International Journal of Industrial Engineering, 30(6), 1365-1381, 2023

A NOVEL DECENTRALIZED APPROACH FOR PRODUCTION SCHEDULING

Shriprasad Chorghe[1, *], Makarand Shrikrishna Kulkarni[2], and Bhupesh Kumar Lad[1]

[1]Department of Mechanical Engineering
Indian Institute of Technology Indore
Indore, India
[*]Corresponding author's e-mail: shriprasadchorghe@gmail.com

[2]Department of Mechanical Engineering
Indian Institute of Technology Bombay
Mumbai, India

This paper proposes a novel decentralized approach for multi-stage job-shop scheduling. The method divides the larger job-shop scheduling problem into several smaller problems, which various agents solve. The collaboration among these agents ensures the exploration of globally superior solutions while allowing enhanced local exploration. Based on an extensive investigation, the current work shows that the proposed approach outperforms the centralized approach, especially for problems with increasing problem size, in a faster manner. Since the proposed approach is based on decentralized information processing, it is easily adaptable to next-generation cyber-physical manufacturing systems.

Keywords: Decentralized Approach, Industry 4.0, Multi-Agent System, Scheduling

(Received on July 4, 2023; Accepted on October 13, 2023)

1. INTRODUCTION

Globally, manufacturing industries are witnessing the fourth industrial revolution, Industry 4.0. While the third industrial revolution (Industry 3.0) used electronics and information technology to automate production, Industry 4.0 aims to transform conventional manufacturing systems into cyber-physical manufacturing systems by introducing machine intelligence and the Internet of Things (Rossit *et al*., 2019). Smart machines, smart jobs, smart processes and decentralized information processing and communication between various smart entities are typical characteristics that one may expect in such cyber-physical manufacturing systems (Yang *et al*., 2019). If utilized correctly, these characteristics can bring unprecedented leanness to industrial operations. Conventional shop floor planning approaches are inherently not designed for systems with such characteristics; hence, they may not work well for systems with these characteristics. Hence, the shop floor planning approaches must evolve to meet the requirements of the next-generation manufacturing systems (Li *et al*., 2022).

Scheduling is an essential shop floor planning aspect that helps in the effective utilization of limited resources while meeting customer demand. Manufacturing systems with newer capabilities of smart and connected systems require fundamental changes in addressing the scheduling problem. Multi-agent systems (MAS) provide one alternative: they possess characteristics like autonomous decision-making by individual agents and interaction between agents, which matches well with the characteristics of cyber-physical systems. Rossit *et al*. (2019) suggest that revisiting scheduling problems for cyber-physical systems (CPS) or Industry 4.0 is essential. Their review concluded that the real-time availability of the information significantly impacts scheduling and that scheduling of the future is a decentralized decision process. Parente *et al*. (2020) highlighted the need for evolving the production scheduling approaches in the context of Industry 4.0 to address elevated production flexibility and complexity requirements.

Similarly, Rossit and Tohmé (2022) highlighted the difficulty of using centralized scheduling approaches to deal with real-time data and arrive at a dynamic schedule for Industry 4.0. It is highlighted that due to the problem's complexity, simplifications are usually done in modeling, which may result in inferior performance. The authors highlighted that an agent-based approach could overcome this by allowing detailed modeling. Based on the above brief background, the literature on decentralized scheduling on multi-agent systems is further explored in the next section. It was identified that there are still various unanswered issues about a decentralized approach. The following section highlights some of the gaps in the literature.

Consequently, we propose a novel multi-agent-based decentralized production scheduling approach in this paper. The proposed decentralized approach reduces overall problem complexity by dividing a problem into smaller problem segments,

each of which can be solved independently by a separate decision-making entity. A major research challenge is to highlight the trade-offs between the need for a global view of the decentralized approach and the need for proper exploration of problems due to the complexity of the centralized approach. A detailed numerical investigation is done in this paper to address this issue. Based on the extensive investigation, the current work shows that the proposed approach outperforms the centralized approach, especially for problems with increasing problem size, in a faster manner.

The rest of the paper is structured as follows. Section 2 examines related work on agent-based decentralized scheduling. Section 3 describes the proposed approach and explains how the proposed decentralized approach works. Section 4 presents a comprehensive numerical investigation to test the performance of the proposed approach and identify various internal and external parameters that influence its performance. Section 5 concludes the work while pointing the way forward.

2. RELATED WORKS

This section reviews some recent work on agent-based decentralized scheduling aspects. Decomposition, negotiation, and communication are common in agent-based scheduling systems. Toptal and Sabuncuoglu (2010) classified agent-based distributed scheduling techniques based on communication mechanisms and information flow structure. Decomposition and negotiation-based classification is done by Parente *et al*. (2020). Most agent-based production scheduling techniques are developed in literature considering dynamic shop floor environments. Several methods involve flexible manufacturing shop floors with multiple resources for a given task. Agents then negotiate to find the best resources. In Nie *et al*. (2021) dynamic scheduling strategy, machine agents bid to automated storage and retrieval system agents. All possible solutions are sent to the digital twin agent by this agent. The digital twin agent optimizes the schedule based on multi-objective decision criteria and feasibility during the counting window. He *et al.* (2014) suggest agent-based dynamic scheduling for make-to-order manufacturing systems. In case of a change in demand or product type, job, system, and resource agents, discover updated schedules: production cost and lead time drive resource agents' bids. The system agent makes the schedule based on the best possible bid using heuristics or metaheuristics.

Li *et al*. (2014) developed an agent-based scheduling approach for a uniform, flexible job shop with machines of different speeds. Sorting machines by speed and sorting jobs using the dispatching rule is done. Agents set a specified job at the end of each machine and assign it to the machine with the fastest completion time. In Liu *et al*. (2022), resource agents auction jobs while consumer agents compete to reduce tardiness. After receiving information from resource agents, consumer agents bid on processing time slots and pricing. The resource agent then maximizes profit by solving the winner determination problem using simulated annealing. Resource agents assign resources for tasks.

Similarly, Weiss-Cohen *et al.* (2017) also used a negotiation approach for their multi-agent-based scheduling for a flexible shop floor. Authors have also examined the efficacy of using agent-based systems for single-machine scheduling problems. For example, Kaplanoğlu (2014) proposes a collaborative multi-agent-based optimization technique for single-machine scheduling with sequence-dependent setup time and maintenance constraints. Authors could show that even for a single-machine scheduling problem, a multi-agent system with its distributed mechanism reduces the computation load of central optimization and responds faster to dynamic events.

Agent-based, decentralized scheduling is also used in literature for assembly lines and flow shops. Vatankhah Barenji and Hashemipour (2019) devised a multi-agent distributed approach for production scheduling, yielding enhanced uptime and productivity. Maoudj *et al*. (2019) made a distributed multi-agent system (DMAS) for scheduling and controlling Robotic Flexible Assembly Cells (RFAC). It solves the problem of scheduling product operations, which involves allocating assembly cells; three different kinds of agents are suggested. The execution of the intended sequence on the robot is handled by a remote agent (RA). Local agents (LA) work together to divide tasks and follow the dispatching rules. The Supervisory Agent (SA) prioritizes operations.

Shi *et al*. (2021) suggest a multi-agent-based dynamic scheduling optimization of a hybrid flow shop. It has a scheduling agent, a machine agent, a dispatching agent, and an inspection agent. The job of the dispatching agent is to find out what jobs are planned and put them in order of importance based on their process data. The job of the machine agent is to keep track of how the machine works in real time. The job of the inspection agent is to get the processing quality inspection data for the jobs in an inspection buffer. Wang *et al*. (2018) talk about the same kind of role for a scheduling agent when they talk about how to make sure that precast construction components are delivered on time. The scheduling agent uses a genetic algorithm (GA) to reduce lateness and make the best schedule in a flow shop setting. Resource agents and the job management agent assign resources to tasks using the contract net protocol. This assignment reduces the amount of time that schedule execution has to wait.

