Decomposition-Based Scheduling for Makespan Minimisation of Flexible Flow Shop with Stochastic Processing Times

K. Wang and S.H. Choi

***Abstract*—Since real manufacturing is dynamic and tends to suffer a wide range of uncertainties, research on production scheduling under uncertainty has received much more attention recently. Although various approaches have been developed for scheduling under uncertainty, this problem is still difficult to tackle by any single approach, because of its inherent difficulties. This paper considers makespan minimisation of a flexible flow shop (FFS) scheduling problem with stochastic processing times. It proposes a novel decomposition-based approach (DBA) to decompose an FFS into several machine clusters which can be solved more easily by different approaches. 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 a weighted cluster validity index. A back propagation network (BPN) is then adopted to assign either the shortest processing time (SPT) or the genetic algorithm (GA) to generate a sub-schedule for each machine cluster. If two neighbouring machine clusters are allocated with the same approach, they are subsequently merged. After machine grouping and approach assignment, an overall schedule is generated by integrating the sub-schedules of the clusters. Computation results reveal that the proposed approach is superior to SPT and GA alone for FFS scheduling under stochastic processing times.**

***Keywords*—back propagation network, decomposition, flexible flow shop, neighbouring K-means clustering algorithm, stochastic processing times.**

I. Introduction

Ever since the flexible flow shop (FFS) scheduling problem was identified in 1970's [1], it has attracted considerable attention during the past decades [2]. An FFS 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. Research efforts on FFS scheduling problems generally consider a static environment with no unexpected events that would influence the job processing when the schedule is executed.

Real manufacturing, however, is dynamic and tends to suffer a wide range of uncertainties, such as stochastic processing times, machine breakdown, rush orders, job cancellations, and change of due date. This paper is primarily concerned with the scheduling problem of FFS with stochastic processing times. The FFS scheduling problem [3] has been proven NP-hard in nature which is difficult to solve [4, 5]. Consideration of stochastic processing times aggravates its complexity.

As a research issue, production scheduling under uncertainty has indeed drawn considerable attention in recent years. The completely reactive approach, the robust approach, and the predictive-reactive approach are three fundamental ways [6, 7] to tackle this issue.

The completely reactive approach changes decisions during execution when necessary. The dispatching rule is a typical reactive one, in which jobs are selected by sorting them according to predefined criteria. It is easy to understand, and can find a reasonably good solution in a relatively short time. However, it uses only local information to generate a schedule, which may not be globally optimal in nature.

Hunsucker and Shah [8] compared the performance of dispatching rules in a constrained multiprocessor flow shop, and concluded that the Shortest Processing Time (SPT) algorithm was superior for the makespan criterion. Similarly, Rajendran and Holthaus [9] 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 rules dominated in all criteria.

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

Tang et al. [10] examined the dynamic FFS scheduling problem with random job arrivals. He applied the neural network to dynamically select a dispatching rule to generate schedules. Experiment results indicated that the neural network approach consistently performed better than a single traditional dispatching rule. Singh et al. [11] 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 scheduling approach takes into account possible uncertainties to construct solutions. Uncertainties, known as a priori, can be modelled by some random variables [12]. If such uncertainties are difficult to quantify, a range of scenarios will be considered and a solution is developed to optimise the performance under different scenarios [13]. In

K. Wang is with the Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China (phone: 852-6874-0287; fax: 852-2858-6535; e-mail: wkxy8009@hkusua.hku.hk).

S. H. Choi is with the Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China (e-mail: shchoi@hku.hk).

this case, the approach is viewed as a form of under-capacity scheduling to maintain robustness under different scenarios.

Wang et al. [14] presented a class of hypothesis-test-based genetic algorithms to address flow shop scheduling problems with stochastic processing times. The solutions were generated by GA and evaluated by multiple independent simulations. 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.

Ahmadizar et al. [15] addressed a stochastic group shop scheduling problem with random release dates and processing times. They developed a simulation optimisation approach, which was a hybrid of an ant colony optimisation algorithm, a heuristic algorithm to generate good solutions, and a discrete event simulation model to evaluate the solution performance.

The predictive-reactive approach is indeed 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. This approach is by far the most studied. 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 [16, 17]. 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.

To address the second issue, the most common rescheduling methods include the right-shift schedule repair, the partial schedule repair, and the completed scheduling [18]. 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. The completed scheduling regenerates a completely new schedule for all the unprocessed operations. Although the completed scheduling may construct a better solution in theory, it is rarely applied in practice due to high computation burden and increasing scheduling instability [12]. 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.

