirically.

4.2.4 Rescheduling strategies of SPT, GA, and DBA

The performance of SPT, GA, and DBA are compared with consideration of processing time uncertainty in this study. In order to handle the job processing delay caused by stochastic processing times, rescheduling strategies of these three approaches are needed to be identified respectively.

SPT is a completely reactive approach and the job allocation follows an initial sequence at the first stage and FIFO rule at all other stages. Therefore, there is no need to establish a reschedule scheme for SPT. For GA, in order to reduce computation effort while not increasing schedule instability significantly, the right-shift schedule repair is used to deal with job processing delay. The operations affected are postponed without changing the job sequence in comparison with that of the schedule with deterministic processing times. In the proposed DBA, the right-shift schedule repair is adopted for machine clusters to be solved by GA, while FIFO rule is applied to those by SPT.

5. Computational results and analysis

5.1 Design of experiments

Three experiments are designed and conducted for performance evaluation of the proposed DBA. The first experiment generates optimal BPNs for the MDSG estimation. The second experiment compares the proposed weighted CVIs to find out which one is the most suitable to decompose an FFS. After these two experiments, the proposed DBA is well-prepared to optimize an FFS scheduling problem, and lastly the third experiment analyzes its performance based on makespan criterion. All these experiments have been implemented in Java and run on a PC with Intel Pentium 4 2.80GHz processor and 1.00GB of RAM.

In the experiments above, the expected processing times of operations are generated uniformly in the time unit interval [1, 20] and their average value equals 10 time units, while the actual processing times are uncertain and follow the gamma distribution. We use CPTV, defined as $CPTV = \sigma / E(P)$, to indicate the degree of processing time uncertainty. For instance, if the expected processing time of a job E [P] = 30 time units and CPTV = 0.3, σ is 9 time units. Thus, the standard deviation of actual processing time from E [P] is 9 time units.

The second and the third experiments are conducted on a test-bed containing 27 FFS scheduling problems with different stages, jobs, and parallel machines, as shown in Table 8. The number of jobs is chosen to be 20, 30, and 40. The stage can be 6, 10, and 15. The number of parallel machines ranges from 2 to 4. For each FFS scheduling problem, ten instances with different expected processing times of operations are randomly generated and the simulation is iterated 50 times for each instance.

5.2 Experiment results and discussion

5.2.1 Experiment I: generation of optimal BPNs

In order to train, validate, and test the BPNs, it is necessary to generate training examples first. With regard to the two scenarios of simultaneous and non-simultaneous job arrivals, two sets of training examples are generated. The levels of four possible BPN inputs used in the experiments, including CPTV, stage size, job size, and parallel machine size, are shown in Table 4.

[Insert Table 4 here]

For each combination of BPN inputs, two experimental FFS scheduling problems to minimize makespans with stochastic processing times, corresponding to the two scenarios of simultaneous and non-simultaneous job arrivals, are generated, in each of which all the parallel machines share the same CPTV. The two experimental FFS scheduling problems are solved by GA and SPT, respectively. Subsequently, the MDSG, which is the output of BPN, can be obtained by Equation (13) for each problem. Thus, training examples can be generated by covering all the possible combinations of BPN inputs and their corresponding outputs. This results in a total of 3,600 ($10\times10\times6\times6 = 3{,}600$) training examples.

For a better analysis of the factors that affect the MDSG, we analyze the training examples by means of a multi-factor Analysis of Variance (ANOVA) technique, using a 5% significance level. A standard ANOVA table contains columns for the sums of squares (SS), degrees-of-freedom (DF), mean squares (MS=SS/DF), F-ratio (MS/MS residual), and P-values. Based on DF and F-ratio, P-value can be derived from the F-distribution. If the P-value of a factor is smaller than the significant level (5%), we can conclude that the factor significantly affects the MDSG.

Tables 5 and 6 list the statistical results generated by ANOVA for the scenarios of simultaneous and non-simultaneous job arrivals, respectively. From these tables, it is obvious that all the P-values are less than 0.05, indicating that the MDSG are significantly influenced by all the four factors in both scenarios. Therefore, it is reasonable to adopt these four factors as BPN inputs.

[Insert Tables 5 and 6 here]

Based on the data of the training examples, scatter plots are generated to help visualize the relationship between the MDSG and the four BPN inputs, including CPTV, stage size, job size and parallel machine size. As shown in Figure 4, circles and squares represent the results derived for the scenarios of simultaneous and non-simultaneous job arrivals, respectively. For a specific x-value in Figure 4, the y-value is the mean of MDSGs of all the corresponding training examples. Accordingly, the following two observations can be made from the scatter plots:

- In general, the MDSG is approximately linearly proportional to CPTV, stage size, and job size, while it firstly decreases and then increases with increasing parallel machine size.

