IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 March 2024, accepted 20 March 2024, date of publication 2 April 2024, date of current version 11 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3384252

RESEARCH ARTICLE

An IoT-Enabled Real-Time Dynamic Scheduler for Flexible Job Shop Scheduling (FJSS) in an Industry 4.0-Based Manufacturing Execution System (MES 4.0)

ANUM TARIQ[1], SHOAB AHMED KHAN[1], WASI HAIDER BUT[1], ASIM JAVAID[2], AND TEHMINA SHEHRYAR[2]

[1]Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering (CEME), National University of Science and Technology (NUST), Islamabad 44000, Pakistan
[2]Department of Software Engineering, Mirpur University of Science and Technology (MUST), Mirpur 10250, Pakistan

Corresponding author: Anum Tariq (anum.tariq@ceme.nust.edu.pk)

ABSTRACT The most challenging aspect identified in this study revolves around effectively managing machine breakdowns to ensure uninterrupted production. This paper presents a real-time dynamic scheduling model that addresses the challenges of the Flexible Job Shop Scheduling Problem (FJSSP) while considering the occurrence of random machine breakdowns. An improved hybrid metaheuristic and rule-based multi-strategy technique has been proposed that regenerates an optimized dynamic schedule when a random machine is interrupted. The proposed technique establishes that the presence of real-time system updates from IoT devices will improve scheduling decisions. The proposed methodology's efficacy is showcased through an extensive computational investigation encompassing 9 benchmark problems and a real-world case study, considering three performance objectives (Robustness, Stability and Compound Effectiveness). The results have been compared with three related techniques from literature. The proposed technique gives better results in most cases and can be adopted in increasing the performance of Manufacturing Execution Systems in an Industry 4.0 setup (MES 4.0).

INDEX TERMS Industry 4.0, digital transformation, manufacturing execution systems (MES), job shop scheduling problem (JSSP), flexible job shop scheduling problem (FJSSP), dynamic job shop scheduling problem (DJSSP), real-time rescheduling, Internet of Things (IoT), genetic algorithm (GA).

I. INTRODUCTION

This work can find its way in the ongoing drive for digital transformation of MES through Industry 4.0. There will be no Industry 4.0 without an effective MES with a dynamic job shop schedule as a hub of connectivity and integration, as MES provides real-time visibility and control on the shop floor. Consequently, the implementation of MES has become crucial as part of the Industry 4.0 [1]. Production Planning and Control (PPC) stands as a fundamental element within the MES and PPC is simply incomplete without an effective Job Shop Scheduling (JSS) module. This work showcases the importance of integrating MES, PPC, JSS, and real-time IoT data to enable efficient and agile decision-making, optimize production processes and costs, optimize resource allocation, enhance overall operational efficiency and achieve the strategic objectives outlined in their production plans.

During requirements elicitation, we identified a key challenge in industry settings: machine breakdowns disrupting pre-planned production schedules, leading to shop floor chaos. Typically, a fixed schedule is established and implemented on the shop floor. However, when a machine breaks down, it is very difficult to adjust the schedule manually due to the numerous combinations and

The associate editor coordinating the review of this manuscript and approving it for publication was Byung-Seo Kim.

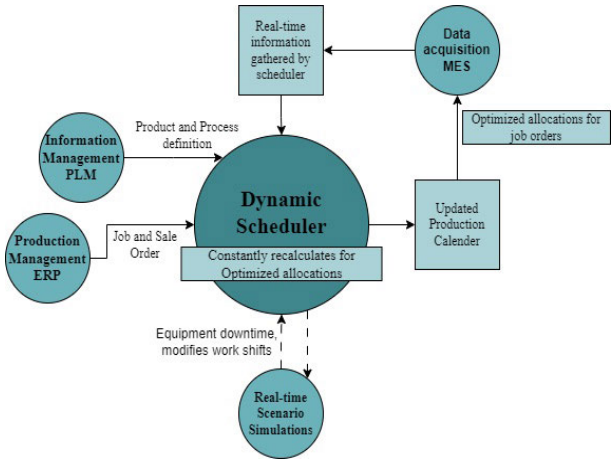

FIGURE 1. Generic dynamic scheduler within MES 4.0 interacting with other components.

permutations of breakdowns that may occur. This pervasive issue, highlighted across various industries, forms the core focus of our paper. In our framework, we address this challenge by ensuring that our algorithm gracefully handles the readjustment of the schedule, regardless of the machine or the number of breakdowns encountered. Our proposed solution utilizes IoT devices to automatically detect and report machine failures, prompting immediate production schedule adjustments. By integrating this functionality into our framework, manual reporting is eliminated, allowing for proactive scheduling adjustments based on predefined machine downtime thresholds. This enhancement is also seamlessly integrated into our rough-cut estimate calculation methodology, ensuring effective production planning despite machine breakdowns.

The classical Job Shop Scheduling Problem (JSSP) pertains to the arrangement of a set of jobs across multiple machines, with each job consisting of multiple operations. Over time, variants of the JSSP have emerged, including the basic JSSP, the Dynamic Job Shop Scheduling Problem (DJSSP), and the Flexible Job Shop Scheduling Problem (FJSSP) [2]. The FJSSP introduces complexity by incorporating machine assignment as an additional decision level, thereby making it an NP-hard problem that necessitates challenging combinatorial optimization [2], [3], [4]. DJSSP involves dynamically adjusting the schedule plan in case of occurrence of a disruptive event. The goal is to identify the most optimal sequence of operations on machines to optimize specific performance indicators. Different objectives can be considered in JSSPs, including minimizing makespan, minimizing tardiness, and maximizing throughput [5]. Over time, rescheduling methods have shifted their focus from traditional performance criteria to emphasize reactivity, adaptability, and robustness. A schedule is deemed stable and robust when there is minimal deviation between the predicted and actual schedules in terms of time or sequence. Recent advancements have introduced specific measures of stability and robustness tailored to the FJSSP [1], [6], [7], [8].

The concept of real-time dynamic scheduling involves utilizing current data to make scheduling decisions [2], [3], [4]. Figure 1 illustrates the architecture and information flow of a generic real-time dynamic scheduler system within an Industry 4.0 based MES (MES 4.0). Overall, the dynamic scheduler acts as a critical component within an Industry 4.0 MES environment, leveraging real-time information, advanced simulation techniques, and integration with Product Lifecycle Management (PLM) and ERP systems. In dynamic manufacturing environments, disruptions and unforeseen events are inevitable, requiring real-time rescheduling utilizing shop-floor information [9], [10]. Dynamic changes can be divided into two groups: job-related factors, including job cancellations, rush jobs, variations in job processing time, changes in due dates or job priority; and resource-related factors, encompassing machine breakdowns, material shortages, and tool unavailability [1], [2]. In the predictive-reactive scheduling strategy for dynamic scheduling, an initial predictive schedule is generated to guide shop floor activities for a specific time period. Rescheduling is then triggered by real-time events to accommodate disruptions and ensure schedule feasibility. Traditional rescheduling approaches, relying on manual adjustments by shop foremen based on experience, often prove to be inefficient and error prone. To address these challenges, computerized reactive schedule repair tools need to be developed, leveraging real-time data for more effective rescheduling [1]. Total rescheduling reschedules all remaining operations with the technique used for generating the initial base schedule with main focus on time optimization in case of any disruption. This disruption affects personnel assignment, resource allocation, raw material delivery, and job processing in other facilities, commonly known as shop floor nervousness [11]. In contrast, repaired schedules with partial rescheduling techniques (i.e. Rescheduling only affected operations) experience a slight decline in quality but retain similarity to the original schedule. Various rescheduling methods, such as Affected Operation Rescheduling (AOR), Modified AOR (mAOR), and Right-Shifting Rescheduling (RSR) are classified as partial rescheduling techniques in the literature [1]. Total rescheduling becomes necessary in cases of frequent and significant disruptions [1], [7], [8], [11]. This research focuses on addressing the FJSSP within the framework of Industry 4.0 manufacturing environments. The aim is to optimize the allocation and sequence of operations on each machine, with the goal of minimizing both the makespan and deviation from the initial schedule. To achieve this, we propose an approach that incorporates IoT-enabled real-time dynamic data-driven production planning and Control. Obtaining stable and robust properties in full rescheduling is a time-consuming process, making it highly impractical and unfavorable in practice. Another focus of our work was to avoid shop floor nervousness. Therefore, we have worked on partial rescheduling (i.e. rescheduling only affected operations). The majority of research approaches are based on the issue of breakdown with a single strategy, making it challenging