Since each of these three approaches has its own strength and weakness, some research work has focused on comparing their effectiveness. Lawrence and Sewell [19] studied the static and dynamic applications of heuristic approach to job shop scheduling problems when processing times were uncertain. Experiment results indicated that the predictive methods based on overall information were highly likely to perform better than completely reactive approaches in an environment under little uncertainty. However, the predictive methods might lead to poor result when the uncertainty in a system exceeded a certain level. Sabuncuoglu and Bayiz [20] 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 the stochastic environment. This conclusion was consistent with that of Lawrence and Sewell's [19].

In order to handle a complex environment, it is beneficial and imperative to take advantage of mixing these three approaches to deal with uncertainty. Matsuura et al. [21] developed a predictive approach on a periodic basis, called switching. The system switched to using a dispatching rule for the remaining operations when the deviation between the realized and predictive schedule exceeded a certain level. They concluded that the proposed approach dominated the dispatching rules when the frequency of disruption was low, but it yielded worse results than the dispatching rules when the disruption reached some level. A search of available literature indicates not much research works have been attempted to address the combination of different approaches.

This paper studies the problem of scheduling an FFS under the uncertainty of stochastic processing times, with the objective to minimise the makespan. Enlightened by the work of Lawrence's [19], a decomposition-based approach (DBA) is proposed. In this approach, a neighbouring K-means clustering algorithm first groups the machines of an FFS into several machine clusters based on their stochastic nature when processing jobs. Then the completely reactive approach or the predictive-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 into an overall solution.

This study contributes to the development of an integrated approach that combines and takes advantage of the completely reactive approach with the predictive-reactive approach to deal with the uncertainty. On the contrary, the techniques reported in available literature on scheduling under uncertainty were mostly based on a single approach, yielding some initial yet limited performance. The proposed 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. Section II is devoted to problem description. Section III describes the framework of DBA, while it is explained in detail in Section IV. To evaluate the effectiveness of DBA, simulation is conducted and computation results are analysed in Section V. Finally, conclusions are summarised and some directions of future work are discussed in Section VI.

II. Problem Description

In the FFS discussed above, machines sharing a similar characteristic are arranged into stages in series. Jobs have to pass all the stages in the same order. In each stage, there are a number of functionally identical machines in parallel, and a job is to be processed on one of these machines. The processing time may be highly uncertain due to quality problems, equipment downtime, tool wear, and operator availability [19].

The stochastic processing time can be described by 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 processing times are deterministic, and increases as the uncertainty increases.

In order to simplify the typical FFS scheduling problem in consideration of stochastic processing times, the following assumptions are made:

- Preemption is not allowed for job processing;
- All jobs are released at the same time for the first stage;
- All machines are available when jobs are released to the FFS. Each machine can process at most one operation at a time;
- There is no travel time between machines;
- There is no setup time for job processing;
- Infinite buffers exist for machines;
- For the same job, the expected processing time at any parallel machine at a stage is identical;
- The actual processing time of a job on a machine is uncertain, and it can be longer or shorter than the expected one;
- As parallel machines at a stage are functionally identical, they lead to the same CPTV when processing any jobs, but the CPTV may be different for other stages;
- Except for stochastic processing times, there are no other types of uncertainties to disturb job processing.

The scheduling objective 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 minimised 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:

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

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

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

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

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

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

$$ST_{(k+1)i_1j} - ST_{ki_2j} \geq P_{ki_2j} \tag{8}$$

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

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
$B_{kij_1j_2}$:	a 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}:	a Boolean variable, 1 if Job j is the first job on machine i at stage k, and 0 otherwise
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, (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, (4) and (5) determine the completion time of the first job and that of each subsequent job on the machines, respectively.

While (6) ensures non-negative start time of job processing, (7) stipulates that each of the parallel machines at a stage takes equal time to process the same job.

Lastly, (8) requires the processing sequence of each stage to satisfy the processing time, and (9) guarantees that each machine can process only one job at a time.

III. The Framework of The Proposed Decomposition-based Approach (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 a high stochastic environment [19, 20]. Based on this observation, a decomposition-based approach (DBA) is proposed to provide better performance for FFS scheduling problems in any stochastic environment. The DBA framework consists of three modules, as shown in Fig. 1.