- The MDSG is different for the two scenarios of simultaneous and non-simultaneous job arrivals. Hence, two BPNs, each corresponds a scenario, are needed to estimate the MDSG.

It should be pointed out that the scatter plots in Figure 4 illustrate only the general trend of the average MDSGs of the training examples with various BPN inputs, and they should not be interpreted as to indicate whether SPT is superior to GA, or otherwise. Indeed, these scatter plots should not be used to predict the MDSG and determine the approach for machine clusters of a specific FFS scheduling problem in a particular scenario. Instead, the BPNs should be used for such purpose.

[Insert Figure 4 here]

Now, two types of BPNs, corresponding to the two scenarios of simultaneous and non-simultaneous job arrivals, can be obtained on the basis of the training examples. In order to identify the optimal BPN for each scenario, we establish BPNs with different number of hidden neurons and measure their prediction accuracy by MSE. Figure 5 shows the relationship of the minimal MSE with the number of hidden neurons for the two scenarios. It can be seen that the numbers of hidden neurons that give rise to the least MSEs for simultaneous and non-simultaneous job arrivals are 12 and 14, respectively. Accordingly, the two BPNs corresponding to these two numbers of hidden neurons are optimal ones and used to estimate the MDSGs.

[Insert Figure 5 here]

5.2.2 Experiment II: weighted CVI analysis

After the two optimal BPNs are established, either SPT or GA can be assigned to a machine cluster according to its estimated MDSG. Integrated with W-Dunn, W-DB, W-Vsv, and W-DVI respectively, the neighbouring K-means clustering algorithm is evaluated to determine which CVI can offer good clustering for all the problems of the test-bed. Table 7 gives the ratios of the average makespan of various CVIs to that of Dunn. It is obvious that each weighted CVI gives a smaller makespan than its corresponding CVI. Among the four weighted CVIs, W-DB achieves the best performance; it is thus recommended for FFS decomposition.

[Insert Table 7 here]

5.2.3 Experiment III: DBA analysis

In order to evaluate the effectiveness of the proposed DBA, SPT, GA, and DBA are analyzed in a stochastic environment in which CPTVs are uniformly distributed in the interval [0.1, 1]. The experiment results of these three approaches with stochastic processing times (denoted by SPT_S, GA_S and DBA_S respectively) are shown in Table 8. The results of SPT and GA with deterministic processing times (denoted by SPT_D and GA_D respectively) are also given. All the results are the ratios of the average makespan of various scheduling algorithms to that of GA.

[Insert Table 8 here]

From the experiment results, the following conclusions can be made: (1) SPT_S is superior to GA_S. The reason for such poor performance of GA lies in using the right-shift schedule repair when job processing delay occurs; (2) DBA_S performs better than SPT or GA in most cases. Compared with SPT_S and GA_S, the DBA_S reduces the makespan by about 3% and 13%, respectively. The better performance of DBA is due to its decomposition strategy, which combines the strengths of GA and SPT to deal with stochastic processing times.

6. Conclusion

A decomposition-based approach (DBA), which combines and takes advantage of the characteristics of SPT and GA, has been proposed to minimize the makespan of a flexible flow shop (FFS) scheduling problem with stochastic processing times. In this proposed approach, a neighbouring K-means clustering algorithm, involving the machine allocation algorithm and weighted CVIs, is developed to decompose an FFS without predefining the number of machine clusters. After decomposition of an FFS, either GA or SPT, determined by the optimal BPNs, is employed to generate the schedule for each machine cluster.

The effectiveness of DBA has been validated with experiment results. For most problems in the test-bed, DBA is found to be superior to either GA or SPT alone. The better performance of DBA is due to the decomposition strategy – to schedule with GA in a low stochastic environment and with SPT in a high stochastic environment. Using this strategy, DBA inherently yields good solutions when addressing FFS scheduling problems in any stochastic environment.

Although the development of DBA has been motivated by addressing the FFS scheduling problem with stochastic processing times, the generality of the decomposition strategy makes it possible to apply the DBA to deal with other types of uncertainty, such as machine breakdown. In order to extend the DBA, new stochastic vectors for machine clustering, and hence the difference of stochastic nature between machines, may have to be investigated.

References

Abumaizar, R. J., Svestka, J. A., 1997. Rescheduling job shops under random disruptions. International Journal of Production Research 35 (7), 2065-2082.