Wang *et al.* (2022) scheduled jobs on a parallel workstation system using a digital twin-based multi-agent system. On the conveyor, the task management agent gets information about the parts arriving at the target workstation. It sends the information to all the gantry crane agents that are currently available so they can wait for their turn to move. With particle swarm optimization, the task management agent assigns tasks to parts on the conveyor. This assignment is done for all the

parts that are on the conveyor. Through experiments, the authors showed that the average use rate of all workstations is up by 17.39% and that the standard deviation is down by 83.31% compared to traditional methods. In Saeidlou *et al.* (2019), ontology-based knowledge management is used with a multi-agent system. The problem consists of jobs with multiple operations to be processed by multiple manufacturers. The parts of their agent system are job, operation pair, and manufacturer agents. The people in charge of a job agree on which manufacturer should do the scheduling. Agents for operation pairs make up operation pairs at the manufacturer. Operation pairs are two operations that are done right after each other. Manufacturer agent sequences and re-sequences these pairs. Ontology is used to avoid any constraint violation.

Instead of focusing only on jobs and resources in the agent-based scheduling system, many approaches also use agents of various extended elements of the factory, which work in a heterarchical and decentralized manner. For example, Cupek *et al*. (2016) present an agent-based approach to scheduling a short production series. The agent system comprises a client, supervisory, order, and workplace agent. Klein *et al*. (2018) also deploy an agent-based approach to generate a scheduling-support system. These agents represent the production units, storage centers, transportation means, and production orders. These agents interact through auction-based negotiation.

Another research category on agent-based scheduling focuses on learning-based approaches (Rossit and Tohmé, 2018; Ma *et al*., 2022; Yan *et al*., 2022). In general, scheduling agents learn from past knowledge in such approaches and act accordingly in case of any dynamic events.

Some agent-based scheduling research also focuses on developing a framework or Cyber-Physical Production Systems architecture for scheduling. For example, Sun *et al.* (2022) focus on designing intelligent manufacturing systems by integrating various information systems. Similarly, in Seitz *et al*. (2021), a platform-independent multi-agent system addresses two Industry 4.0 scenarios: order-controlled production and adaptable factory. Order control production relates to the flexibility of the production network within and across production facilities to produce new products.

The aforementioned concentrated literature study on agent-based decentralized scheduling shows that the techniques mainly focus on bidding and negotiation-based systems. Each agent contributes their information to the bid. Some of the agents analyze bids by one or more criteria. Such bidding-based systems have shown themselves efficiently determining the best course of action when faced with various possibilities in a flexible production environment. The literature clarifies how well-suited these methods are for dynamic situations with disturbances like changes in demand, lead times and equipment problems. Several of the existing methods ignore operational dependencies. Problem decomposition becomes very challenging due to the interdependencies of processes and the decision factors for various agents. For their network to provide the requisite solution quality in a fair amount of time, decomposition needs proper decision-making entities and segregating decision variables among them. Such decentralized scheduling methods based on decomposition are scarce in the literature. Although decomposition aids in more effective local-level exploration, adopting such agent-based methodologies still needs thorough study to provide adequate quality for overall decision-making. Based on the above description, the current paper aims to create a decentralized agent-based scheduling approach concentrating on decomposing scheduling problems into smaller problems. The problem of multi-stage multi-machine system is decomposed into small problems for job type agents, machine agents and inter-stage agents in this paper. Each agent has its localized objective, and through communication among these agents, the approach achieves a solution for the entire problem. The decomposition and communication used in the paper is the key to addressing the challenge of interdependencies of processes and decision factors. In addition, the paper examines the method's efficacy and discovers a few intriguing insights via thorough assessment.

3. DECENTRALIZED APPROACH

The system considered in this work is a multi-stage, multi-machine shop floor. There are different job types, and each job type has several jobs. Each job is processed by one machine at each stage. The job type, quantity of jobs, and Due Date (DD) are all specified in the customer order.

Due to the dynamic nature of the next-generation production system, planning horizons are shorter. When planning for the next horizon, we assume tasks from the previous horizon do not occupy machines. Conventional approaches to production scheduling are typically centralized because a single computational entity is responsible for making allocation decisions. The computational involvedness of any scheduling problem grows exponentially with the number of jobs, machines, and stages in any production system, reducing the responsiveness of such decision-making systems. It precludes any possibility of dynamic scheduling, thereby restricting the achievement of possible leanness, which further leads to a fundamental rethinking of the approach to scheduling problem. In this paper, we design a novel agent-based decentralized approach for production scheduling. Figure 1 explains the working of the developed decentralized production scheduling approach. The basis of the approach is that a more complex production scheduling problem is decomposed into multiple smaller and relatively more straightforward problems for different agents, and through communication among these agents, a solution to the original complex problems is obtained. Decomposition helps in better local-level modeling and exploration, while communication strategies among agents are uniquely designed to ensure getting a solution closer to the centralized approach. In the present

multi-agent systems, there are three agents, viz., job type agents, machine agents, and inter-stage agents, which make local-level decisions and communicate with each other to reach the final shop floor-level decision.

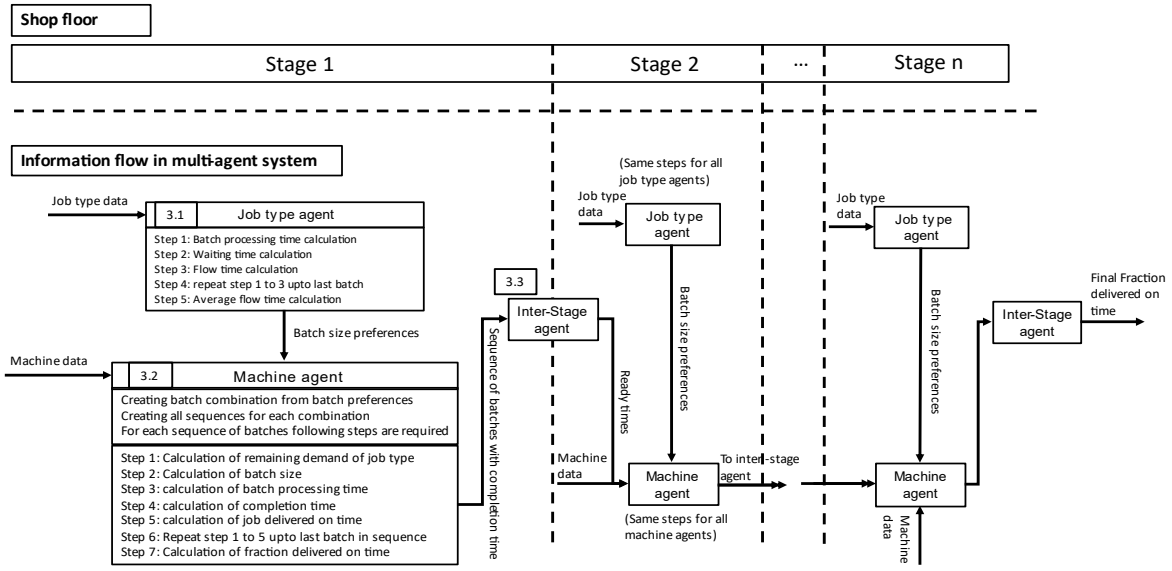

Figure 1. Graphical representation of decentralized approach for flow of decision-making

The following subsections explain the details of each of these agents and their work. Further, these subsections are marked in Figure 1 to explain the flow of information through the shop floor.

3.1. Job type Agent

A job type agent is a digital entity of a specific job type. In this paper, each job type agent is responsible for making local-level decisions about their respective batch sizes. It determines the batch size to minimize average flow time (AFT). To minimize AFT, a job-type agent requests the information for the processing time and Sequence Dependent Setup Time (SDST) from relevant machine agents in the route of the job. This information is used to calculate the AFT of a particular job type on a given machine for considered batch size. Job type agents evaluate a discrete number of batch sizes to optimize AFT. The detailed working of a job-type agent is described below.

The job type agent evaluates multiple batch sizes for their objective function value, i.e., AFT and prepares a list of batch sizes in ascending order of their AFT. Job type agent follows an algorithm to calculate the value of AFT and prioritize batch sizes in the list to be forwarded. Table 1 describes notations, followed by the steps of the algorithm.