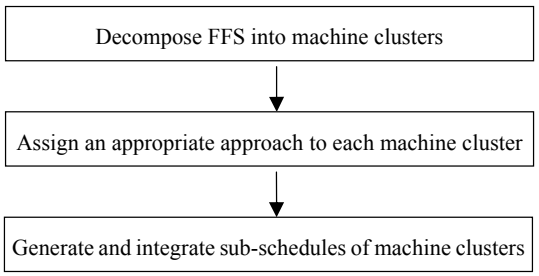

Figure 1. The framework of the proposed decomposition-based approach (DBA)

A. FFS Decomposition

An FFS is firstly decomposed by a clustering algorithm into machine clusters, each of which contains a number of machines sharing 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. The high stochastic nature of a machine usually leads to a large difference between the actual and the planned schedule. As the actual processing times of jobs on a machine

may be non-deterministic, the processing time uncertainty during job processing 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 trait. Quite a few algorithms, such as K-means, fuzzy C-means, and self-organization maps etc., have been proposed to perform the classification. Since the K-means clustering algorithm is simple and widely used, a neighbouring K-means clustering algorithm is proposed and adopted to decompose an FFS in this paper.

After decomposition of an FFS, machines in the same machine cluster share a similar stochastic nature of job processing and can be scheduled by the same approach. Machine clusters with low stochastic natures are solved by the predictive-reactive approach, while those with high stochastic natures are scheduled by the completely reactive approach. Due to their better performance, the Genetic Algorithm (GA) and the Shortest Processing Time (SPT) algorithm are identified as the predictive-reactive approach and the completely reactive approach, respectively.

B. Approach Assignment

In order to assign an appropriate approach to a machine cluster, it is critical to establish an effective model to estimate the makespan difference (MDSG) when generating the schedule by both 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 [22]. 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.

Fig. 2 shows the decomposition result of an FFS with 7 stages and 3 parallel machines at each stage. Geometric figures with the same shape represent the parallel machines. One of the two approaches, GA or SPT, is assigned to each machine cluster.

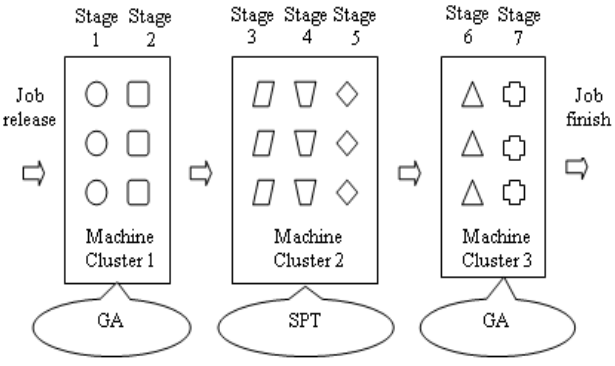

Figure 2. Machine clusters in an FFS

C. Sub-schedule Generation and Integration

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

The major feature of DBA is its decomposition strategy – combining and taking advantage of GA and SPT to generate a better result when scheduling in any stochastic environment.

IV. Detailed Algorithm

A. Neighbouring K-means Clustering Algorithm

For the purpose of decomposing an FFS, the machines of an FFS are grouped into a few machine clusters in which machines share a similar stochastic nature. Since the CPTV represents processing time uncertainty, it is adopted to form the stochastic vector U_i to group machines into machine clusters of similar stochastic natures, giving

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

Where $CPTV_i$ is the CPTV of the parallel machines at stage i. U_i represents the stochastic nature of machines at stage i. A machine with a large CPTV indicates a high stochastic nature of job processing.

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 machine distance $D(U_i, U_j)$, which represents not the physical distance but the difference of stochastic nature between the parallel machines at stages i and j. The machine distance is calculated as follows.

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

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.

Considering $D(U_i, U_j)$, the FFS can be decomposed into machine clusters by K-means clustering algorithm. The major problem to apply K-means clustering algorithm is the choice of machine cluster number. Neither a small nor a large machine cluster number can offer a satisfactory classification of the data objects. Recently, cluster validity indices (CVIs), indicating how well the clustering algorithm classifies the given data set, have attracted much attention as an approach to determining the optimal cluster number.

Most CVIs are defined by combining the intra-cluster distances and inter-cluster distances. The former one measures the distances of objects within a cluster to represent its compactness, while the latter one computes the distance between two different clusters and acts as an indicator of cluster separability. Therefore, a good clustering algorithm should have small intra-cluster distances and large inter-cluster distances. Dunn [23], DB [24], Vsv [25] and DVI [26] are some typical CVIs.