Akturk, M. S., Gorgulu, E., 1999. Match-up scheduling under a machine breakdown. European Journal of Operational Research 112 (1), 81-97.

Andres, C., Gomez, P., Garcia-Sabater, J. P., 2006. Comparing dispatching rules in dynamic hybrid flow shops. In: Proceedings of 2006 IEEE International Conference on Emerging Technologies and Factory Automation, Prague, Czech Republic, pp. 233-239.

Arthanari, T. S., Ramamurthy, K. G., 1971. An extension of two machines sequencing problem. Opsearch 8, 10-22.

Aytug, H., Lawley, M. A., McKay, K., Mohan, S., Uzsoy, R., 2005. Executing production schedules in the face of uncertainties: A review and some future directions. European Journal of Operational Research 161 (1), 86-110.

Chang, I. C., Hwang, H. G.., Liaw, H. C., Hung, M. C., Chen, S. L. and Yen, D. C., 2008. A neural network evaluation model for ERP performance from SCM perspective to enhance enterprise competitive advantage. Expert Systems with Applications 35(4), 1809-1816.

Church, L. K., Uzsoy, R., 1992. Analysis of periodic and event-driven rescheduling policies in dynamic shops. International Journal of Computer Integrated Manufacturing 5 (3), 153-163.

Davies, D.L., Bouldin, D.W., 1997. A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (4), 224–227.

Dunn, J. C., 1973. Fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. Journal of Cybernetics 3 (3), 32-57.

Garey, M. R., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York.

Gholami, M., Zandieh, M., Alem-Tabriz, A., 2009. Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. The International Journal of Advanced Manufacturing Technology 42 (1), 189-201.

Gupta, J. N. D., 1988. Two-stage, hybrid flowshop scheduling problem. Journal of the Operational Research Society 39 (4), 359-364.

Hunsucker, J. L., Shah, J. R., 1994. Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment. European Journal of Operational Research 72 (1), 102-114.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., Werner, F., 2008. Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. The International Journal of Advanced Manufacturing Technology 37 (3), 354-370.

Kim, D. J., Park, Y. W., Park, D. J., 2001. A novel validity index for determination of the optimal number of clusters. IEICE Transactions on Information and Systems E84-D(2), 281-285.

Kim, M., Ramakrishna, R. S., 2005. New indices for cluster validity assessment. Pattern Recognition Letters 26 (15), 2353-2363.

Kouvelis, P., Daniels, R. L., Vairaktarakis, G., 2000. Robust scheduling of a two-machine flow shop with uncertain processing times. IIE Transactions 32 (5), 421-432.

Lawrence, S. R., Sewell, E. C., 1997. Heuristic, optimal, static, and dynamic schedules when processing times are uncertain. Journal of Operations Management 15 (1), 71-82.

Liu, L., Gu, H. Y., Xi, Y. G., 2007. Robust and stable scheduling of a single machine with random machine breakdowns. The International Journal of Advanced Manufacturing Technology 31 (7), 645-654.

Lin, D. Y., Hwang, S. L., 1999. Use of neural networks to achieve dynamic task allocation: a flexible manufacturing system example. International Journal of Industrial Ergonomics 24 (3), 281-298.

Matsuura, H., Tsubone, H., Kanezashi, M., 1993. Sequencing, dispatching, and switching in a dynamic manufacturing environment. International Journal of Production Research 31 (7), 1671-1688.

MacQueen, J. B. 1967. Some methods for classification and analysis of multivariate observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, University of California Press, 281–297.

O’Donovan, R., Uzsoy, R., McKay, K. N., 1999. Predictable scheduling of a single machine with breakdowns and sensitive jobs. International Journal of Production Research 37 (18), 4217-4233.

Pinedo, M., 1995. Scheduling Theory, Algorithms, and Systems. Prentice Hall, New Jersey, USA.

Priddy, K. L., Keller, P. E., 2005. Artificial Neural Networks: An introduction. Bellingham, Washington: SPIE Press.

Rajendran, C., Holthaus, O., 1999. A comparative study of dispatching rules in dynamic flowshops and jobshops. European Journal of Operational Research 116 (1), 156-170.

Ripley, B.D., 1996. Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge.

Sabuncuoglu, I., Bayiz, M., 2000. Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research 126 (3), 567-586.

Sabuncuoglu, I., Kizilisik, O. B., 2003. Reactive scheduling in a dynamic and stochastic FMS environment. International Journal of Production Research 41 (17), 4211-4231.