Table 1. Notations for the algorithm of job type agent

BPT_i	Batch processing time for the iteration
WT_i	Waiting time for a batch of jobs in a queue for the iteration
ST	Average setup time for a machine for a job type The average setup time for the j^{th} job type is calculated as $ST_j = \frac{\sum_{i=1}^{i=n} SDST_{ij}}{n}\ i \neq j$ Where $SDST_{ij}$ is sequence-dependent setup time from i^{th} job type to j^{th} job type for n job types
JPT	Processing time for a machine to complete a job from job type
FT_i	Time that a batch of jobs spends in a system for i^{th} iteration (Flow time)
BS_i	Batch size of arriving batch on a machine for a job type for i^{th} iteration
AFT	Average Flow Time
n	number of iterations

f

Step 1: Calculation the batch processing time for i[th] iteration

$$BPT_i = ST + BS_i * JPT \quad (1)$$

i=1, 2,..,n; where i is the batch number arriving on a machine of the same job type

Step 2: Calculation the waiting time for i[th] iteration

$$WT_i = \sum_{k=1}^{k=i-1} BPT_k \quad (2)$$

k=1, 2…i-1; where k is the batch number arriving on a machine of the same job type

Step 3: Calculation the flow time for i[th] iteration

$$FT_i = WT_i + BPT_i \quad (3)$$

$i = i + 1$

Step 4: Repeat steps 1 to 3 until all batches are considered; otherwise, go to step 5
Step 5: Calculation of average flow time

$$AFT = \frac{\sum_{i=1}^{n} FT_i}{n} \quad (4)$$

The following example explains the algorithm. A job type has a demand of 20 jobs. For a batch size of 4, five batches would have arrived on the shop floor. For the average setup time of 7 and job processing time of 2, the batch processing time is 15.

The first batch's waiting time is 0, and the flow time is 15. The flow time for succeeding batches are 15, 30, 45, 60 and 75, and the average flow time is 4. The parameter percentage of job type agent forwarding (PJTAF) determines the number of batch sizes forwarded by the job type agent. For example, when a job type agent evaluates 10 batch sizes, the PJTAF of 40 would result in 4 batch sizes being forwarded to the machine agent. The number of batch sizes to be forwarded is communicated to the machine agent as a list.

3.2. Machine Agent

The machine agent enables a machine by solving a single machine scheduling problem and communicating with the job type agent and inter-stage agent to get the necessary information. Machine agents generate preferred sequences of batches received from job type agents for different job types based on the objective values of Fraction delivered on time (FDOT), where due dates for batches are calculated using the total work content method (Baker, 1984). In the Total Work Content (TWC) method, the complete available time between the ready time and due date is divided among operations based on the weights of the machine's processing time on which the operation would be performed. The machine agent decides the processing sequence for batches with their completion times and forwards it to the inter-stage agent. The machine agent is responsible for evaluating multiple sequences and selecting the sequence to be forwarded to the inter-stage agent. The machine agent receives input from multiple job-type agents and facilitates the sequence forwarding to the inter-stage agent. The working of a machine agent is described below.

Job types arriving on the machine form multiple batch size combinations. For example, if two job types are arriving on a machine and the preferences of batch sizes given by the Job Type Agent are [4, 8] and [4, 20] for job types J1 and J2, respectively, the total possible batch size combinations would be as shown in table 2.

Within each batch combination, multiple sequences are possible. For example, in combination with number 4 from Table 2, the possible sequences representing job type numbers are [1 1 1 2], [1 2 1 1], [1 1 2 1] and [2 1 1 1]. The machine agent follows an algorithm to decide the sequence of batches to be forwarded by calculating the value of the objective function for the corresponding sequence. Table 3 Describes notations, followed by the steps of the algorithm.

Table 2. Batch combinations

Combination no.	Batch size (J1)	Batch size (J2)
1	4	4
2	4	20
3	8	4
4	8	20

Table 3. Notations for the algorithm of machine agent

D_i	Demand for the i^{th} job type
BS_i	Batch size for i^{th} job type
BS_i*	Initial batch size received from job type agent for i^{th} job type
JPT_i	Job processing time of i^{th} job type
BPT_i	Batch processing time of i^{th} job type
ST_i	Average sequence-dependent setup time for i^{th} job type
CT_j	Completion time for the j^{th} position in sequence
PT_j	Processing time for a batch at the j^{th} position in sequence
$JDOT_i$	Jobs delivered on time for i^{th} job type
DD	The due date for the job on the considered machine
n	number of job types,
l	total length of the sequence

Algorithm:

Step 1: Calculation of demand for i^{th} job type at the j^{th} position in the sequence

$$D_i = D_{i-1} - BS_{i-1}, if\ i \neq 1 \tag{5}$$

where i= Job-ID, j= position number

i= 1, 2 …n; j=1, 2…. l

Step 2: Calculation of batch size for i^{th} job type at the j^{th} position in the sequence

$$BS_i = BS_i^*\ if\ BS_i < D_i, \qquad D_i\ otherwise \tag{6}$$

Step 3: Calculation of batch processing time for i^{th} job type at the j^{th} position in the sequence

$$BPT_i = ST_i + BS_i * JPT_i \tag{7}$$

Step 4: Calculation of Processing time and completion time for a batch at the j^{th} position in the sequence

$$PT_j = BPT_i \tag{8}$$

$$CT_j = \sum_{k=1}^{k=j} PT_k \tag{9}$$

where k=position number
Step 5: Calculation of jobs delivered on time for i^{th} job type at j^{th} position in sequence

$$JDOT_i = JDOT_i + BS_i, if\ CT_j < DD \tag{10}$$

Step 6: Repeat steps 1 to 5 until consideration of all batches; otherwise, go to step 7

f

Step 7: Calculation of Fraction Delivered On Time (FDOT)

$$FDOT = \frac{\sum_{i=1}^{i=n} JDOT}{D \times n} \tag{11}$$

The sequences with the best value of the objective are sorted in descending order. For example, sequences with job type IDs [1 2 3 4], [4 2 1 3], [1 4 2 3] and FDOT of 1, 0.7 and 0.8, respectively, are arranged in descending order as [1 2 3 4], [1 4 2 3] and [4 2 1 3]. The machine agent forwards the best-performing sequence from the sequences in the preference list to the inter-stage agent. Machine agent encounters varying problem complexity. Depending on the problem's complexity, the machine agent takes different modes of a decentralized approach to obtain the optimum sequence.

3.3. Inter-stage Agent

To prepare an optimal sequence of batches, the machine agent requires ready times of batches from the previous stage. The inter-stage agent acts as a coordinator between stages. From machine agents in the same stage, sequences of batches with their completion time are received at inter-stage agents. The inter-stage agent forwards the received completion times to machine agents in the next stage as ready times. Based on which machine processes which job type, the inter-stage agent distributes ready times for each machine agent in the next stage. The inter-stage agent calculates FDOT for the preceding stage by taking the mean of FDOT from all the machines in the preceding stage.

It is also possible that machine agents forward multiple solutions to the inter-stage agent. In that scenario, the inter-stage agent forms a combination of independent solutions received from individual machines. The interstage agent sorts These multiple combinations based on the on-time delivery performance, and one best combination is forwarded to the machines in the next stage.

For example, let machine 1 with job type 1 and job type 2 and machine 2 be with job type 3 and job type 4 as arriving job types are two machines on the shop floor in stage 1. Let machine 1 agent and machine 2 agent correspond to machine 1 and machine 2, respectively. The solution received by the inter-stage agent by machine 1 agent is [2 1 2 1] with FDOT equal to 0.72 and [1 1 2 2] with FDOT equal to 0.81, and the solution received by machine 2 agent be [3 4 3 4] with FDOT equal to 0.8 and [3 3 4 4] with FDOT equal to 0.91. then, four combinations of solutions are formed as follows:

Combination 1: [2 1 2 1] and [3 4 3 4], combined FDOT = (0.72+0.8)/2 =0.76

Combination 2: [2 1 2 1] and [3 3 4 4], combined FDOT = (0.72+0.91)/2=0.815

Combination 3: [1 1 2 2] and [3 4 3 4], combined FDOT = (0.81+0.8)/2=0.805

Combination 4: [1 1 2 2] and [3 3 4 4], combined FDOT = (0.81+0.91)/2=0.86

Then, the formed combination would be sorted by inter-stage agent in descending order as combination 4, combination 2, combination 3, and 1. Completion times of batches in combination 4 would be transferred to machines in the next stage.