to ensure both robustness and stability in performance. Consequently, we have developed a multi-strategy technique to address this limitation.

II. RELATED WORK

A. STATIC FLEXIBLE JOB SHOP SCHEDULING PROBLEM (FJSSP)

The Genetic Algorithm (GA) has been widely utilized as the primary approach for solving the FJSSP, with 54.69% of studies adopting GA [12]. Notable contributions in the literature include studies such as [13] and [14]. Additionally, there have been several significant papers in the field of FJSP that propose novel genetic algorithms. For instance, [15] presents a genetic algorithm that integrates various strategies, outperforming other genetic algorithms and demonstrating comparable performance to the well-known tabu search algorithm. In the work by Liang and Xiao [16], they introduce an enhanced genetic algorithm that specifically targets the minimization of the makespan. This algorithm incorporates a novel initialization method, optimized chromosome encoding, crossover, and mutation operators. Moreover, in the study by Zhang et al. [17], an efficient genetic algorithm for the FJSP is presented. This algorithm is designed to minimize the makespan and incorporates Local Selection (LS) and Global Selection (GS) mechanisms, diverse crossover and mutation strategies, and an enhanced chromosome representation. Finally, [18] introduces an enhanced Genetic Algorithm designed for addressing the Distributed and FJSSP. This algorithm extends the solution representation, employs a greedy decoding procedure, and introduces a local search operator.

B. REAL-TIME DYNAMIC FLEXIBLE JOB SHOP SCHEDULING PROBLEM (DFJSSP)

The literature has extensively explored the optimization of FJSP, particularly emphasizing real-time dynamic rescheduling and accounting for random machine breakdowns.

A certain part of literature also addresses challenges associated with integrating new jobs into existing schedules [19], [20]; however, our work focuses on the scenario where machine breakdowns necessitate schedule adjustments without disrupting the established workflow on the shop floor. One notable literature review by [21] examines 140 related articles and presents an overview of mathematical models, integration frameworks, and qualitative analyses of different approaches in the context of Industry 4.0. Addressing machine breakdown disruptions, [22] explores rerouting policies for affected jobs, aiming to redirect them to alternative machines when the primary machine is unavailable. The authors consider resource costs, processing times, and downstream operation completion time in their decision-making process. Authors [23] consider predictive and reactive scenarios for machine unavailability and aim to achieve robust schedules in the presence of disruptions. Job shop rescheduling outcomes, including rework and reconditioning, are investigated using an event-driven scheduling method with Petri nets [7]. The authors propose the Beam A* Search (BAS) algorithm, which demonstrates favorable performance with high threshold values. Moreover, [24] formulates a mathematical model considering completion time, Lower Bound (LB), and Absolute Deviation (AD) objectives. They optimize the model using a dynamic multi-objective Simulated Annealing algorithm (SSA). A novel graph-based algorithm is presented by [25] to address scheduling-related problems.

To optimize mean tardiness and energy efficiency, [26] proposes a method for the multi-objective DFJSS problem. The Hybrid Rescheduling Strategy (HRS) with an Iterated Genetic Algorithm (IGA) and local search is introduced by [27] for rescheduling in dynamic manufacturing environments. Additionally, [3] and [28] propose a new jaya algorithm for FJJSP with machine breakdown, effectively handling both constrained and unconstrained optimization problems. In this study [29], authors approach the Multi-Objective FJSSP using a decomposition approach to simultaneously minimize makespan, total workload, and critical workload. Furthermore, [2] presents a real-time scheduling (RTS) model that incorporates various rescheduling strategies, policies, and methods to address different uncertainties and leverage real-time information. In the work by Wang et al. [9], a rescheduling decision methodology is introduced specifically for hidden disturbances in job shops utilizing RFID technology. some authors also explored Deep Reinforcement Learning (DRL) based solutions for FJSSP [30], [31], [32], [33]. Using a discrete event simulation model, this paper [34] examines multiple performance measures to evaluate the efficacy of nine dispatching rules in addressing a stochastic DJSSP. Reference [35] addresses the challenge of machine breakdown in workshop production by proposing a mathematical model for a multi-objective DFJSSP.

The following previous studies worked as base for our work:

1) RIGHT SHIFT (RS)

The methodology introduced by [36] is built upon the binary branching algorithm, as originally proposed by Li et al. (1993). Machine and job activities are illustrated using a binary tree structure, where each node corresponds to an operation. Within the binary tree structure, each node's left branch denotes the subsequent operation in the job sequence, guided by technological limitations. The right branch signifies the subsequent operation on the machine, following the operation sequences from the initial schedule. The core principle of the RS is to handle disruptions by delaying the starting times of operations. This delay is kept to a minimum, ensuring that technological constraints are still satisfied while preserving the original sequence of operations.

2) ROUTE CHANGING (RC)

In case of a machine failure, this technique [37] redirect affected jobs to alternative machines to maintain job

processing. Each job operation has a primary machine and multiple alternatives. During a primary machine breakdown, jobs are rerouted to available alternative machines within the existing subset. Careful selection of the alternative machine is crucial for optimal rerouting. Therefore, it uses a cost calculation to choose the alternative machine with the lowest cost. The cost considers processing times, and expected completion time of downstream operations.

3) IMPROVED HYBRID ALGORITHM (IHA)

Rather than relying solely on right-shift adjustments, this [38] approach explores two additional options within the framework of route changes. The objective is to identify the option that results in the most efficient cost reduction, particularly concerning the total makespan of production schedules. A notable improvement in rescheduling strategies concerns the handling of the final operation affected by machine breakdowns. The final operation is strategically placed after the last task on an appropriate machine, ensuring minimal disruption and preserving sequencing integrity. This innovation proves particularly valuable when dealing with uncertain repair times, offering a structured approach to sequence adjustments.