However, the FFS decomposition above is different from the traditional clustering problem. Since this study aims 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, a modified DB (MDB), giving the weight to the inter-cluster distance, is proposed as follows:

$$MDB = \frac{1}{n}\sum_{i=1}^{n} \max_{i \neq j} \left(\frac{S_i + S_j}{W_{ij} \times D_{ij}} \right) \tag{12}$$

$$w_{ij} = 1 \Big/ \left(F_i + F_j \right) \tag{13}$$

Where n is the number of machine clusters; S_i denotes the intra-cluster distance, which measures the average machine distance of all objects from the machine cluster i to their cluster centre; D_{ij} represents the inter-cluster distance, which measures the machine distance between machine cluster centre i and j; W_{ij} is the weight of D_{ij}; F_i is the first stage of i^{th} machine cluster. A small value of MDB indicates a good clustering.

In order to avoid specifying the machine cluster number, a neighbouring K-means clustering algorithm, incorporated with MDB, is established. Its procedure is shown in Fig. 3.

For k=2 to K_{max} (K_{max} = number of stages/2)
  For i=1 to I_{max} (I_{max} = 10)
    Apply the K-means clustering algorithm to decompose an FFS into k machine clusters;
    Compute the MDB for i^{th} iteration of decomposing an FFS into k machine clusters;
  End
End
Return machine clusters where the MDB is minimal over all i and all k.

Figure 3. The proposed neighbouring K-means clustering algorithm

B. Back Propagation Network for Approach Assignment

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

$$MDSG = \left(M_{SPT\_S} - M_{GA\_S} \right) \Big/ M_{GA} \tag{14}$$

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

In order to accurately estimate the MDSG for each machine cluster, the back propagation network (BPN) is adopted in this study. For a machine cluster, if the MDSG 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.

Under the assumptions we made on the FFS scheduling problem, jobs are released simultaneously in the first stage. However, in subsequent stages, they are allocated by the FIFO rule and may arrive non-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 allows the jobs arrive non-simultaneously.

Accordingly, two BPNs, each corresponds to a scenario, are generated. Their architectures are identical, as illustrated in Fig. 4. The details of BPN establishment for each scenario are as follows:

- Inputs: Four parameters, namely CPTV, stage size, job size, and parallel machine size. These parameters are found to affect the performance of MDSG significantly according to the experiment results in Section V. The first input is the mean of 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. Therefore, the BPN only consists of one hidden layer;
- Number of hidden neurons: 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 extremely well, but would perform poorly to new and unseen data. For these reasons, the number of hidden neurons is intentionally selected 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 mean square error (MSE), and the one that corresponds to the number of hidden neurons that give rise to the least minimal 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 (14);
- Number of epochs per replication: 10000;
- Number of replications: 100. The performance of a BPN is sensitive to the initial condition of network. 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 minimal MSE is kept for the purpose to further identify the optimal BPN.
- Training examples: A training example consists of a set of values for the input neurons and the output neurons, including CPTV, stage size, job size, parallel machine size, and the expected MDSG. The details of training example generation are described in Section V.

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

After training, validation, and testing, the optimal BPN can be identified and used to determine the MDSG for each scenario. 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.

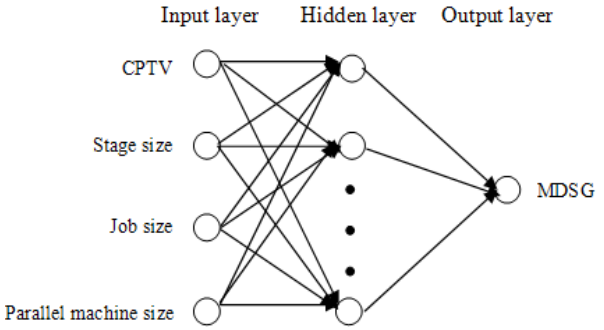

Figure 4. The architecture of BPNs

The neighbouring K-means clustering algorithm cannot avoid the possibility that two neighbouring machine clusters are to be suitably solved by the same approach. Therefore, it is reasonable to conduct a machine cluster merging process (MCMP) 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 MCMP, the complete process of approach assignment is summarised as follows:

(1) Generate training examples for both scenarios.

(2) Train, validate and test the BPNs;

(3) Estimate the MDSG for each machine cluster by optimal BPNs;

(4) Assign either SPT or GA to each machine cluster according to the positive or negative sign of its estimated MDSG, respectively;

(5) Conduct MCMP.