Shen, J., Chang, S. I., Lee, E. S., Deng, Y., Brown, S. J., 2005. Determination of cluster number in clustering microarray data. Applied Mathematics and Computation 169 (2), 1172-1185.

Singh, A., Mehta, N., Jain, P., 2007. Multicriteria dynamic scheduling by swapping of dispatching rules. The International Journal of Advanced Manufacturing Technology 34 (9), 988-1007.

Tang, L., Liu, W., Liu, J., 2005. A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment. Journal of Intelligent Manufacturing 16 (3), 361-370.

Vidal, T., 2004. The many ways of facing temporal uncertainty in planning and scheduling. In: Proceedings of 14th IEEE International Symposium on Temporal Representation and Reasoning, Tatihou, France, pp 9-10.

Vieira, G. E., Herrmann, J. W., Lin, E., 2000. Predicting the performance of rescheduling strategies for parallel machine systems. Journal of Manufacturing Systems 19 (4), 256-266.

Vieira, G. E., Herrmann, J. W., Lin, E., 2003. Rescheduling manufacturing systems: A framework of strategies, policies, and methods. Journal of Scheduling 6 (1), 39-62.

Wang, H, 2005. Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. Expert Systems 22 (2), 78-85.

Wang, L., Zhang, L., Zheng, D. Z, 2005a. A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing time. The International Journal of Advanced Manufacturing Technology 25(11), 1157-1163.

Wang, L., Zhang, L., Zheng, D. Z, 2005b. Genetic ordinal optimisation for stochastic flow shop scheduling. The International Journal of Advanced Manufacturing Technology 27(1), 166-173.

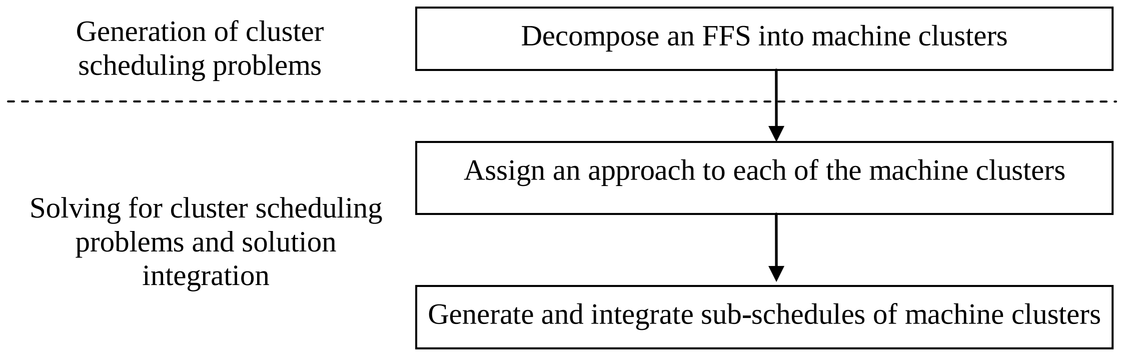

Generation of cluster scheduling problems
Decompose an FFS into machine clusters
Solving for cluster scheduling problems and solution integration
Assign an approach to each of the machine clusters
Generate and integrate sub-schedules of machine clusters

Figure 1: The framework of the proposed decomposition-based approach (DBA)

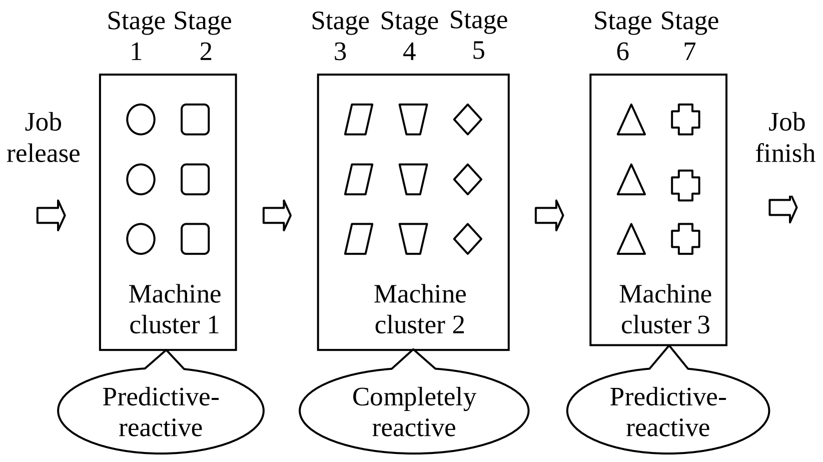


Figure 2: Machine clusters in a flexible flow shop (FFS)