4) SHIFTED GAP REDUCTION (SGR)

In real-world situations, the occurrence of intervals, often termed as machine idle time, between consecutive tasks processed on a machine is not unusual. These gaps can emerge due to precedence constraints or the generation of suboptimal solutions during implementation, as highlighted by Hasan [39], [40]. In tackling the latter scenarios, there exists the potential to improve solutions by strategically incorporating appropriate tasks into these gaps. Within the SGR approach, for a given candidate solution, any gaps between consecutive tasks on a machine are identified. If the gap is large enough to accommodate the operation of a job without violating precedence constraints, a job from the right of the gap can be placed in it. Even when the gap is not extensively large, there remains the possibility of assigning the job within a specified operation time tolerance limit, provided that such placement contributes to the enhancement of the overall performance score. However, in this scenario, a right shift of other jobs is necessary to accommodate the inserted task.

C. RESEARCH GAP

During requirements elicitation, we discovered that the most challenging aspect on the shop floor is the breakdown of machines, especially in complex manufacturing setups where a job must pass through multiple machines for completion. This situation often leads to chaos, yet the current literature fails to adequately address this real-world scenario. Despite the longstanding popularity of DJSSPs, no algorithm can guarantee optimal solutions for all test problems, particularly for larger instances. Therefore, there is a need to analyze the difficulties associated with DFJSSPs and devise improved algorithms capable of effectively solving them. Moreover, existing studies on the DFJSSP reveal certain areas where further research is needed, particularly concerning machine breakdowns:

- A multi-strategy combining RS, RC, and SGR has not been developed and experimented with.
- Most of the researchers performed experimentation on benchmark datasets for industrial processes available online. These datasets are based on hypothetical processes generated with randomization for jobs, operations, machines, and their expected processing times. Real industrial datasets have not been experimented with mostly.
- No one has integrated the DFJSSP techniques with an IoT-enabled Shop Floor Digitization Framework to deal with Dynamic real-time events more effectively.

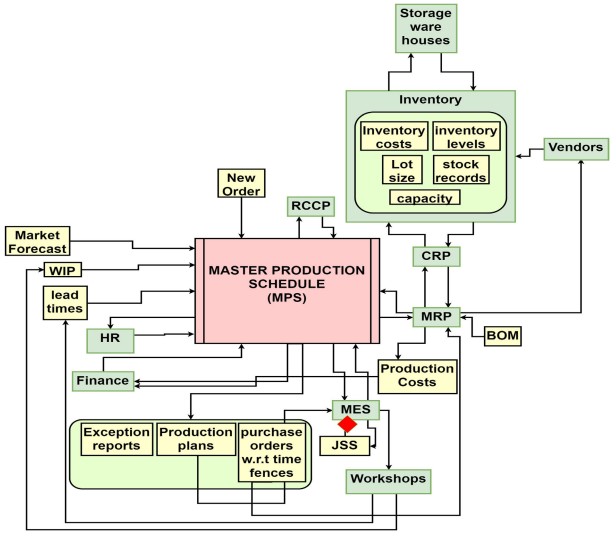

FIGURE 2. Master Production Schedule (MPS) generation process.

III. PROPOSED MODEL FOR THE MASTER PRODUCTION SCHEDULE (MPS)

The MPS serves as a pivotal component in the manufacturing process, offering a high-level production plan that dictates both the ''what'' and ''when'' of product manufacturing to fulfill customer demand, as depicted in Figure 2. MPS relies on a range of critical inputs to effectively plan and coordinate production activities. These include customer demand, which encompasses market forecasts and actual orders, ensuring alignment with market needs. Inventory levels and work in progress (WIP) are essential factors, as they dictate the availability of components and provide visibility into the production pipeline. Lead times are critical for coordinating manufacturing timelines and ensuring materials are on hand when needed.

Moreover, incorporating market forecasts guides proactive planning, while understanding production costs informs

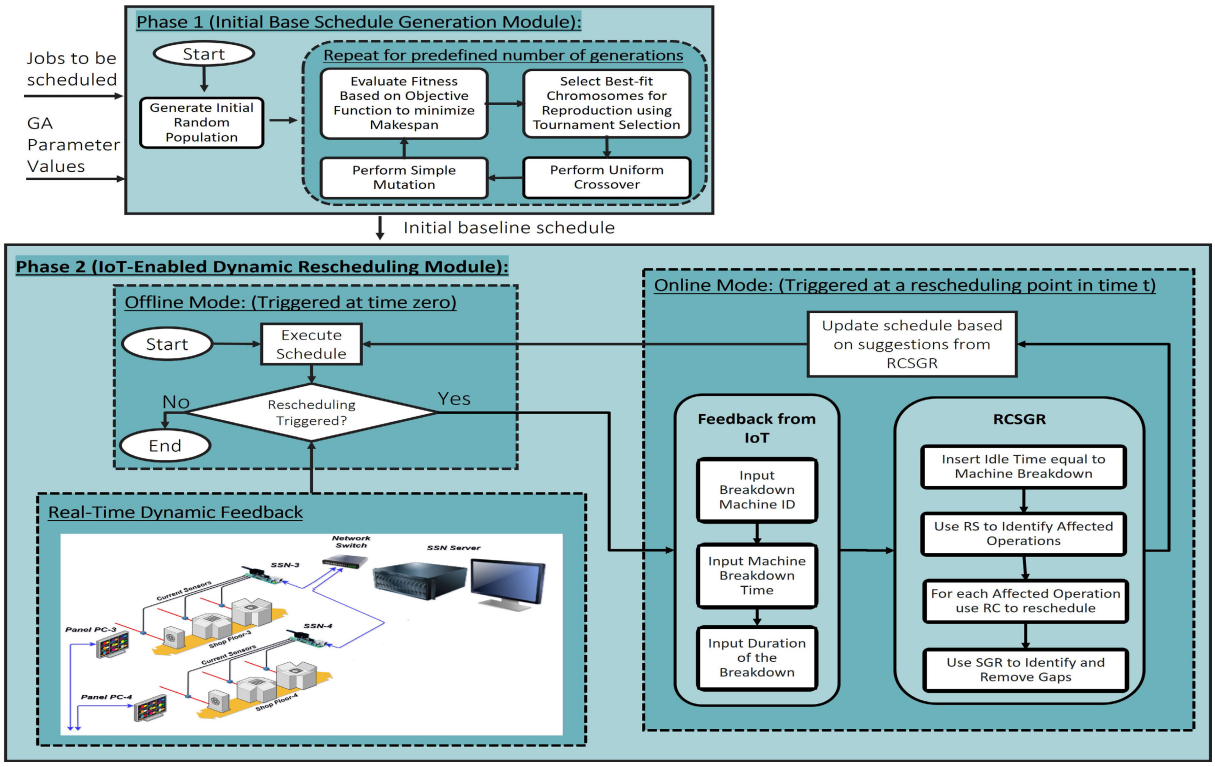

FIGURE 3. Two phase IoT-enabled real-time dynamic scheduling framework for FJSS.

cost-effective strategies. Material Requirements Planning (MRP) focuses on ensuring that the right materials are available at the right time, while Capacity Requirements Planning (CRP) concentrates on whether the production capacity can handle the planned workload.

Bill of Materials (BOM) serve as a blueprint for production, guiding procurement, assembly, and cost estimation processes by specifying the structure and composition of products. Beyond its role in outlining production requirements, such as quantities and deadlines, the MPS plays a crucial function of connecting planning (MPS) and execution (shop floor) levels. Acting as the primary input to the MES, it channels essential production information to bridge this critical gap. The MES, in turn, transforms this data into actionable tasks for the shop floor, generating job shop schedules. Operating at the granular level of workshops and machines, job shop schedules specify when and how individual tasks or jobs are executed, aligning closely with the production requirements defined by the MPS.