Although the BPN provides a powerful way to model any complex relationship between input and output data, it has its own disadvantage. In order to improve the prediction accuracy, a large number of training examples are required and a long training time therefore is unavoidable. Moreover, the scale of training examples and training time grows with the increasing number of BPN inputs and their levels. Although it is time-consuming to train the BPNs, it is a one-off and the computation cost of DBA based on the trained BPNs is comparable to that of GA or SPT.

C. Machine Cluster Scheduling

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

SPT, firstly handling the job with the shortest processing time, performs better with low computation cost when the machines in a machine cluster with 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, a typical local search algorithm, is used prior to the dispatching rules when scheduling a machine cluster with a low stochastic nature. The overall structure of our GA is briefly described as follows:

- Coding: The job sequence is used as the chromosome for the FFS scheduling problem. Integer coding scheme is adopted for chromosome representation in the study. For example, job sequence {2, 3, 5, 1, 4, 9, 8, 6, 7, 10} is a chromosome with ten jobs in an FFS;
- Initialization: The initial population is randomly generated;
- Fitness function: For the purpose to minimise 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;
- Selection strategy: Roulette wheel selection is applied to reproduce the next generation;
- Crossover operation: Order preserved crossover (OPX) is adopted. The OPX emphasizes the relative order of the genes from both parents. To create the first child, a gene segment from the first parent is firstly randomly selected and placed into the first child corresponding to the first parent position, and then the genes of the second parent, not included in the segment, are put into the first child according to the second parent order. To generate the second child, the gene in the segment are copied to the second child and preserves the relative positions corresponding to the second parent, and then the genes of the first parent, except for the ones in the segment, are filled in the second child according to the first parent order. For instance, {2, 3, 5, 1, 4, 9, 8, 6, 7, 10} and {1, 2, 4, 5, 6, 7, 8, 3, 9, 10} are randomly selected as parents, and the gene segment is {4, 9, 8, 6}. Such crossover operation produces the children as {1, 2, 4, 9, 8, 6, 5, 7, 3, 10} and {2, 3, 4, 5, 6, 1, 8, 7, 9, 10};
- Mutation operation: Shift move mutation (SM) is used. The SM changes the relative position of one job. This operation shifts a gene from the current position to a new one while leaving all other relative gene orders unchanged. For instance, one parent, {9, 8, 6, 7, 10, 2, 3, 5, 1, 4}, is randomly selected, and the couple of genes at position 3and 7 are selected to performing the operation. The offspring will be {9, 8, 7, 10, 2, 3, 6, 5, 1, 4};
- The crossover rate and mutation rate: The rates are analysed 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;
- Termination criterion: The algorithm continues until 200 generations have been examined. This value is chosen empirically.

To react to job processing delay caused by stochastic processing times, SPT and GA adopt different policies for machine cluster scheduling. For SPT, job allocation follows an initial sequence at the first stage and FIFO rule at all other stages. For GA, considering reducing computation effort and scheduling instability, the right-shift scheduling repair is used, which postpones the operations affected without changing the

job sequence in comparison with that of the schedule with deterministic processing times.

V. Computation Results And Analysis

A. Experiment Design

Two experiments are designed and conducted for the performance evaluation of the proposed DBA. The first experiment aims at establishing the BPNs for the MDSG estimation. The second experiment focuses on analysing the performance of DBA on makespan criterion. All the two 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 two experiments above, the expected processing times of operations are generated uniformly in the time unit interval {1, 20} and their average value P equals 10 time units, while the actual processing times are uncertain and follow the gamma distribution with the expected processing time E(P) and standard deviation $\sigma = E(P) \times CPTV_j$, where $CPTV_j$ is the coefficient of processing time variation at stage j.

In order to evaluate the proposed DBA in the second experiment, a test-bed, containing 27 (3*3*3 = 27) problems with different stages, jobs and parallel machines, is established and shown in Table II. 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 problem, ten instances with different expected processing times of operations are randomly generated and the simulation is iterated 50 times for each instance.

B. Experiment I: Generation of BPNs for MDSG Estimation

In order to train, validate, and test the BPNs, it is necessary to firstly generate training examples. Corresponding to the two scenarios of simultaneous and non-simultaneous job arrivals, two sets of training examples are generated, respectively.

The levels of BPN input used in the experiment, including CPTV, stage size, job size, and parallel machine size are shown in Table I.