Without an inter-stage agent, if machines forward multiple solutions, then machines in the next stage would have to evaluate their solutions corresponding to each of the received ready times. The complexity grows as the number of machines in each stage grows because each machine must choose the same ready time from machines in the preceding stage to get a feasible overall solution. As can be seen, the significance of the inter-stage agent lies in its operational mechanism of controlling the growing complex network when multiple solutions are forwarded. It would be more significant if the material flow is comparatively more complex, such as job shops, flexible shops, and shops with re-entrant jobs.

4. NUMERICAL INVESTIGATION

An exhaustive numerical investigation is performed to assess the performance of the decentralized approach. It is compared with the centralized approach for various problem categories to understand the efficacy of the proposed decentralized approach. Various problem categories are constructed by varying the number of job types on each machine (n), the number of machines in each stage (m) and the number of stages (s).

All the categories and their corresponding problem sizes (for complete enumeration) are listed in Table 4. It can be seen from Table 4 that problem complexity increases exponentially from problem category 1 to problem category 5. With predefined problem parameters given in Table 5, the problem size for problem category 1 is 105, whereas for problem category 5, the problem size is 10^{229}.

Table 4. Problem categories and their properties

Problem categories	n	m	s	Problem size (Complete enumeration)
Problem Category 1	2	1	2	$3.2 * 10^5$
Problem Category 2	2	2	2	$1.03*10^{11}$
Problem Category 3	3	1	2	$1.35*10^{12}$
Problem Category 4	4	2	2	$5.99*10^{40}$
Problem Category 5	7	3	3	$4.84*10^{229}$

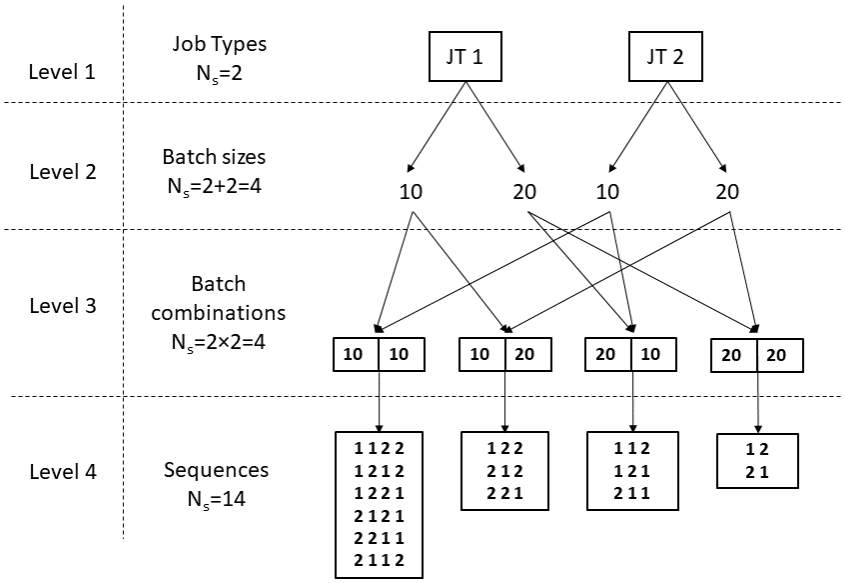

Figure 2. Increasing complexity through different levels for a single machine

Table 5. Problem Parameters

Processing Time	Uniform (1,5)
Sequence-dependent setup time	Uniform (6,10)
Demand	20
Step size	4

We generate integer-valued processing and setup times from uniform distribution (Table 5). The generated time values are assumed to be in minutes. Step size and demand are problem parameters which affect problem complexity in terms of time. A step is the next batch size for evaluation, and the step size is the incremental quantity added in the preceding batch size. For example, a step size of 4 would make it possible to evaluate batch sizes of 4,8,12,16,20 and so on. A decrease in step size increases computational complexity as the smaller batch sizes for a given demand increase the total number of possible batch size combinations at a machine. Another critical factor that contributes to the higher number of batch sizes is demand. For constant batch size, if demand for a job type increases, the corresponding possible number of batch sizes would also increase, increasing the number of sequences. We limit the work to a step size of 4 and demand of 20 for all problem categories. Further, for each problem category, two cases are considered, viz. slack and tight due dates cases, where a tight due date is more likely to produce a schedule with late jobs than a slack due date.

Initially, an attempt is made to solve the problem using complete enumeration, viz. brute force approach. The results are in serial number 16 ($CA_{BruteForce}$, where CA is the Centralized approach) in Table 6. It can be seen from the table that complete enumeration is feasible only for problem category 1 because, with increasing complexity, evaluating all possible sequences is computationally inefficient. Conventionally, high-complexity issues are addressed using metaheuristics to limit

the search scope in solution space. One can use any metaheuristics to solve such problems; however, the difference in solution quality and time for various metaheuristics is insignificant compared to the difference in centralized and decentralized approaches (Upasani *et al*., 2017). Hence, the focus of this paper is not to identify a better metaheuristic approach but to compare the decentralized approach with any of the centralized approaches. In this work, we have utilized a genetic algorithm (GA) as metaheuristics.

Further, we have gradually increased the population size in the GA from 10 to 40000, corresponding to the gradual increase in solution space exploration for a centralized approach. It provides an idea about computational expenses vs accuracy for centralized GA compared to a decentralized approach. Results are presented in serial numbers 17-22 (CA_{GA10} to $CA_{GA40000,}$ which stands for centralized approach with GA using a population size of 10 to 40000) in Table 6. In this work, GA implements the restart scheme by Ruiz and Maroto (2006) to avoid local optima. Other parameters of the GA are as follows: Crossover probability: 0.85, Mutation probability: 0.15, Elitism rate 0.9, Termination criterion: 70, Restart criteria: 10.

All cases are also solved using the proposed decentralized approach, and Table 6 presents the results. In the decentralized approach, job-type agents evaluate all possible combinations as the problem size is small and provide a solution in batch sizes to machine agents.

The problem complexity at the machine agent depends on the number of solutions received from the job type agents. This work considers five possibilities with increasing problem complexity, viz., 20%, 40%, 60%, 80%, and 100% forwarding by the job type agents (PJTAF). These cases are shown in Table 6 in serial number 1 to 5 (DA_{20} to DA_{100}, where 20 and 100 is PJTAF). The gradual increase in time in all cases results from increasing complexity due to higher PJTAF. Brute force at the machine agent level is feasible only for problem categories 1-3. For problem categories 4 and 5, the complexity of complete enumeration increases as forwarding multiple batch sizes results in many possible sequences of batches.

For individual problem categories, at machine agents, there exists the opportunity to perform brute force up to a certain 'level' depending upon the number of solutions forwarded from job type agents and corresponding batch sizes. To understand the concept of 'level' in a problem, let us consider an example of a single machine on which two job types are arriving. From Figure 2, when the number of job types increases at level 1, the corresponding sequences also increase at level 4. At level 2, a discrete number of batch sizes are evaluated, and the number depends on the step size. At level 3, combinations of the batches are formed from the batch sizes. With increasing PJTAF, solutions at level 3 would increase, leading to an increase in time. At level 4, all possible sequences are created. Creating all possible sequences is feasible for problem categories 1, 2 and 3 owing to relatively minor memory requirements for their problem size for a decentralized approach. However, subsequent problem categories can only be done by restricting the number of created sequences.