Our research delves into job shop scheduling within manufacturing operations but extends beyond individual workshop-level optimization. We aim to enhance the MPS, recognizing its pivotal role in coordinating production. By integrating real-time data, innovative scheduling strategies, and advanced algorithms, our work improves MPS accuracy, adaptability, and responsiveness. Our research demonstrates the inherent collaboration between job shop scheduling and the broader MPS, emphasizing how enhancements at the workshop level impact the larger production planning and control framework. Within MPS, the MES continually monitors production progress, collecting real-time data on performance. This valuable data is then fed back into the MPS, establishing a feedback loop that enables the adjustment of the master production plan based on real-world conditions, such as unexpected machine breakdowns or resource shortages. The JSS is run in two stages. First, it's executed to provide the MPS with an estimate of production time, excluding any considerations for Internet of Things (IoT) data. Then, in the second run, it's executed with smarter machines and IoT integration, allowing for real-time detection of machine breakdowns and resource optimization.

IV. PROPOSED IOT-ENABLED REAL-TIME DFJSS FRAMEWORK

The proposed framework as shown in Figure 3 uses a two-phase methodology that is based on the predictive-reactive strategy of JSSP: during phase 1 an initial schedule is generated with the help of GA as a solution to a basic FJSSP whereas the second phase performs event-driven rescheduling based on IoT feedback regarding machine breakdown given by the Smart Sensing Node (SSN) for Dynamic FJSSP. Phase 2 has two modes of execution including an offline mode and an online mode. The first mode is the offline mode where the SJSS executes as per initial assignments until any disruptive event occurs. As soon as a machine breakdown information is reported by the

TABLE 1. Comparison of Phases 1 and Phase 2.

	Phase 1 Initial Baseline Schedule Generation	Phase 2 Event-Driven Rescheduling Based on IoT Feedback
Inputs	Jobs, Operations, Processing Times, Genetic Algorithm parameter values	Baseline Schedule, Feedback from IoTs
Outputs	Baseline Schedule	Updated optimized Schedule
Objectives	Makespan	Robustness and Stability
Strategy	Predictive-Reactive	Predictive-reactive
Policy	NA	When to reschedule (event-driven), How to reschedule (partial rescheduling)
Method	Genetic Algorithm (GA)	Route Changing Shifted Gap Reduction (RCSGR)

Smart Sensing Network Server (SSNS), the online mode is triggered. During online mode the complete machine breakdown event information is taken from the SSNS and updated schedule is generated by using RCSGR technique. The updated schedule is then considered as baseline schedule for future processing and again assigned to the machines through the SSNS. This cycle keeps on continuing until the completion of every job. A summary of strategies, policies and methods used for both the phases is given in Table 1.

A. PHASE 1: INITIAL BASELINE SCHEDULE GENERATION (PRE-SCHEDULING STRATEGY)

In the context of pre-scheduling, the underlying assumption is that all machines are currently operational, and the likelihood of them encountering breakdowns is assumed to be zero. GA is used to generate an initial schedule based on the input of machines, jobs, operations, start time and end time. Selection, mutation, and crossover are basic GA operators and have obvious effects on the obtained solutions. We have applied various combinations of each of the operator types for experimentation and parameter selection and selected an optimal combination that produces best results in terms of minimum makespan in our ablation study as described in VI-A. The experimentation shows that by using tournament selection, uniform crossover and simple mutation optimal solution is achieved with minimum makespan.

The pre-scheduling strategy used in [37] is based on adding delays in the baseline schedule to accommodate the schedule slips, which is not an optimized solution. We need to generate stable and robust updated schedule of an optimized solution with minimum makespan. Therefore, the pre-scheduling strategy used in our framework does not add any delays in the baseline schedule. Figure 4 is the screen print from our SJSS tool which is based on our proposed GA model. The tool is very flexible and useful for admin at each shop floor as it has multiple views for keeping track of progress for each operation.

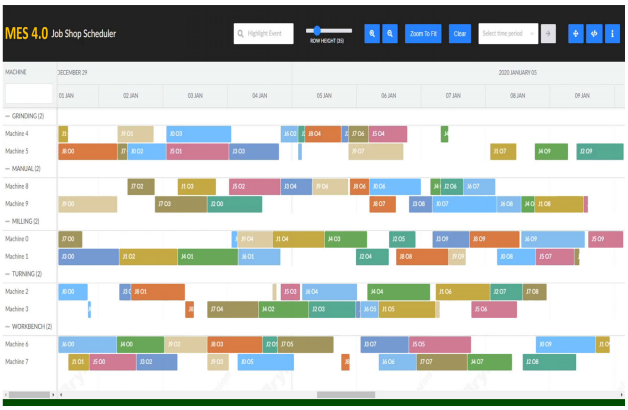

FIGURE 4. Panel PC Interface of SJSS for MES 4.0.

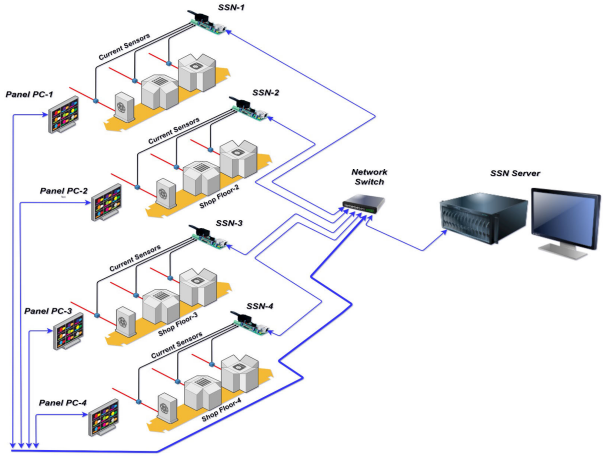

FIGURE 5. IoT-enabled shop floor digitization framework.

B. PROPOSED IOT-ENABLED SHOP FLOOR DIGITIZATION FRAMEWORK

As soon as an initial schedule is executed it is subject to several disruptive events immediately when we consider the actual shop floor scenarios. The main contribution of our work is to answer the question of "How do we get to know that the schedule has slipped?". For this we have proposed a Shop Floor Digitization Framework powered by IoTs shown in Figure 5.

This shop floor digitization framework acts as a smart sensing network deployed in shop floors for real-time monitoring and scheduling. The shop floors consist of multiple machines, each equipped with a current sensor that continuously provides feedback on its operation. This feedback is transmitted to SSN present at each shop floor. Additionally, panel PCs are installed, serving as visual interfaces for displaying schedules and current profiles of the machines. All SSNs and panel PCs are interconnected with a central SSNS, enabling bidirectional communication and data exchange. This scenario demonstrates the implementation of true Industry 4.0 based digitization with machine-to-machine communication.

In this IoT based Digitization setup, an initial schedule is generated and distributed to each machine. The IoT devices

deployed on the machines constantly sense and report their status back to the server. If the server detects a machine malfunction or breakdown, it reschedules the affected operations and reassigns them to alternative machines, ensuring the continuity of the production process.