TABLE I. 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

For each scenario, by exploring all possible combinations ($10 \times 10 \times 6 \times 6$ = 3,600) of BPN inputs, the experimental FFS scheduling problems to minimise makespans with stochastic processing times are firstly generated, in each of which all the parallel machines share the same CPTV. Subsequently, these problems are solved by GA and SPT, respectively. Lastly, the MDSG, which is the output of BPN, can be obtained by (14) for each problem. Thus, this procedure results in a total of 3,600 training examples for each scenario.

Based on the data of the training examples, scatter plots are generated to visualize the relationship of four factors, including CPTV, stage size, job size and parallel machine size, on the MDSG. As shown in Fig. 5, circles and squares represent the results derived by the scenarios of simultaneous job arrivals and non-simultaneous job arrivals, respectively. For a specific x-value in Fig. 5, the y-value is the mean of MDSGs of all the corresponding training examples. Accordingly the following conclusions are drawn from the scatter plots:

- The MDSG decreases with the increasing of CPTV, stage size and job size. Parallel machine size affects the MDSG as well. Therefore, it is reasonable to adopt these four factors as BPN inputs;
- The MDSG is different for the two scenarios of simultaneous and non-simultaneous job arrivals. Hence, two BPNs are needed to estimate the MDSG.

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, BPNs with different number of hidden neurons are established and their prediction accuracy is measured by MSE.

Fig 6 shows the relationship of the minimal MSE with various numbers of hidden neurons. It is apparent 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.

C. Experiment II: DBA Analysis

In order to evaluate the effectiveness of the proposed DBA, it is analysed and compared with SPT and GA in a stochastic environment, in which CPTV is uniformly distributed in the interval {0.1, 1}. The experiment results of these three algorithms with stochastic processing times (denoted by SPT_S, GA_S, and DBA_S respectively) are shown in Table II. The results of SPT and GA with deterministic processing times (denoted by SPT, and GA respectively) are also given.

All the results are the ratios of the average makespan of various scheduling algorithms to that of GA. From the experiment results, the following conclusion can be drawn:

- SPT_S performs better than GA_S. The reason for such poor performance of GA lies in using the right-shift schedule repair in response to job processing delay caused by stochastic processing times.
- DBA_S gives the best performance in most cases, decreasing the makespan by about 3% and 13% in comparison with SPT_S and GA_S, respectively. Although DBA_S does not always perform best, it is the only algorithm that can consistently give good performance. The good performance of DBA is due to its decomposition strategy, which combines the strengths of GA and SPT to deal with stochastic processing times.