The compounding effect is due to solution forwarding, increasing information at level 4. Hence, metaheuristics (in this case, GA) are applied to further control sequences selectively at levels 3 and 4. In Table 6, serial No. 6 to serial No. 10 shows the results of the use of GA at level 4 in the machine agent and PJTAF of 20, 40, 60, 80 and 100 by job type agent (DA_{GA20} to DA_{GA100}, which indicate the use of GA as metaheuristic and PJTAF from 20 to 100). It can be seen from the table that for problem category 5 with GA only, beyond PJTAF of 60, the solution is not possible. Hence, GA is then extended to level 3. Table 6, serial No. 11 to 15, shows the use of GA both in level 3 and 4 at machine agents with varying PJTAF (DA_{GAGA20} to $DA_{GAGA100}$, which indicate GA at level 4 and 3 with PJTAF from 20 to 100). In summary, in the conventional centralized approach, the possibility of brute force is exhausted in problem category 1. In contrast, for the decentralized approach, local-level brute force existed much beyond problem category 1, and it depends on percentage forwarding (PJTAF) by job type agent and step size for batch sizes.

Algorithms of the decentralized and centralized approaches are coded in MATLAB R2019b. A numerical investigation is performed on a computer system with Windows 10 platform having Intel(R) Core (TM) i7-3770 CPU @ 3.40GHz processor and 8 GB RAM. Also, the time values shown in Table 6 after running the code are in seconds.

4.1. Observations and Discussion

In order to comprehend the results presented in Table 6, the best-performing approaches among varieties of centralized and decentralized approaches are plotted for tight and slack due date settings (figure 3) across five problem categories. By best performance, we mean maximum FDOT in the shortest time. For example, for problem category 1, for tight due date setting, $CA_{GA10000}$ and $CA_{GA20000}$ give the same performance, but $CA_{GA10000}$ achieves it in a lower time. So, $CA_{GA10000}$ is the best-performing centralized approach for problem category 1. Similarly, for problem category 2, the same performance is obtained for the tight due date setting for DA_{40} and DA_{60}; however, DA_{40} achieves it in a lower time. Some key observations from Figure 3 and Table 6 are discussed below.

f

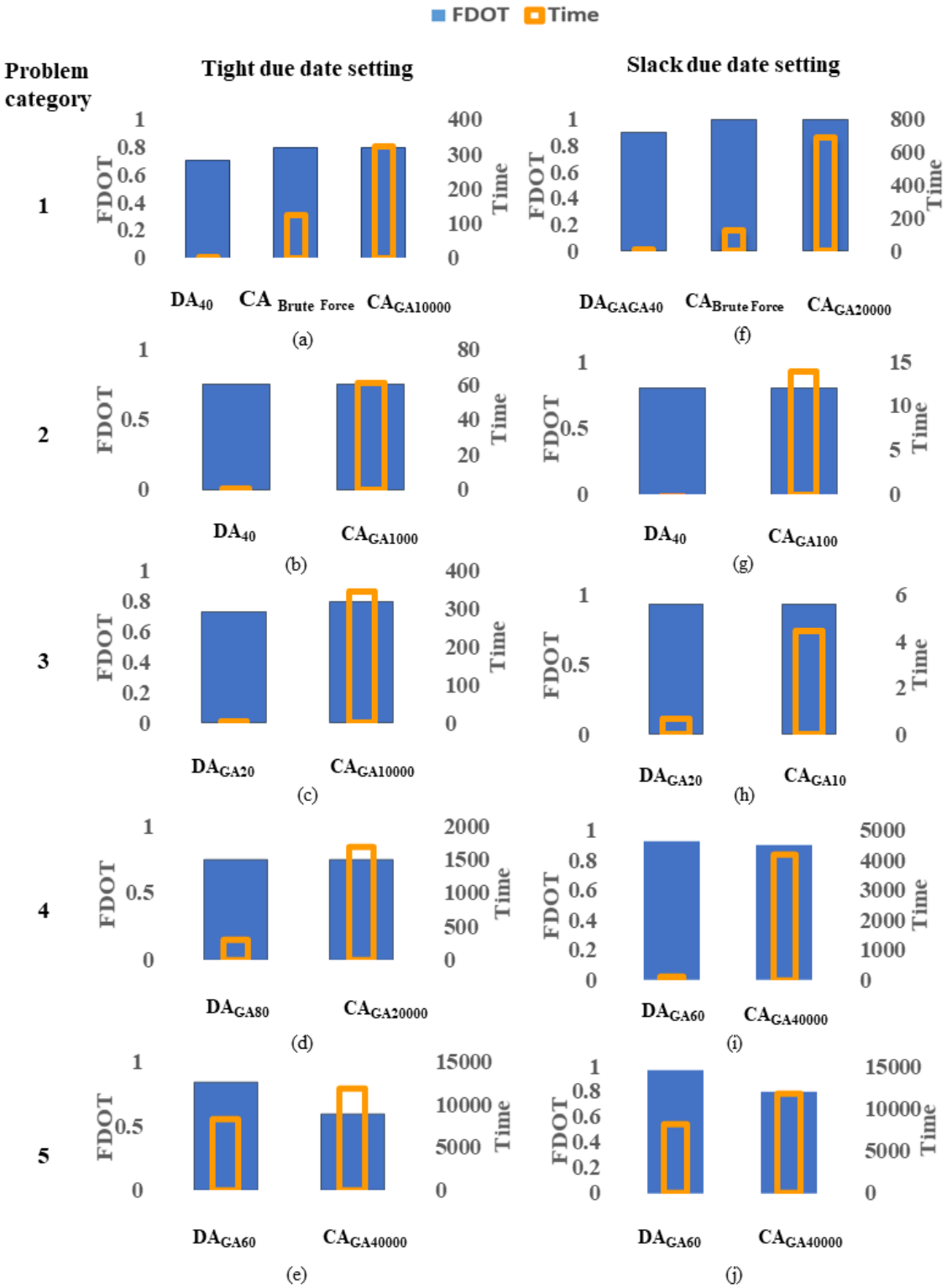

Figure 3 (a) to (i): Comparison of best-performing decentralized and centralized approaches for different problem categories and due date settings

f

Table 6. Analysis of Decentralized and Centralized Approaches with FDOT as Machine Agent's Objective