The integration of IoT technologies presents a significant advancement in enhancing the stability and robustness of the rescheduling process. By leveraging real-time data from IoT devices deployed across the manufacturing environment, our framework can dynamically adapt to changing conditions and disruptions, such as machine breakdowns. This real-time data enables proactive decision-making and allows for timely adjustments to the production schedule, minimizing the impact of disruptions on overall productivity and performance. Once a machine experiences a breakdown, sometimes we have to cause a lot of disturbance to already planned processes while optimizing the makespan. If the more important objective is not to disturb other operations on other machines, then we can give more priority to stability. It will ensure that with minimum disturbance to already planned operations, we can still optimize the makespan. The predictive capabilities of IoT sensors enable early detection of potential issues, allowing for preventive maintenance measures to be implemented before critical failures occur. As a result, the integration of IoT technologies not only improves the responsiveness and agility of the rescheduling process but also directly impacts the improvement of robustness and stability of manufacturing operations.

C. PHASE 2: EVENT-DRIVEN RESCHEDULING BASED ON IOT FEEDBACK

Phase 2 works on a DFJSSP which is subject to the following assumptions:

1) Each machine is capable of processing a single operation during any specific interval.
2) Tasks for each job must adhere to a predetermined order.
3) Each operation is exclusively processed on a single machine.
4) Technological constraints and processing times are predefined and constant.
5) Each machine has adequate capacity to process all assigned operations.
6) The setup time and transport time for any operation are fixed, regardless of the schedule, and are integrated into the respective processing duration.

We have designed a new rescheduling methodology which combines three strategies and responds to real-time dynamic feedback about disruptions from IoT devices. In the context of our approach tasks that are completed prior to a breakdown event are categorized as ''completed'' and do not require consideration during reactive scheduling or rescheduling. Tasks necessitating relocation owing to an interruption are classified as ''affected.'' The determination of affected tasks is based on the inter-task precedence relationships. If the reassignment of a task, prompted by a breakdown, has no effect on its subsequent task, the following task is not classified as affected. This is done with the help of binary tree used in Affected Operation Rescheduling (AOR) [36]. Reactive schedules comprise the affected tasks, which commence from an adjusted starting time specific to each machine. These starting times are computed considering the completion time of the tasks that have been executed. RS combined with RC is applied to reschedule those affected operations based on a defined cost of rescheduling. At the end SGR is applied finally to make the solution more robust and stable.

TABLE 2. Notations for proposed RCSGR technique.

Symbol	Description
k	Machine index (k = 1, 2, 3 . . . , m)
i	Job index (i = 1, 2, 3 . . . , n)
j	Operation index (j = 1, 2, 3 . . . , l)
$O_{i,j}$	Operation sequence for operation j of job i
M_k	Machine Under Breakdown
t^k_{BD}	Breakdown time of machine M_k
$S_{ij,k}$	Start time of O_{ij} on machine M_k
$rt_{j,k}$	jth repair time interval of machine k
$t_{ij,k}$	The processing time for operation j of job i on machine k
R	Set of remaining operations to be rescheduled
AO	Set of potentially affected operations
A	Set of affected operations after rescheduling
$s_{ij,k}$	Start time of O_{ij} on machine M_k in the initial schedule
$e_{ij,k}$	End time of O_{ij} on machine M_k in the initial schedule
$ns_{ij,k}$	Start time of O_{ij} on machine M_k in the new schedule
$ne_{ij,k}$	End time of O_{ij} on machine M_k in the new schedule
no_j	Next operation of job (job branch)
no_m	Next operation of machine (machine branch)
h	Index of set A
g	Index of set AO
$G_{ij,k}$	Gap between O_{ij} and current operation on machine M_k
$P_{ij,k}$	Set of tasks immediately following O_{ij} on machine M_k
k'	Machine of next job operation
$S_{SUC,k}$	Next operation of machine k
$E_{Pre,k}$	Next operation of machine k

1) PROBLEM DESCRIPTION

Two performance measures are considered in our proposed methodology, namely Robustness Measure (RM) and Stability Measure (SM), and these measures are derived from those documented in existing literature [36], [37], [40]. Robustness Measure (RM) basically represents the average difference between pre-scheduling and post-scheduling makespan whereas Stability Measure (SM) is the average deviation of sequence in pre-schedule and post-schedule. It is not possible to obtain high robustness and high stability at the same time in most of the cases therefore we have also analyzed compound effectiveness (Z) which includes both robustness and stability measures.

$$RM = \left(\frac{(MS\_R) - (MS\_P)}{MS\_P}\right) \times 100 \tag{1}$$

$$SM = \frac{\sum_{i=1}^{n'} \sum_{j=1}^{q'} |CO_{ij}^{P} - CO_{ij}^{R}|}{\sum_{i=1}^{n} O_i} \tag{2}$$

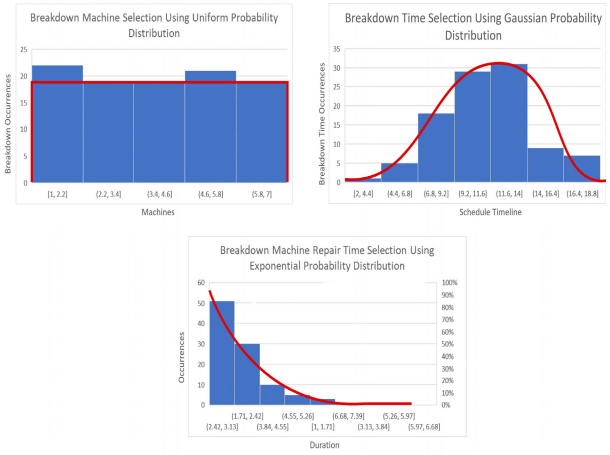

FIGURE 6. Breakdown event generation using different probability distributions.

$$Z = (\gamma * RM) + (1 - \gamma)SM \tag{3}$$

where,

MS_R = Real Makespan

MS_P = pre-scheduling Makespan

$CO_{ij,p}$ = predicted completion time of operation j of job i, in pre-scheduling

$CO_{ij,R}$ = realized completion time of operation j of job i

γ = weightiness that belongs to [0, 1]

The variables involved in proposed RCSGR technique are defined in Table 2.

2) MACHINE BREAKDOWN MODEL

Due to the reason that no benchmark dataset for breakdown machines and the duration of breakdown is available for experimentation, therefore we used slightly changed version of the machine breakdown model proposed by [40] which is based on random probability distributions. A breakdown scenario is denoted as $\Lambda(M_k, t^k_{\mathrm{BD}}, rt_{j,k})$, indicating that machine M_k requires $rt_{j,k}$ units of time to recover at time t^k_{BD}. Each breakdown instance is generated randomly using appropriate probability distributions as shown in Figure 6. The selection of the breakdown machine is determined using a uniform distribution, a Gaussian distribution is used for breakdown time, and an exponential distribution for repair