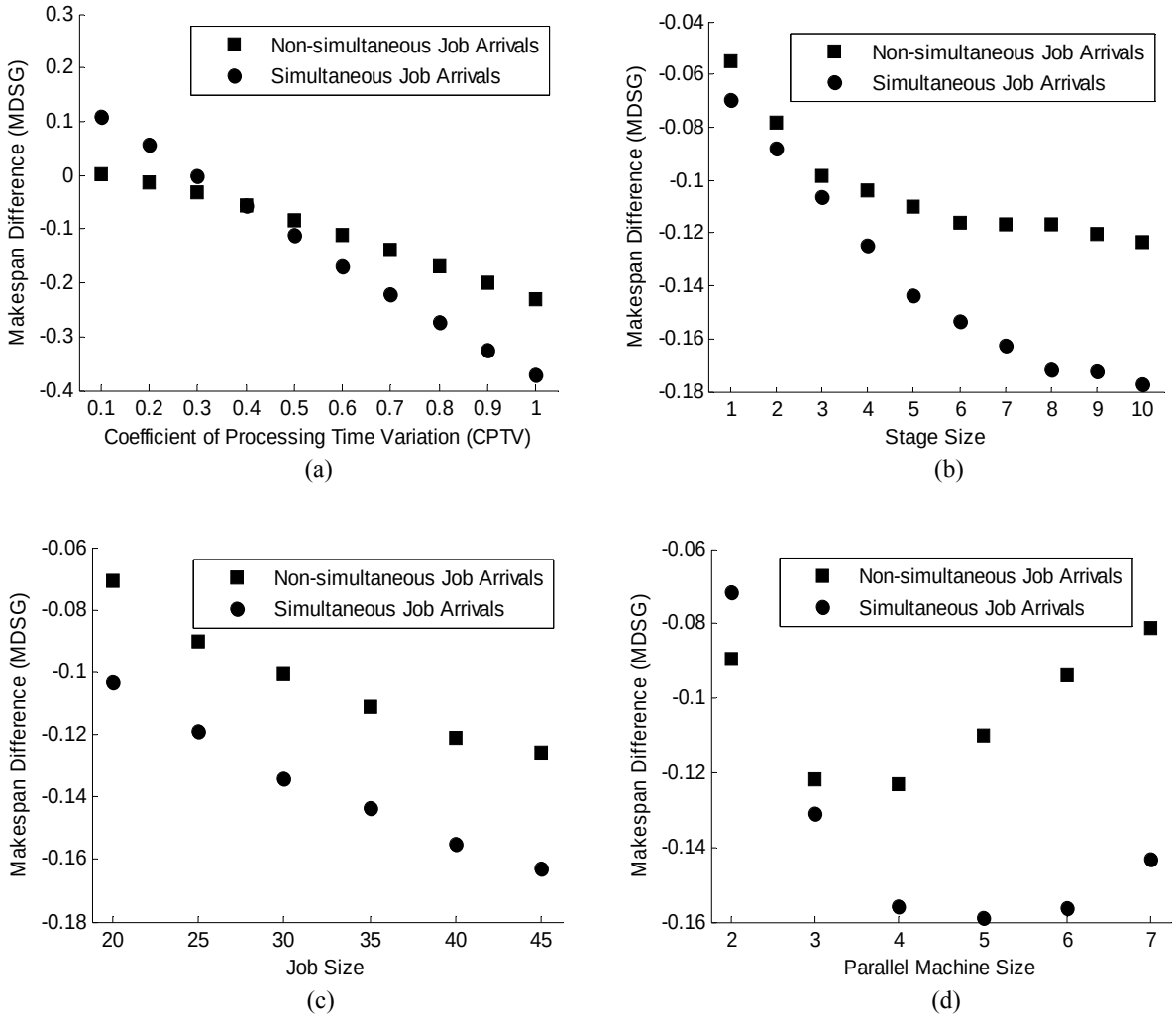

Figure 5. Scatter plots of the MDSG with (a) coefficient of processing time variation (CPTV), (b) stage size, (c) job size, (d) parallel machine size.

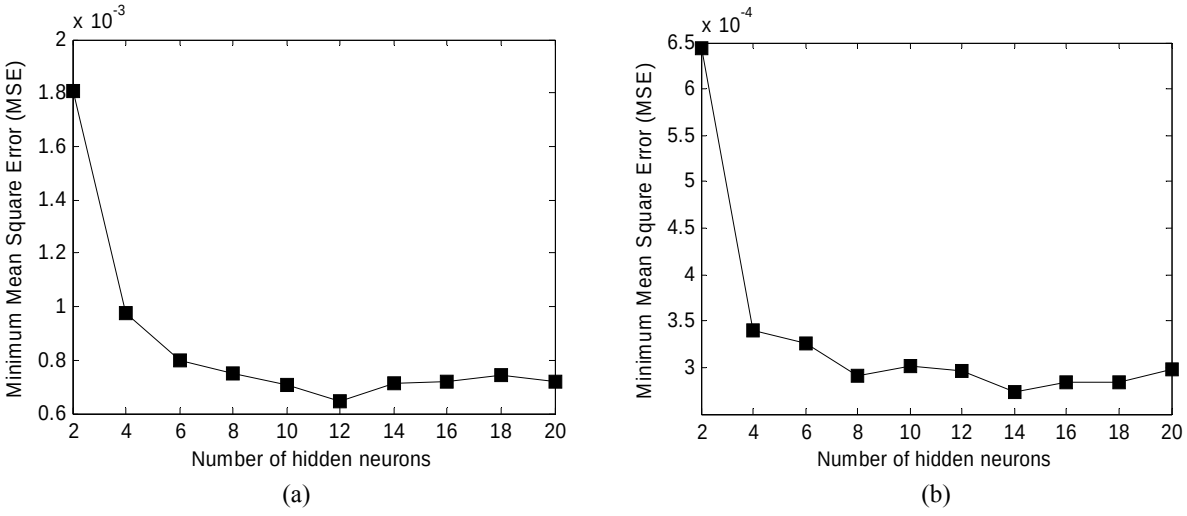

Figure 6. The minimal MSE with various numbers of hidden neurons for (a) simultaneous job arrivals, (b) non-simultaneous job arrivals.