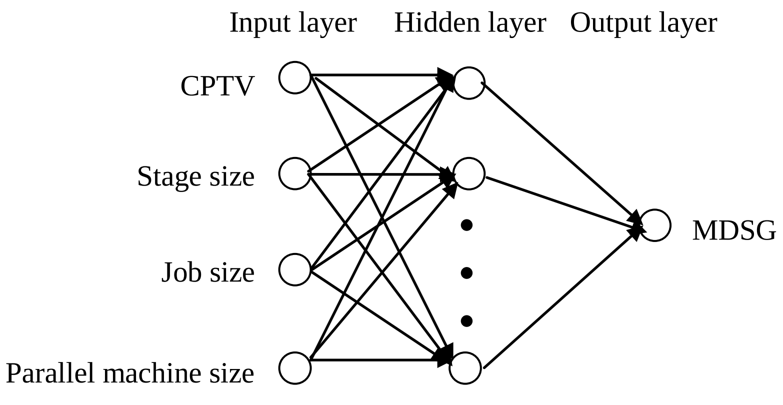
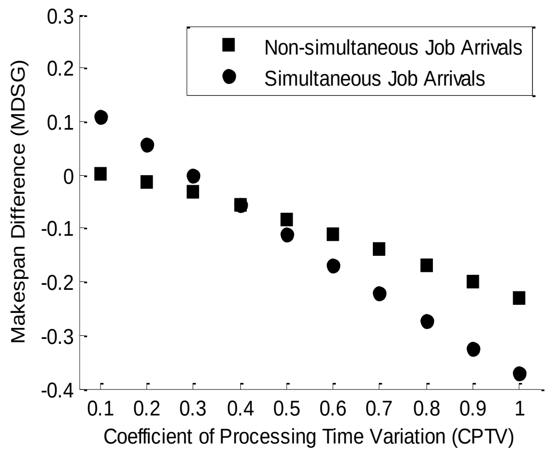
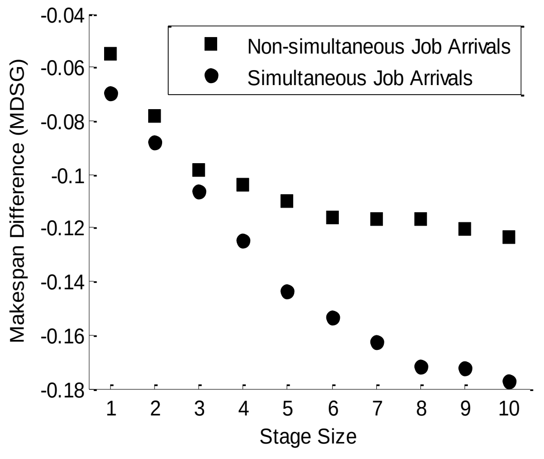


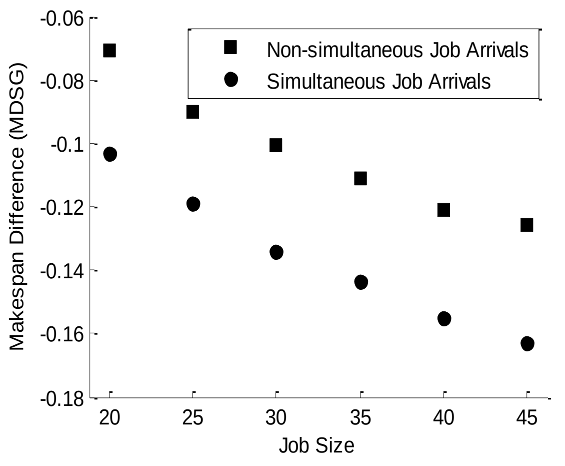
Figure 3: The architecture of a back propagation network (BPN)