Algorithm 1 Route Change Shifted Gap Reduction (RCSGR) Pseudocode

Step 1: FindBreakdownInterval($M_k, t^k_{\mathrm{BD}}, rt_{j,k}$)
Step 2: **if** *interruptedOperationExists()* **then**
  AO[1] = getInterruptedOperation()
else
  if *remainingOperationOnMachineExists()* ***AND*** $S_{ij,k} < rt_{j,k}$ **then**
    AO[1] = getFirstRemainingOperation()
  else
    TerminateAlgorithm()
  end if
end if
Step 3: **if** $(t^k_{BD} + rt_{j,k} + t_{ij,k}) \leq min(S_{i(j+1),k'}, S_{SUC,k})$ **then**
  RightShiftOperation(AO[1],$t^k_{BD} + rt_{j,k}$)
else
  AO = determineAffectedOperations(AO[1])
end if
Step 4: InitializeVariables(h,g,AO[],A[])
Step 5: SetCurrentOperation(AO[1])
  SetMachineStartOfCurrentOperation($rt_{j,k}$)
  Increment g
Step 6: alternativeMachines = getAlternativeMachines(O_{ij})
Step 7: **foreach** *machine* M_n *in alternativeMachines* **do**
  ComputeCost(M_n)
  SelectMachineWithMinimumCost()
  if M_n *is the breakdown machine* **then**
    RightShiftOperation()
  else
    if $max(t^k_{BD}, E_{Pre,k'}) + t_{ij,k'} \leq min(S_{i(j+1),k''}, S_{SUC,k'})$ **then**
      InsertOperationWithoutRightShifting(M_n)
    else
      InsertOperationWithRightShifting(M_n)
    end if
  end if
end
Step 8: **if** *currentJobMatchesAffectedOperation()* **then**
  ResetAttributesOfAffectedOperation(A[v])
  Goto Step 11
end if
Step 9: SetAffectedOperation(A[h!])
  Increment h
Step 10: UpdateMakespanAndDeviation()
Step 11: no_j = getNextOperationOfJob()
Step 12: **if** *nojExists() and* $s_{ij,k}$ *of* $no_j < ne_{ij,k}$ *of current operation* **then**
  AO[g] = no_j
  SetJsOfAO[g] = $ne_{ij,k}$ of current operation
  Increment g
end if
Step 13: no_m = getNextOperationOnMachine()
Step 14: **if** *nomExists() and* $s_{ij,k}$ *of* $no_m < ne_{ij,k}$ *of current operation* **then**
  AO[g] = no_m
  SetMsOfAO[g] = $ne_{ij,k}$ of current operation
  Increment g
end if
Step 15: RemoveCurrentOperationFromAOAndAdd NewMembers(no_j, no_m)
Step 16: **if** $AO = \Phi$ **then**
  Goto Step 17
else
  SetCurrentOperationFromAO()
  GotoStep 9
end if
Step 17: *CalculateDifferenceFromPreviousOperation*()

Step 18: if *difference* ≤ 0 **then**
  OperationCannotBeShiftedBack()
else
  intersectingOperations = *getIntersectingOperations*()
  foreach *intersectingOperation in intersectingOperations* **do**
    Gijk = *getGapBetweenOperations*(*Oij*, *intersectingOperation*)
    if *gapSizeIsSufficient*(*Gijk, tijk*) **then**
      ShiftCurrentOperationToGap(*Gijk*)
    else
      if *gapSizeIsInsufficient*($G_{ij,k}$, $t_{ij,k}$) **then**
        if $G_{ij,k} \geq (1-\psi)t_{ij,k}$ **then**
          ShiftCurrentOperationToGap($G_{ij,k}$)
          RightShiftIntersectingOperations($G_{ij,k}$)
        else
          OperationCannotBeShifted()
        end if
      end if
    end if
  end foreach
end if
Step 19: *OutputFinalSchedule*()

TABLE 3. Example process dataset ([24].

Job	Operation	M1	M2	M3	M4	M5	M6
0	0	2	3	4	0	0	0
0	1	0	3	0	2	4	0
0	2	1	4	5	0	0	0
1	0	3	0	5	0	2	0
1	1	4	3	0	0	6	0
1	2	0	0	4	0	7	11
2	0	5	6	0	0	0	0
2	1	0	4	0	3	5	0
2	2	0	0	8	0	9	12
3	0	9	0	7	9	0	0
3	1	0	6	0	4	0	5
3	2	1	0	3	0	0	3

time. These distributions effectively simulate breakdown scenarios that closely resemble real-world situations commonly observed in practical breakdown events.

3) RCSGR ALGORITHM PSEUDOCODE

Initially, the algorithm determines the interval of time during which a breakdown occurs. Subsequently, a binary branching approach is employed to pinpoint affected operations. For each affected operation, a route change algorithm is invoked for rescheduling. Following this, the shifted gap reduction technique is applied to detect and eliminate any existing gaps, further optimizing the solution.

V. DEMONSTRATION OF THE PROPOSED METHODOLOGY: EXAMPLE PROCESS

By showcasing the application of proposed methodology on a specific process, we aim to illustrate the practical implementation and effectiveness of the proposed methodology.

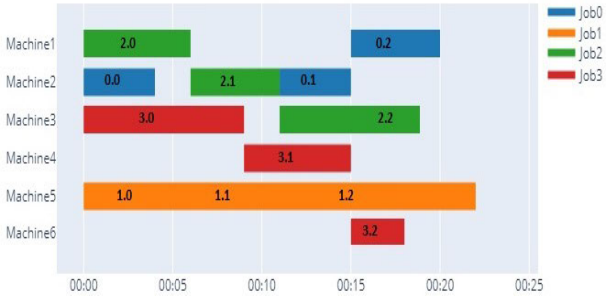

FIGURE 7. Initial baseline schedule generated with GA.

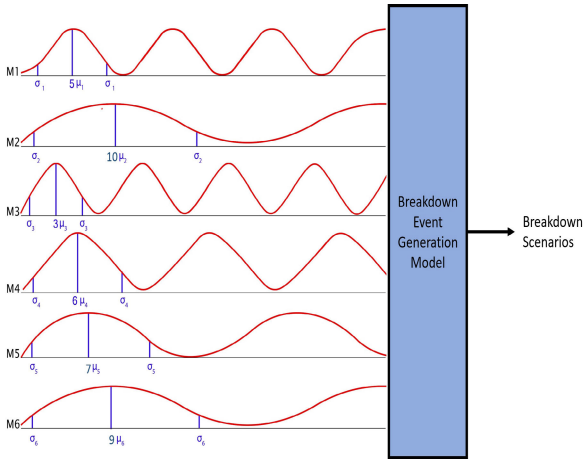

FIGURE 8. Breakdown event generation model.

While larger datasets often provide a broader perspective, we wanted to use the actual industrial dataset that we worked on for the purpose of demonstrating the proposed technique, but that data is quite big. Working with a smaller dataset allows for precise explanations and descriptions of the technique's application steps, enhancing the reader's understanding of its implementation. Therefore, in this demonstration we intentionally opted for a smaller dataset from a previous study conducted by [24]. This data set gives patrial flexibility as shown in Table 3.

As a first step an initial baseline schedule was generated with the help of SJSS tool which is based on GA. The baseline schedule is shown in Figure 7 with a makespan of 22.

In accordance with the methodology detailed in Section IV-C2, we proceeded to generate a set of 100 distinct breakdown event scenarios utilizing our machine breakdown model. Each machine in our study possesses its unique Mean Time To Failure (MTTF) and is characterized by a range of values expressed through Standard Deviation. These statistical parameters are employed to construct Gaussian probability distributions, which, in turn, serve as the basis for simulating random breakdown events within our proposed breakdown model framework. The ensuing visual representation, as depicted in Figure 8, showcases the specific MTTF and standard deviation values assumed for the illustrative example currently under examination.

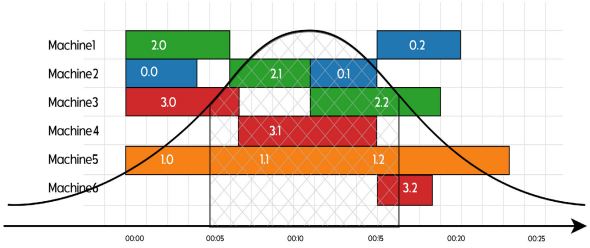

FIGURE 9. Gaussian distribution for the example under discussion.