TABLE II. COMPARISON OF PERFORMANCE OF VARIOUS SCHEDULING ALGORITHMS [a]

Problem size (No. of jobs x no. of stages)	No. of Parallel Machines in each stage	SPT	GA	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**

a. All the results are the ratios of the average makespan of various scheduling algorithms to that of GA.

VI. CONCLUSION

This paper proposed a decomposition-based approach (DBA) to minimise the makespan of an FFS scheduling problem with stochastic processing times. In this approach, machines are grouped into several machine clusters by a neighbouring K-means clustering algorithm without predefining the number of machine clusters, and each machine cluster is scheduled by either SPT or GA.

The effectiveness of DBA was validated with experiment results. For most problems in the test-bed, DBA is superior to SPT and GA. The better performance of DBA results from the decomposition strategy – to schedule with GA in a low stochastic environment and with SPT in a high stochastic environment. This strategy ensures DBA's good performance when addressing FFS scheduling problems in any stochastic environment.

The proposed DBA provides a promising way to address FFS scheduling under stochastic processing times. Further research can be devoted to evaluating the performance of DBA by optimising the FFS scheduling problem with respect to tardiness-related criteria, such as minimising the mean tardiness of jobs. In addition, as a job shop is essentially more complex than an FFS, another possible research area can focus on extending the proposed DBA to solve job shop scheduling problems with stochastic processing times.

REFERENCES

[1] T. S. Arthanari and K. S. Ramamurthy, "An extension of two machines sequencing problem," Opsearch, 8, pp. 10-22, 1971.

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

[3] R. Linn and W. Zhang, "Hybrid flow shop scheduling: a survey," Computers and Industrial Engineering, vol. 37, pp. 57-61, 1999.

[4] M. R. Garey, Computers and intractability: a guide to the theory of NP-completeness. New York: W. H. Freeman, 1979.

[5] J. N. D. Gupta, "Two-stage, hybrid flowshop scheduling problem," Journal of the Operational Research Society, vol. 39, pp. 359-364, 1988.

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

[7] T. Vidal, "The many ways of facing temporal uncertainty in planning and scheduling," Tatihou, France, 2004, pp. 9-12.

[8] J. L. Hunsucker and J. R. Shah, "Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment," *European Journal of Operational Research,* vol. 72, pp. 102-114,1994.

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

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

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

[12] L. Liu, H. Y. Gu, and Y. G. Xi, "Robust and stable scheduling of a single machine with random machine breakdowns," The International Journal of Advanced Manufacturing Technology, vol. 31, pp. 645-654, 2007.

[13] P. Kouvelis, R. L. Daniels, and G. Vairaktarakis, "Robust scheduling of a two-machine flow shop with uncertain processing times," IIE Transactions, vol. 32, pp. 421-432, 2000.

[14] L. Wang, L. Zhang, and D.Z. Zheng, "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), pp. 1157-1163, 2005.

[15] F. Ahmadizar, M. Ghazanfari, and S. M. T. Fatemi Ghomi, "Group shops scheduling with makespan criterion subject to random release dates and processing times," *Computers & Operations Research*, 37(1), pp. 152-162, 2010.

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

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

[18] G. E. Vieira, J. W. Herrmann, and E. Lin, "Rescheduling manufacturing systems: A framework of strategies, policies, and methods," *Journal of Scheduling,* vol. 6, pp. 39-62, 2003.

[19] S. R. Lawrence and E. C. Sewell, "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain," *Journal of Operations Management,* vol. 15, pp. 71-82, 1997.

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

[21] H. Matsuura, H. Tsubone, and M. Kanezashi, "Sequencing, dispatching, and switching in a dynamic manufacturing environment," *International Journal of Production Research,* vol. 31, pp. 1671-1688, 1993.

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

[23] J. C. Dunn, "Fuzzy relative of iosdata process and its use in detecting compact well-separated clusters," *J Cybern,* vol. 3, pp. 32-57, 1973.

[24] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. PAMI-1, pp. 224-227, 1979.

[25] D. J. Kim, Y. W. Park, and D. J. Park, "A novel validity index for determination of the optimal number of clusters," *IEICE Transactions on Information and Systems,* vol. E84-D, pp. 281-285, 2001.

[26] J. Shen, S. I. Chang, E. S. Lee, Y. Deng, and S. J. Brown, "Determination of cluster number in clustering microarray data," *Applied Mathematics and Computation,* vol. 169, pp. 1172-1185, 2005.