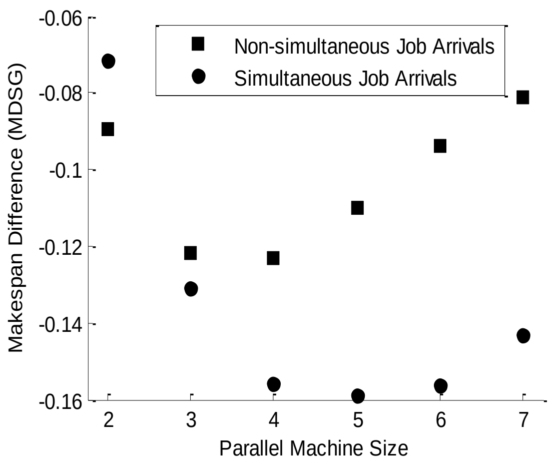
(a) MDSG against CPTV



(b) MDSG against stage size

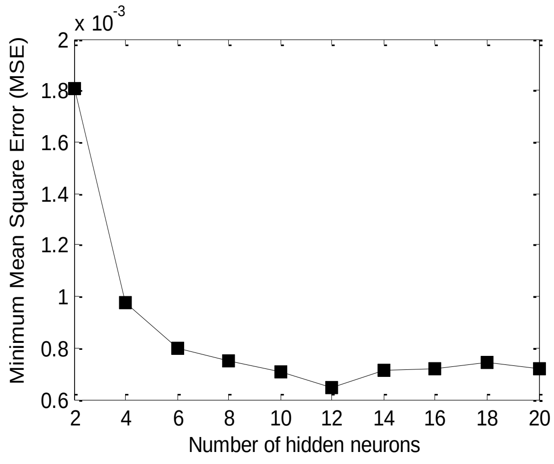


(c) MDSG against job size

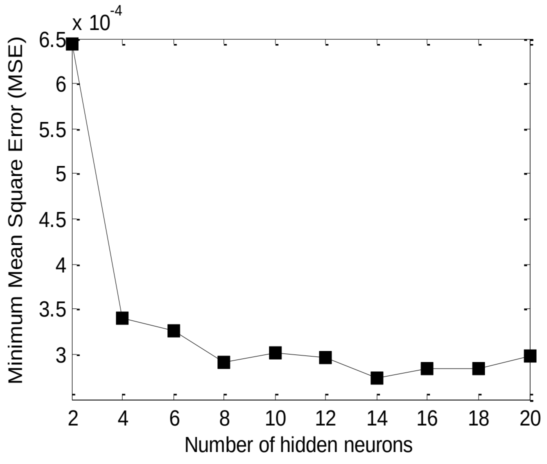


(d) MDSG against parallel machine size

Figure 4: Scatter plots of makespan difference (MDSG)



(a) Simultaneous job arrivals



(b) Non-simultaneous job arrivals

Figure 5: Relationship of minimum mean square errors (MSEs) with number of hidden neurons

Table 1: The proposed machine allocation algorithm

```
For k=1 to N-1 (N = number of machine cluster centres)
    For i= $S_k$ to $S_{k+1}$
        Allocate the machines between $S_k$ and ith stage to kth machine cluster centre, and compute
        the sum of DSNs from stages of parallel machines to their allocated machine cluster
        centres, using the formula: $\sum_{j=S_k}^{i} D\left(U_j, U_{S_k}\right)$;
        Allocate the machines between (i-1)th stage and $S_{k+1}$ to (k+1)th machine cluster centre, and
        compute the sum of DSNs from stages of parallel machines to their allocated machine
        cluster centres, using the formula: $\sum_{j=i-1}^{S_{k+1}} D\left(U_j, U_{S_{k+1}}\right)$;
        Compute the total DSN of the allocation using the formula:
        $\sum_{j=S_k}^{i} D\left(U_j, U_{S_k}\right) + \sum_{j=i-1}^{S_{k+1}} D\left(U_j, U_{S_{k+1}}\right)$;
        Compare the total DSN with the previous one, and keep the minimal total DSN and its
        corresponding result of machine allocation;
    End
    Allocate the machines according to the recorded result;
End

If $S_1$ is not the first stage
    Allocate the machines before $S_1$ to the first machine cluster centre;

If $S_N$ is not the last stage
    Allocate the machines after $S_N$ to the last machine cluster centre.
```

Table 2: Cluster validity indices (CVI)