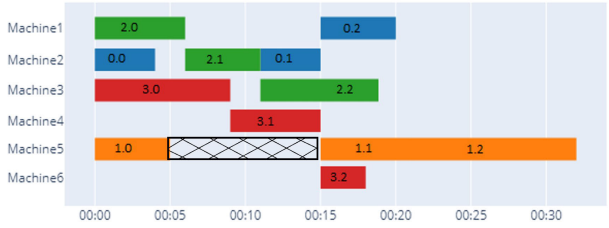

FIGURE 10. Schedule after applying right shift.

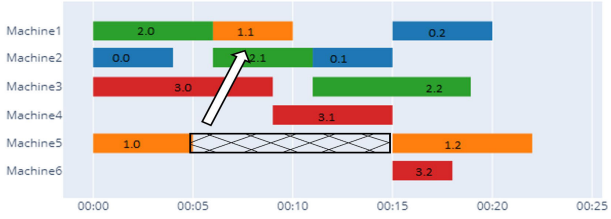

FIGURE 11. Schedule after applying route change.

A Gaussian fit of values generated using the breakdown simulation model for the initial schedule generated from the Pre-scheduling phase is illustrated for this particular example in Figure 9.

We then proceeded to analyze a stochastic breakdown event, specifically focusing on machine 5 as the selected breakdown machine. The breakdown duration was determined to be 10 time units, with an anticipated repair time of 10 time units. Consequently, machine 5 was rendered unavailable for utilization during the interval from time unit 5 to time unit 15. If we simply right shift the operations affected by given machine breakdown scenario the makespan will increase drastically as a result an increase in the robustness, stability and compound effectiveness. Gantt chart is shown in Figure 10 with a makespan of 32.

A better approach is to reroute the affected operations on alternative machines if the alternative machine is free. The choice of machine from a subset of eligible machines is made with the help of a cost function in eq. 3. Figure 11 shows the resulting schedule after applying route changing technique and the makespan is minimized to 22.

Then, we applied the RCSGR by integrating SGR with route changing which further optimizes the schedule and reduced the makespan to 20 as shown in Figure 12.

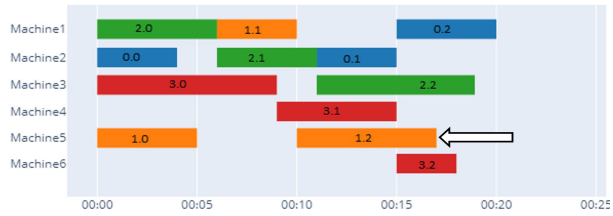

FIGURE 12. Schedule after applying RCSGR.

TABLE 4. Results comparison.

Technique	Makespan	Robustness Measure	Stability Measure	Compound Effectiveness
RSS	32	45.45	1.67	27.94
RC	22	0.00	0.92	0.37
RCSGR	20	0.00	0.58	0.23

Table 4 shows the comparison results after applying RSS, RC and the proposed technique on above example. It can be clearly seen that RCSGR gives the best value of Makespan, Robustness, Stability and Compound Effectiveness. These results demonstrate the practicality and effectiveness of the proposed methodology in enhancing the performance of real-world processes.

VI. ABLATION STUDY

In our research, a comprehensive experimental approach was adopted to enhance job shop scheduling. Firstly, Experiments were conducted to determine the optimal combination of GA parameters employed in the proposed SJSS module. Secondly, we conducted experiments to evaluate our DFJJS module, which incorporates multi-strategy techniques. This module was rigorously compared with other state-of-the-art techniques from the literature.

A. SJSS EXPERIMENTAL SETUP

First of all, we designed the experimentation setup for GA based model. For this we set the Cross Over Rate and Mutation Rate equal to 1. The Population Size was 24. Five number of runs were executed with Generation Size 75000. For conducting this experimentation, we assigned a code to each experiment. These short codes are given in the Table 5.

For example, an experiment was performed with tournament selection, simple crossover and simple mutation, the experiment name will be T_SC_SM. Our model designed for this ablation study produces the result in a CSV file and later the CSV file is used to generate the graph representing makespan for each run and generations up to 75000. Figure 13 shows the result of minimum makespan calculated with each experiment. Experiment no 2 gave the minimum makespan value of 1165 where tournament selection, uniform crossover and simple mutation was used shown in Figure 14.

B. DFJJS MODULE EXPERIMENTAL SETUP

For the DFJSS module we have performed extensive experimentation using 10 examples of different sizes and

TABLE 5. GA parameter types and codes for experimentation.

GA Parameter Type	Code for Experiment	GA Parameter Type	Code for Experiment
Simple Crossover	SC	Tournament Selection	T
Double Point Crossover	DPC	Roulette Wheel Selection	RW
Double Point Crossover2	DPC2	Rank Selection	RS
Uniform Crossover	UC	Rotate Mutate	RM
Order Crossover	OC	Simple Mutate	SM
Discrete Crossover	DC	Single Point Scramble Mutate	SPSM
Random Index Crossover	RIC	Double Point Scramble Mutate	DPSM
Random Index Uniform Crossover	RIUC	Inverse Mutate	IM
Testing Instance: ft20_J20_M5_MS1165,			
File Format: selection_crossover_mutation.csv,			
Example: T_SC_SM.csv			

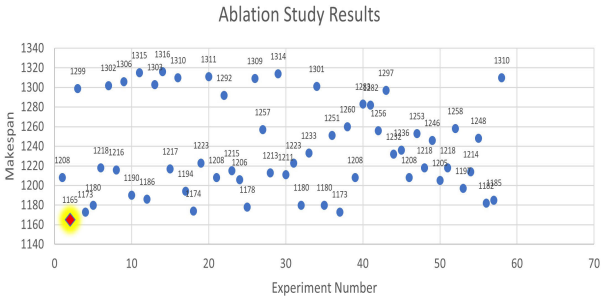

FIGURE 13. Combined results of all experiments.

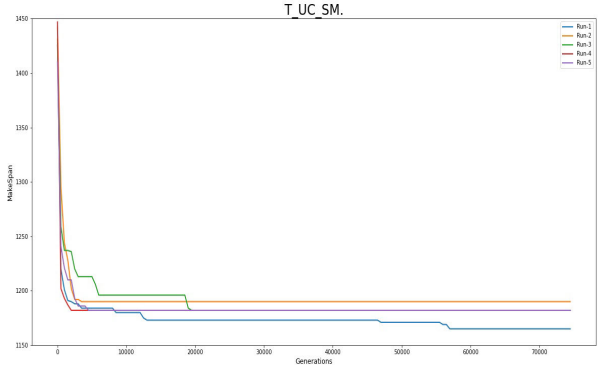

FIGURE 14. Experiment with best makespan value.

dynamics. We have considered the dataset from two categories. Examples of different sizes from FJSSP related other research articles considered as benchmarks for FJSSP and actual dataset from a case study manufacturing organization. The details of data used to perform the experimentation are given in Table 6.

TABLE 6. Dataset dynamics.