Sr. No	Approach	Problem Category 1				Problem Category 2				Problem Category 3				Problem Category 4				Problem Category 5			
		Tight DD		Slack DD		Tight DD		Slack DD		Tight DD		Slack DD		Tight DD		Slack DD		Tight DD		Slack DD	
		FDOT	Time	FDOT	Time	FDOT	Time	FDOT	Time	FDOT	Time	FDOT	Time	FDOT	Time	FDOT	Time	FDOT	Time	FDOT	Time
1	DA_{20}	0.6	0.02	0.8	0.03	0.7	0.09	0.75	0.1	0.733	4.93	0.933	5.15	-	-	-	-	-	-	-	-
2	DA_{40}	**0.7**	**0.05**	0.8	0.06	**0.75**	**0.15**	**0.8**	**0.17**	0.733	8.84	0.933	9.16	-	-	-	-	-	-	-	-
3	DA_{60}	0.7	0.07	0.8	0.07	0.75	0.29	0.8	0.32	0.733	26.81	0.933	28.17	-	-	-	-	-	-	-	-
4	DA_{80}	0.7	0.12	0.8	0.12	0.75	0.42	0.8	0.4	0.733	83.05	0.933	85	-	-	-	-	-	-	-	-
5	DA_{100}	0.7	0.3	0.8	0.32	0.75	0.57	0.8	0.62	0.733	614.15	0.933	638.09	-	-	-	-	-	-	-	-
6	DA_{GA20}	0.6	0.47	0.8	0.47	0.7	0.9	0.75	0.9	**0.733**	**0.55**	**0.933**	**0.67**	0.575	1.29	0.8	1.32	0.552381	4.18	0.761905	4.3
7	DA_{GA40}	0.7	1.84	0.8	1.84	0.7	3.55	0.8	3.52	0.733	4.31	0.933	4.14	0.725	19.14	0.9	19.24	0.752381	514.61	0.952381	517.42
8	DA_{GA60}	0.7	4.1	0.8	4.17	0.7	7.97	0.8	8.3	0.733	13.88	0.933	14.11	0.725	96.2	**0.925**	**96.34**	**0.847619**	**8266.62**	**0.971429**	**8099.21**
9	DA_{GA80}	0.7	7.42	0.8	7.29	0.7	14.34	0.8	14.33	0.733	33.7	0.933	32.61	**0.75**	**303.36**	0.9	299.5	-	-	-	-
10	DA_{GA100}	0.7	11.77	0.8	11.33	0.7	22.36	0.8	22.16	0.733	64.44	0.933	63.84	0.725	724.55	0.9	740.33	-	-	-	-
11	DA_{GAGA20}	0.6	9.44	0.8	9.66	0.7	18.03	0.75	18.3	0.733	11.27	0.933	11.10	0.7	25.83	0.85	25.21	0.73333	84.57	0.8667	86.21
12	DA_{GAGA40}	0.7	9.49	**0.9**	**9.69**	0.7	18.64	0.8	18.8	0.733	10.24	0.933	10.61	0.7	24.59	0.875	24.97	0.704762	83.7	0.9333	81.42
13	DA_{GAGA60}	0.7	9	0.8	9.5	0.65	19.32	0.8	19.42	0.733	10.99	0.933	10.66	0.7	23.93	0.825	26.43	0.67619	82.24	0.9333	83.15
14	DA_{GAGA80}	0.5	8.81	0.5	9.08	0.5	17.73	0.75	18.74	0.733	11.04	0.933	10.97	0.65	23.82	0.8	24.99	0.695238	81.03	0.895238	78.33
15	$DA_{GAGA100}$	0.5	9.24	0.5	8.98	0.55	18.42	0.5	17.7	0.667	10.43	0.933	10.94	0.625	24.29	0.75	25.5	0.594706	83.49	0.857149	78.81
16	$CA_{\text{Brute Force}}$	**0.8**	**121.7**	**1**	**122.47**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	CA_{GA10}	0.5	4.5	0.8	4.84	0.6	8.4	0.7	8.78	0.733	2	**0.933**	**4.45**	0.375	2.79	0.675	8.9	0.228571	21.26	0.295238	20.57
18	CA_{GA100}	0.7	6.9	0.9	7.21	0.7	13.1	**0.8**	**13.96**	0.533	5	0.933	7.23	0.5	8.55	0.675	15.1	0.314286	42.42	0.52381	41.79
19	CA_{GA1000}	0.7	31.5	0.9	32.76	**0.75**	**60.9**	0.8	64.6	0.733	32.4	0.933	36.45	0.65	67.29	0.775	78.44	0.428571	262.03	0.619048	244.9
20	$CA_{GA10000}$	**0.8**	**321.09**	0.9	320.09	0.75	655.07	0.8	674.19	**0.8**	**345.14**	0.933	340.62	0.725	802.49	0.825	779.6	0.52381	2522.3	0.714286	2438.1
21	$CA_{GA20000}$	0.8	717.35	**1**	**691.71**	0.75	1539.43	0.8	1595.86	0.8	756.52	0.933	759.57	**0.75**	**1700.4**	0.875	1693.09	0.571429	5450.6	0.752381	5215.8
22	$CA_{GA40000}$	0.8	1686.62	1	1672.21	0.75	4043.29	0.8	4075.98	0.8	1711.7	0.933	1888.1	0.75	3916.74	**0.9**	**4204.86**	**0.590476**	**11884**	**0.8**	**11821**

f

Observation 1:
Brute force for the centralized approach ($CA_{\text{brute force}}$) was feasible only for problem category 1, as the time required to evaluate all possible cases increased exponentially beyond problem category 1 (i.e., with increasing problem complexity). Conventionally, heuristics/metaheuristics are used to deal with such complex problems. In the present example, for smaller problem size (problem category 1), GA reached the optimal solution given by the brute force (figure 3(a) and (f)) only beyond certain population size (10000 and 20000 for tight and slack due date case respectively) used in GA and because of the inherent nature of the algorithm, GA evaluates some solutions within solution space repeatedly. As a result, for smaller problem sizes, GA even took more time to reach an optimal solution compared to brute force. On the other hand, the decentralized approach produced performance comparable to brute force in significantly less time. In terms of objective function (FDOT) value, the decentralized approach gave 87.5% and 90% of FDOT for tight and slack due date settings, respectively. Regarding evaluation time, the distributed approach took significantly less time than brute force (0.04% for tight due date setting and 7.9% for slack due date setting).

Observation 2:
It can be easily realized that the ability of GA to reach the optimal solution strongly depends on population size. The higher the population, the better would be the performance. However, an increase in population size increases the algorithm's computational complexity and ultimately results in higher computational time. Also, with the increase in problem complexity from Category 1 to Category 5, the requirement of higher population size for GA to reach a better solution is visible (Centralized approach in Figure 3 (a) to (e)), as with centralized GA only.

It leads to an important observation that if the problem size increases, centralized GA may require a significantly larger population size to reach an optimal or near-optimal solution. It increases the computation requirement, and one may have to compromise with smaller population size and relatively poor-quality solutions. With the decentralized approach, this compromise would be much smaller, as observed in Figure 3. For the decentralized approach to obtain reasonable performance, the required time increases slowly with increasing complexity. The decentralized approach matches and gradually surpasses the performance of the centralized approach with increasing problem complexity in a much shorter time. However, some considerations must be considered to maximize the performance of the decentralized approach. Observation 3 makes these considerations clear.

Observation 3:
Going for 100% forwarding by a job agent is optional to get the best result. It can be seen from Table 6, serial numbers 1-5, that even with a lower number of solutions forwarded by a job type agent, the best possible performance was received for all the problem categories. Thus, using lower percentage forwarding by job type agents may be helpful to get quicker solutions.

If GA is applied at various levels, as discussed before. A higher number of solutions forwarded by job-type agents can reduce the performance. For example, figure 4 (a) and (b) summarizes the results reported in serial number 6-15 for all five problem categories for slack due date cases.

Similar results are reported for tight due date cases also. These results are because increased PJATF increases the problem complexity for subsequent GA at machine agents for various levels. Thus, GA fails to explore the solution space properly. Thus, higher performance may be obtained at a lower value of PJTAF. Machine learning models could be used to identify the correct value of PJTAF based on input variables.

The significance of the decentralized approach for problems with high complexity is highlighted in Figure 5. For the problem, the category 5 decentralized approach converges to better performance in less time than the centralized approach, as the curve of trend converges to a lower mean FDOT in the case of the decentralized approach. In contrast, it converges to a higher mean FDOT for the centralized approach.

Table 7 shows time variations across problem categories with minimum and maximum observed values across approaches within the approaches mentioned in the row for the corresponding problem category in the column. From DA_{20} to DA_{100}, the time increases rapidly across problem categories. From DA_{GA20} to DA_{GA100}, time increases less rapidly compared to DA_{20} to DA_{100}. For DA_{GAGA20} to $DA_{GAGA100}$, the rate of increase of time reduces further; within these variations, time values are tightly close in the graph. These results can be attributed to the total evaluated sequences by the machine agent. With GA at level 4 and level 3, total evaluated sequences are reduced across problem categories, leading to a less rapid increase in time. Time calculations are presented further, which justifies time observations.