CVI	Original Algorithm	Modified Algorithm
Dunn	$D_{nc} = \min_{i=1,\dots,nc}\left\{ \min_{j=i+1,\dots,nc}\left(\frac{InterDis(C_i,C_j)}{\max_{k=1,\dots,nc} IntraDis(C_k)} \right)\right\}$ where $InterDis(C_i,C_j)=\min_{x\in C_i,y\in C_j} d(x,y)$, $IntraDis(C_k)=diam(C_k)=\min_{x,y\in C_k} d(x,y)$.	$D_{nc} = \min_{i=1,\dots,nc}\left\{ \min_{j=i+1,\dots,nc}\left(\frac{WeightedInterDis_{i,j}(C_i,C_j)}{\max_{k=1,\dots,nc} IntraDis(C_k)} \right)\right\}$ where $WeightedInterDis(C_i,C_j)=W_{i,j}\times\min_{x\in C_i,y\in C_j} d(x,y)$, $IntraDis(C_k)=diam(C_k)=\min_{x,y\in C_k} d(x,y)$.
DB	$DB=\frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left(\frac{IntraDis(C_i)+IntraDis(C_j)}{InterDis(C_i,C_j)}\right)$, where $InterDis(C_i,C_j)=d(c_i,c_j)$, $IntraDis(C_k)=\frac{1}{n_k}\sum_{x\in C_k} d(x,c_k)$.	$DB=\frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left(\frac{IntraDis(C_i)+IntraDis(C_j)}{WeightedInterDis(C_i,C_j)}\right)$, where $WeightedInterDis(C_i,C_j)=W_{i,j}\times d(c_i,c_j)$, $IntraDis(C_k)=\frac{1}{n_k}\sum_{x\in C_k} d(x,c_k)$.
Vsv	$Vsv=v_{uN}(nc)+v_{oN}(nc)$, where $v_u(nc)=\frac{1}{nc}\sum_{i=1}^{nc}\left(\frac{1}{n_i}\sum_{x\in C_i} d(x,c_i)\right), v_o(nc)=\frac{nc}{d_{\min}}$, $d_{\min}=\min_{i\neq j} InterDis(C_i,C_j)$, $InterDis(C_i,C_j)=d(c_i,c_j)$. $v_{uN}(nc)$ and $v_{oN}(nc)$ are min-max normalized versions of $v_u(nc)$ and $v_o(nc)$	$Vsv=v_{uN}(nc)+v_{oN}(nc)$, where $v_u(nc)=\frac{1}{nc}\sum_{i=1}^{nc}\left(\frac{1}{n_i}\sum_{x\in C_i} d(x,c_i)\right), v_o(nc)=\frac{nc}{d_{\min}}$, $d_{\min}=\min_{i\neq j} InterDis(C_i,C_j)$, $InterDis(C_i,C_j)=W_{i,j}\times d(c_i,c_j)$. $v_{uN}(nc)$ and $v_{oN}(nc)$ are min-max normalized versions of $v_u(nc)$ and $v_o(nc)$
DVI	$DVIIndex=\min_{i=1,\dots,K}\left\{\begin{array}{l} IntraRatio(i) \\ +\gamma InterRatio(i)\end{array}\right\}$,where $IntraRatio(i)=\frac{Intra(i)}{MaxIntra}, InterRatio(i)=\frac{Inter(i)}{MaxInter}$, $Intra(i)=\frac{1}{N}\sum_{j=1}^{i}\sum_{x\in C_j}\lVert x-c_j\rVert^2$, $MaxIntra=\max_{i=1,\dots,K}(Intra(i))$, $Inter(i)=\frac{Max_{k,j}\left(InterDis(C_k,C_j)^2\right)}{Min_{k\neq j}\left(InterDis(C_k,C_j)^2\right)}$, $\times\sum_{k=1}^{i}\left(\frac{1}{\sum_{j=1}^{i}\left(InterDis(C_k,C_j)\right)}\right)$ $InterDis(C_i,C_j)=\lVert c_i-c_j\rVert$, $MaxInter=\max_{i=1,\dots,K}(Inter(i))$.	$DVIIndex=\min_{i=1,\dots,K}\left\{\begin{array}{l} IntraRatio(i) \\ +\gamma InterRatio(i)\end{array}\right\}$,where $IntraRatio(i)=\frac{Intra(i)}{MaxIntra}, InterRatio(i)=\frac{Inter(i)}{MaxInter}$, $Intra(i)=\frac{1}{N}\sum_{j=1}^{i}\sum_{x\in C_j}\lVert x-c_j\rVert^2$, $MaxIntra=\max_{i=1,\dots,K}(Intra(i))$, $Inter(i)=\frac{Max_{k,j}\left(WeightedInterDis(C_k,C_j)^2\right)}{Min_{k\neq j}\left(WeightedInterDis(C_k,C_j)^2\right)}$, $\times\sum_{k=1}^{i}\left(\frac{1}{\sum_{j=1}^{i}\left(WeightedInterDis(C_k,C_j)\right)}\right)$ $WeightedInterDis(C_i,C_j)=W_{i,j}\times\lVert c_i-c_j\rVert$, $MaxInter=\max_{i=1,\dots,K}(Inter(i))$.