f

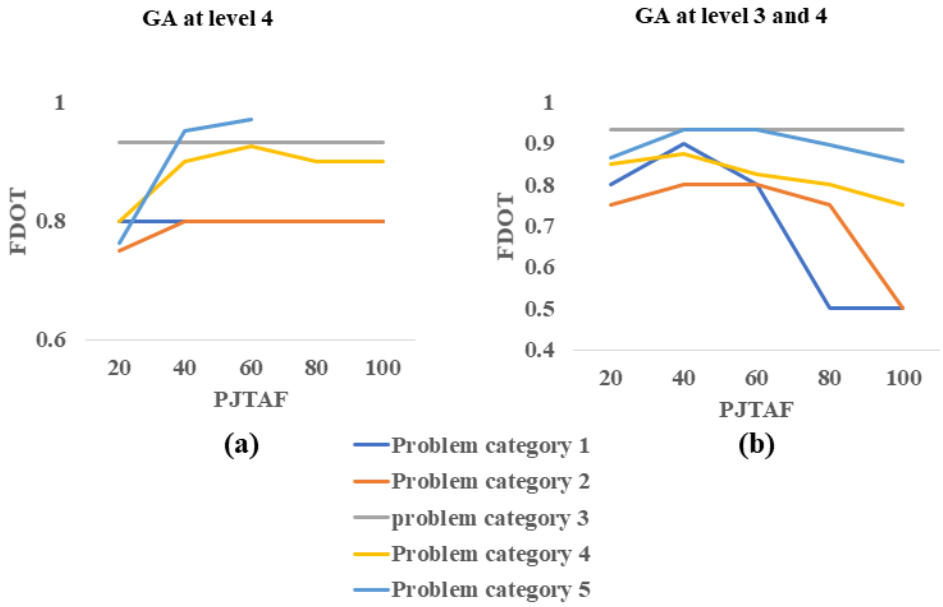

Figure 4. (a) and (b): Performance of decentralized approach with GA across problem categories for slack due date

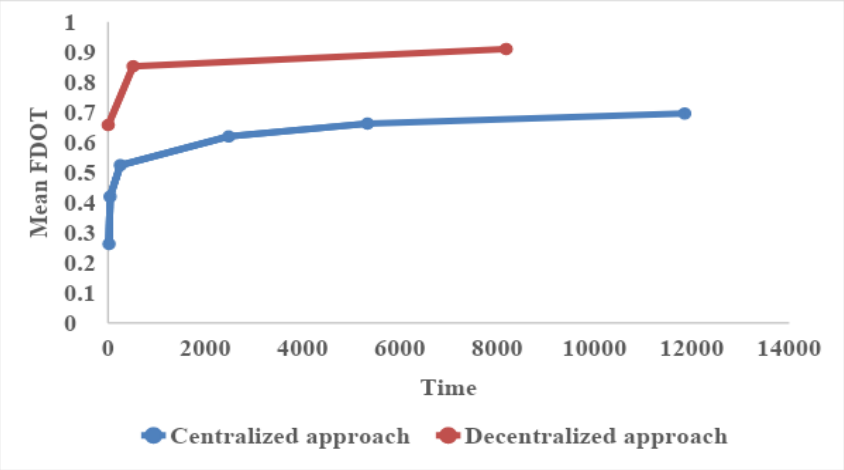

Figure 5. Trend for centralized and decentralized approach for problem category 5 across due date

Table 7. Time variation across problem categories

Approach	Problem Category 1	Problem Category 2	Problem Category 3	Problem Category 4	Problem Category 5
DA_{20} to DA_{100}	0.02-0.32	0.09-0.62	4.93-638.09	-	-
D_{GA20} to DA_{GA100}	0.47-11.77	0.9-22.36	0.55-64.44	1.29-740.33	4.18-8266.62
DA_{GAGA20} to $DA_{GAGA100}$	8.81-9.66	17.7-19.42	10.43-11.27	23.82-26.43	78.33-84.57

Time calculations:
Previous observations show a clear difference in the performance of decentralized and centralized approaches regarding time. Time calculations show how the problem size affects the time required for centralized and decentralized approaches. When real-life problems are enormous and rapidly changing, saving time to obtain near-optimal solutions by a decentralized approach could be gigantic, giving a competitive advantage for Industry 4.0 kind of scenario for dynamic decision-making.

f

The following calculations are presented to understand the reason behind this difference. Table 8 describes notations that demonstrate problem size for decentralized and centralized approaches. The calculations below neglect the time required by the job type agent as it is minimal compared to the time required by the machine agent to list permutations of sequences of batches.

Table 8. Notations used for time calculations

n	No. of job types on the machine
BSV	Total number of possible batch size values
BSC	Total number of batch size combinations
TS	Total sequences for all batch size combinations
m	Total number of machines
D_i	An order quantity for j^{th} job type
BS_{ij}	Batch size for i^{th} batch size combination and j^{th} job type
NB_{ij}	The number of batches formed from demand such that each batch is processed without interruption for i^{th} batch size combination and j^{th} job type
N_i	Total number of batches on the machine for i^{th} batch size combination Where i=1, 2…….BSC j=1, 2……n

At level 3 from Figure 2, BSC is given by,

$$BSC = BSV^n \tag{12}$$

For example, for two job types with code JT1 and JT2, the batch sizes under consideration for both are 4,8,12,16 and 20. Hence, the BSV for both job types is 5. After substituting values, BSC would be

$$BSC = (5)^2 = 25$$

$$NB_{ij} = \frac{D_j}{Batch\ size_{ij}} \tag{13}$$

$$N_i = \sum_{j=1}^{n} NB_{ij} \tag{14}$$

If the two job types mentioned above had parts of demand 3 and 4, respectively, then Ni=3+4=7
A sequence is formed as

$$Sequence = 1\ 1\ 1\ 2\ 2\ 2\ 2$$

Total number of sequences for i^{th} batch size combination (Si)

$$S_i = \frac{N_i!}{\prod_{j=1}^{n}(NB_{ij}!)} \tag{15}$$

Substituting N1=7, Parts of $demand_{11}$=3 and Parts of $demand_{12}$=4

$$S_1 = \frac{7!}{3! \times 4!} = 35$$

The number of total sequences (TS) is given by,

f

$$TS = \sum_{i=1}^{BSC} S_i \tag{16}$$

For the example considered before,

$$TS = S_1 + S_2 + \cdots + S_{25}$$

Computation time depends on TS and is of the order:
Decentralized approach – $O(m \times TS \times N_{avg})$
Centralized approach– $O(TS^m \times (m \times N_{avg}))$
N_{avg} is the average number of batches routed through a machine in all possible TS sequences.

Given by, $N_{avg} = \frac{\sum_{i=1}^{BSC} N_i \times S_i}{TS}$ (17)

For example,
If time is K^{th} multiple of U_D for Decentralized approach and U_C for Centralized approach

Where $U_D = m \times TS \times N_{avg}$ (18)

and $U_C = TS^m \times (m \times N_{avg})$ (19)

Let TS =20, m=4, N_{avg}=3 then the total time required would be
Decentralized approach – $4 \times 20 \times 3 \times k = 240k\ units$
Centralized approach – $20^4 \times (4 \times 3) \times k = 1.92 \times 10^6 k\ units$

It shows a significant difference in time and required memory for the Decentralized and Centralized approaches. Also, enumerating all sequences is infeasible once a specific memory limit has been reached. Hence, the bound is put on the total number of generated sequences. This bound is implemented by fixing the number of batch size combinations generated at level 3 and sequences within each batch size combination generated at level 4, as described earlier.

5. CONCLUSION

In this work, a novel decentralized scheduling approach is developed where intelligence is imparted to jobs and machines. The approach successfully incorporates batch decisions in multi-stage job shop scheduling problems. Multi-stage scheduling problems with varying problem sizes are constructed to assess the efficacy of the proposed decentralized approach. These problems are also solved using a conventional centralized approach. The decentralized approach generally gives a good solution quality in less time.

The capabilities of a decentralized approach to reach performance, which is closer to the global optimum, is demonstrated in problem category 1, even though a centralized approach with GA achieves the global optimum in a much longer time. Further, with increasing problem complexity, finding a global optimum with brute force was not feasible. Hence, a compromise must be made between time and solution quality. With the decentralized approach, this compromise was much smaller than the centralized approach. The significance of a decentralized approach can be highlighted with problems of larger problem size, where time consumed by a centralized approach would be unaffordable. Different variations of approach are found to be suitable to obtain optimum performance. i.e., best FDOT in minimum time. Hence, the time required can be optimized if the right amount of information is passed between agents by reducing the excess computation that comes with it while maintaining solution quality. Because of the combinatorial nature of the problem, problem complexity, and the vast number of problem instances to which it can be applied, finding the right amount of information to be passed between agents and the strategy to achieve the objective for each agent is a challenge. This work attempts to identify the right amount of information passed between agents and the strategy to achieve agents' objectives to maximize on-time delivery.