Table 3: The proposed neighbouring K-means clustering algorithm

```
For k=2 to $K_{max}$ ($K_{max}$ = number of stages/2)
    Choose k as the number of machine clusters;
    Randomly pick up k machine cluster centres from all the stages in an FFS,
    and each machine cluster centre is the parallel machines at a stage;
    Allocate the parallel machines of each stage to the machine cluster centres
    by the machine allocation algorithm;
    Compute the new machine cluster centres;
    Until the allocation result remains constant or a predefined number of
    iterations have been reached (As more iterations increase computation
    burden and may not greatly improve the performance, we fix it as 100
    empirically.)
        Re-allocate the parallel machines of each stage to the machine cluster
        centres by the machine allocation algorithm;
        Re-compute the new machine cluster centres;
    End                                                          } Decomposition
    Compute the weighted CVI of decomposing an FFS into k machine
    clusters;                                                    } Evaluation
End
Return machine clusters at k where the weighted CVI is optimal over all k.
```

Table 4: Back propagation network (BPN) inputs and their levels

Factors	Levels
CPTV	10 levels (0.1, 0.2, …, 1)
Stage size	10 levels (1, 2, …, 10)
Job size	20, 25, 30, 35, 40, 45
Parallel machine size	2, 3, 4, 5, 6, 7

Table 5: ANOVA for MDSG generating from the scenario of simultaneous job arrivals

Factor	SS	DF	MS	F-ratio	P-value
CPTV	86.404	9	9.600450	3410.830	<0.001
Stage size	4.923	9	0.546950	194.320	<0.001
Job size	1.497	5	0.299410	106.370	<0.001
Parallel machine size	3.396	5	0.679260	241.330	<0.001
Residual	10.051	3571	0.002815		
Total	106.274	3599			

Table 6: ANOVA for MDSG generating from the scenario of non-simultaneous job arrivals

Factor	SS	DF	MS	F-ratio	P-value
CPTV	21.220	9	2.357790	1681.110	<0.001
Stage size	1.416	9	0.157310	154.300	<0.001
Job size	1.270	5	0.253930	249.070	<0.001
Parallel machine size	0.932	5	0.186480	182.910	<0.001
Residual	3.641	3571	0.001020		
Total	28.479	3599			

Table 7: Comparison of ratios of average makespans of various cluster validity indices (CVI)

Ratio = Average makespan of CVI / Average makespan of Dunn							
Traditional K-means Clustering Algorithm				The proposed Neighbouring K-means Clustering Algorithm			
Dunn	DB	Vsv	DVI	W-Dunn	W-DB	W-Vsv	W-DVI
1.000	0.999	1.001	1.002	0.998	0.985	0.995	0.998

Table 8: Comparison of ratios of the average makespan of various scheduling algorithms

Problem size No. of jobs x no. of stages	No. of parallel machines at each stage	Ratio = Average makespan / Average makespan of GA				
		Deterministic processing times		Stochastic processing times		
		SPT_D	GA_D	SPT_S	GA_S	DBA_S
20×6	2	1.170	1.000	1.318	1.384	1.257
20×6	3	1.202	1.000	1.345	1.395	1.276
20×6	4	1.220	1.000	1.396	1.557	1.350
20×10	2	1.120	1.000	1.321	1.450	1.305
20×10	3	1.165	1.000	1.296	1.409	1.277
20×10	4	1.119	1.000	1.302	1.504	1.292
20×15	2	1.144	1.000	1.290	1.413	1.255
20×15	3	1.114	1.000	1.265	1.459	1.246
20×15	4	1.083	1.000	1.170	1.295	1.158
30×6	2	1.099	1.000	1.246	1.358	1.212
30×6	3	1.165	1.000	1.309	1.415	1.246
30×6	4	1.163	1.000	1.341	1.535	1.271
30×10	2	1.135	1.000	1.320	1.459	1.298
30×10	3	1.153	1.000	1.287	1.425	1.249
30×10	4	1.121	1.000	1.279	1.504	1.280
30×15	2	1.157	1.000	1.291	1.404	1.261
30×15	3	1.159	1.000	1.284	1.475	1.255
30×15	4	1.125	1.000	1.295	1.499	1.279
40×6	2	1.134	1.000	1.242	1.265	1.199
40×6	3	1.129	1.000	1.257	1.354	1.188
40×6	4	1.139	1.000	1.296	1.514	1.291
40×10	2	1.165	1.000	1.314	1.375	1.249
40×10	3	1.158	1.000	1.326	1.508	1.256
40×10	4	1.086	1.000	1.266	1.528	1.252
40×15	2	1.147	1.000	1.297	1.402	1.255
40×15	3	1.116	1.000	1.265	1.472	1.237
40×15	4	1.118	1.000	1.276	1.538	1.272
Average		1.141	1.000	1.292	1.441	1.258