We could assign more suitable objective functions to existing agents based on a shop floor goal—for example, inventory-related objectives to optimize costs. Then, machine learning-based approaches can lead to the identification of the right amount of information to be passed and a suitable objective for agents such that shop floor performance is maximized in minimum time. The agility of the decentralized approach can be tested in more dynamic environments with machine interventions and new job arrivals.

REFERENCES

Cupek, R., Ziebinski, A., Huczala, L., and Erdogan, H. (2016). Agent-Based Manufacturing Execution Systems for Short-Series Production Scheduling. *Computers in Industry*, *82:* 245–258. DOI: https://doi.org/10.1016/j.compind.2016.07.009

He, N., Zhang, D. Z., and Li, Q. (2014). Agent-Based Hierarchical Production Planning and Scheduling in The Make-To-Order Manufacturing System. *International Journal of Production Economics*, *149*, 117–130. DOI: https://doi.org/https://doi.org/10.1016/j.ijpe.2013.08.022

Kaplanoğlu, V. (2014). Multi-Agent-Based Approach for Single Machine Scheduling With Sequence-Dependent Setup Times and Machine Maintenance. *Applied Soft Computing Journal*, *23*, 165–179. DOI: https://doi.org/10.1016/j.asoc.2014.06.020

Klein, M., Löcklin, A., Jazdi, N., and Weyrich, M. (2018). A Negotiation-Based Approach for Agent-Based Production Scheduling. *Procedia Manufacturing*, *17*, 334–341. DOI: https://doi.org/https://doi.org/10.1016/j.promfg.2018.10.054

Li, K., Leung, J. Y.-T., and Cheng, B.-Y. (2014). An Agent-Based Intelligent Algorithm for Uniform Machine Scheduling to Minimize Total Completion Time. *Applied Soft Computing*, *25*, 277–284. DOI: https://doi.org/https://doi.org/10.1016/j.asoc.2014.09.006

Li, M., Li, M., Ding, H., Ling, S., and Huang, G. Q. (2022). Graduation-Inspired Synchronization for Industry 4.0 Planning, Scheduling, and Execution. *Journal of Manufacturing Systems*, *64*(April): 94–106. DOI: https://doi.org/10.1016/j.jmsy.2022.05.017

Liu, Y., Sun, S., Wang, X. V., and Wang, L. (2022). An Iterative Combinatorial Auction Mechanism for Multi-Agent Parallel Machine Scheduling. *International Journal of Production Research*, *60*(1): 361–380. DOI: https://doi.org/10.1080/00207543.2021.1950938

Ma, Y., Li, S., Qiao, F., Lu, X., and Liu, J. (2022). A Data-Driven Scheduling Knowledge Management Method for Smart Shop Floor. *International Journal of Computer Integrated Manufacturing*, *35*(7): 780–793. DOI: https://doi.org/10.1080/0951192x.2022.2025622

Maoudj, A., Bouzouia, B., Hentout, A., Kouider, A., and Toumi, R. (2019). Distributed Multi-Agent Scheduling and Control System for Robotic Flexible Assembly Cells. *Journal of Intelligent Manufacturing*, *30*(4): 1629–1644. DOI: https://doi.org/10.1007/s10845-017-1345-z

Nie, Q., Tang, D., Zhu, H., and Sun, H. (2021). A Multi-Agent and Internet of Things Framework of A Digital Twin for Optimized Manufacturing Control. *International Journal of Computer Integrated Manufacturing*, *35*(10–11): 1205–1226.DOI: https://doi.org/10.1080/0951192x.2021.2004619

Parente, M., Figueira, G., Amorim, P., and Marques, A. (2020). Production Scheduling in The Context of Industry 4.0: Review and Trends. *International Journal of Production Research*, *58*(17): 5401–5431. DOI: https://doi.org/10.1080/00207543.2020.1718794

Rossit, D., and Tohmé, F. (2018). Scheduling Research Contributions to Smart Manufacturing. *Manufacturing Letters*, *15*(November 2018): 111–114. DOI: https://doi.org/10.1016/j.mfglet.2017.12.005

Rossit, Daniel A., and Tohmé, F. (2022).(Data-Driven) Knowledge Representation in Industry 4.0 Scheduling Problems. *International Journal of Computer Integrated Manufacturing*, *35*(10–11): 1172–1187.
DOI: https://doi.org/10.1080/0951192x.2021.2022760

Rossit, Daniel A., Tohmé, F., and Frutos, M. (2019). Production Planning and Scheduling in Cyber-Physical Production Systems: A Review. *International Journal of Computer Integrated Manufacturing*, *32*(4–5): 385–395. DOI: https://doi.org/10.1080/0951192x.2019.1605199

Saeidlou, S., Saadat, M., and Jules, G. D. (2019). Knowledge and Agent-Based System for Decentralized Scheduling in

Manufacturing. *Cogent Engineering*, *6*(1): 1–19. DOI: https://doi.org/10.1080/23311916.2019.1582309

Seitz, M., Gehlhoff, F., Cruz Salazar, L. A., Fay, A., and Vogel-Heuser, B. (2021). Automation Platform Independent Multi-Agent System for Robust Networks of Production Resources in Industry 4.0. *Journal of Intelligent Manufacturing*, *32*(7): 2023–2041. DOI: https://doi.org/10.1007/s10845-021-01759-2

Shi, L., Guo, G., and Song, X. (2021). Multi-Agent Based Dynamic Scheduling Optimization of The Sustainable Hybrid Flow Shop in A Ubiquitous Environment. *International Journal of Production Research*, *59*(2): 576–597. DOI: https://doi.org/10.1080/00207543.2019.1699671

Sun, M., Cai, Z., and Zhao, N. (2022). Design of Intelligent Manufacturing System Based On Digital Twin for Smart Shop Floors. *International Journal of Computer Integrated Manufacturing*, *00*(00): 1–25. DOI: https://doi.org/10.1080/0951192x.2022.2128212

Toptal, A., and Sabuncuoglu, I. (2010). Distributed Scheduling: A Review of Concepts and Applications. *International Journal of Production Research*, *48*(18): 5235–5262.DOI: https://doi.org/10.1080/00207540903121065

Vatankhah Barenji, A., and Hashemipour, M. (2019). A Multi-Agent-Based Approach to Dynamic Scheduling and Control for A Flexible Assembly Line System. *International Journal of Industrial Engineering: Theory, Applications and Practice*, *26*(3 SE-Production Planning and Control). DOI: https://doi.org/10.23055/ijietap.2019.26.3.1951

Wang, X., Hu, X., and Wan, J. (2022). Digital-Twin Based Real-Time Resource Allocation for Hull Parts Picking and Processing. *Journal of Intelligent Manufacturing*. DOI: https://doi.org/10.1007/s10845-022-02065-1

Wang, Z., Hu, H., Gong, J., and Ma, X. (2018). Synchronizing Production Scheduling With Resources Allocation for Precast Components in A Multi-Agent System Environment. *Journal of Manufacturing Systems*, *49*, 131–142.DOI: https://doi.org/https://doi.org/10.1016/j.jmsy.2018.09.004

Weiss-Cohen, M., Mitnovizky, M., and Shpitalni, M. (2017). Manufacturing Systems: Using Agents With Local Intelligence to Maximize Factory Profit. *CIRP Journal of Manufacturing Science and Technology*, *18*, 135–144. DOI: https://doi.org/https://doi.org/10.1016/j.cirpj.2016.11.005

Yan, H.-S., Jiang, N.-Y., and Wang, H.-X. (2022). Adaptive Scheduling of Aero-Engine Assembly Based on Q-Learning in Knowledgeable Manufacture. *International Journal of Industrial Engineering: Theory, Applications and Practice*, *29*(3 SE-Production Planning and Control). DOI: https://doi.org/10.23055/ijietap.2022.29.3.3311

Yang, H., Kumara, S., Bukkapatnam, S. T. S., and Tsung, F. (2019). The Internet of Things for Smart Manufacturing: A Review. *IISE Transactions*, *51*(11): 1190–1216. DOI: https://doi.org/10.1080/24725854.2018.1555383