Serial Number	Dataset (Reference)	Jobs	Machines	Operations	Flexibility Level
Ex1	[41]	3	3	8	Partial
Ex2	[42]	3	4	8	Total
Ex3	[24]	4	6	12	Partial
Ex4	[43]	10	7	29	Total
Ex5	[36]	6	6	36	Partial
Ex6	[44]	10	20	50	Combination
Ex7	[45] mk1	10	6	54	Partial
Ex8	[45] mk2	10	6	54	Total
Ex9	[45] mk15	30	15	162	Total
Ex10	Case study Dataset	44	48	652	Partial

For each example dataset, preschedules are generated by a GA-based model in phase 1. The size of the population in GA is set to be 8, and the number of generations is 100,000. The crossover probability and mutation rate both were set to be 1.0. We generated 100 different single breakdown scenario events for each example mentioned in the Tabel 6. Thus, there are a total of $10 \times 100 = 1000$ test cases.

The strategies proposed in this paper are compared with RS, RC and IHA proposed by [7], [36], and [37] briefly described in II. The RS approach is built upon pre-scheduling techniques that do not involve the insertion of idle time. On the other hand, the RC and IHA relies on pre-scheduling methods that involve the insertion of idle time. We are focusing on the main objective of gap reduction; therefore, the proposed technique is rooted in pre-scheduling methods that do not involve the insertion of idle time. For calculating the compound effectiveness, γ is set to be 0.6 (value adopted from [37]). The essential requirement for applying SGR can be expressed as $g(\text{time}) \geq (1 - \psi)T(n, m)$, where ψ represents the tolerance limit, ranging from 0 to 1. Therefore, we have used the value 0.2 as the tolerance level declared as most effective value by [40]. All experiments are implemented by Visual Studio 2020 with C# console application and run on an Intel (R) Core (TM) i7-6500U CPU at 2.50 GHz (4 CPUs), 8GB RAM computer with Windows 10.

Table 7 presents data on Average Makespan, Average Robustness Measure (ARM), Average Stability Measure (ASM), and Average Compound Effectiveness for RS, RC, IHA and RCSGR techniques after considering 100 different breakdown scenarios. Lower values in these parameters indicate better performance. Table 8 provides an average improvement percentage of performance parameters for RCSGR when compared to the other three policies, offering a comparative analysis of the obtained results. Negative values indicate improvement, while positive values suggest a degradation in the measured values. A value of zero indicates no change in performance.

The comparison depicted in Table 7 consistently demonstrates that the RCSGR strategy exhibits superior robustness compared to RSS, RC and IHA across all cases, indicating its effectiveness in improving robustness. In the majority

TABLE 7. Average performance measures.

Example	Makespan				Robustness Measure				Stability Measure				Compound Effectiveness			
	RSS	RC	IHA	RCSGR	RSS	RC	IHA	RCSGR	RSS	RC	IHA	RCSGR	RSS	RC	IHA	RCSGR
Example 1	15.44	13.26	11.52	9.32	71.56	47.33	23.10	3.56	2.23	1.29	1.29	1.13	43.83	28.91	11.67	2.58
Example 2	24.87	22.02	21.63	21.49	13.05	1.36	1.36	0.00	0.86	0.22	0.12	0.61	8.17	0.91	0.34	0.24
Example 3	25.83	24.66	24.66	23.95	7.62	3.17	2.21	0.21	0.64	0.23	0.11	0.31	4.83	1.99	1.01	0.25
Example 4	155.11	116.64	115.31	114.17	27.14	2.02	2.02	0.00	3.98	1.50	1.22	1.13	17.87	1.82	1.12	0.45
Example 5	65.95	55.34	55.34	55.01	19.91	0.80	0.71	0.65	3.13	0.55	0.45	0.31	13.20	0.70	0.57	0.52
Example 6	136.25	132.28	132.28	132.21	5.62	2.54	2.54	2.49	0.88	0.59	0.23	0.12	3.72	1.76	1.65	1.54
Example 7	65.06	56.91	56.91	56.69	30.12	13.82	13.64	13.38	4.20	1.59	1.37	0.50	19.75	8.93	8.53	8.23
Example 8	60.73	40.11	40.01	39.92	55.72	2.85	2.42	2.36	3.60	0.49	0.31	0.21	34.87	1.90	1.89	1.50
Example 9	230.23	197.86	197.63	197.4	25.12	7.53	7.31	7.28	5.04	1.77	1.56	0.58	17.09	5.23	4.91	4.60
Example 10	249692.86	249995.46	249552.2	249218.26	2.03	2.15	1.90	1.83	296.93	271.83	193.53	105.66	119.99	110.02	73.47	43.36

TABLE 8. Average improvement measures.

Example	Average Makespan Improvement			Average Robustness Improvement			Average Stability Improvement			Average Compound Effectiveness Improvement		
	RSS vs RCSGR	RC vs RCSGR	IHA vs RCSGR	RSS vs RCSGR	RC vs RCSGR	IHA vs RCSGR	RSS vs RCSGR	RC vs RCSGR	IHA vs RCSGR	RSS vs RCSGR	RC vs RCSGR	IHA vs RCSGR
Example 1	-6.12	-3.94	-2.2	-68.00	-43.78	-19.54	-1.11	-0.16	-0.16	-41.24	-26.33	-9.09
Example 2	-3.38	-0.53	-0.14	-13.05	-1.36	-1.36	-0.25	0.39	0.49	-7.93	-0.66	-0.10
Example 3	-1.88	-0.71	-0.71	-7.42	-2.96	-2.00	-0.33	0.07	0.20	-4.58	-1.75	-0.76
Example 4	-40.94	-2.47	-1.14	-27.14	-2.02	-2.02	-2.85	-0.37	-0.09	-17.42	-1.36	-0.67
Example 5	-10.94	-0.33	-0.33	-19.25	-0.15	-0.06	-2.82	-0.24	-0.14	-12.68	-0.18	-0.05
Example 6	-4.04	-0.07	-0.07	-3.13	-0.05	-0.05	-0.76	-0.46	-0.11	-2.18	-0.22	-0.11
Example 7	-8.37	-0.22	-0.22	-16.74	-0.44	-0.26	-3.69	-1.09	-0.87	-11.52	-0.70	-0.30
Example 8	-20.81	-0.19	-0.09	-53.36	-0.49	-0.06	-3.39	-0.28	-0.10	-33.37	-0.40	-0.39
Example 9	-32.83	-0.46	-0.23	-17.84	-0.25	-0.03	-4.46	-1.19	-0.98	-12.49	-0.62	-0.31
Example 10	-474.6	-777.2	-333.94	-0.19	-0.32	-0.07	-191.28	-166.18	-87.87	-76.63	-66.66	-30.11

of cases, the RCSGR policy also showcases a notable improvement in stability when compared to RSS. However, the enhancement in stability is relatively less pronounced in comparison to RC and IHA. Notably, in Example 2 and Example 3, the stability achieved with the RCSGR policy experiences a degradation when compared to RC and IHA. Specifically, there is a 39% and 7% degradation in stability in Example 2 and Example 3, respectively. This implies that RCSGR does not demonstrate significant improvement over RC and IHA in terms of stability. The reason behind this lies in the fact that in certain cases, robustness and stability cannot be simultaneously optimized. Emphasizing robustness may result in compromised stability, and vice versa. Therefore, the compound effect of both robustness and stability deserves careful consideration in the decision-making process.

VII. CONCLUSION AND FUTURE RECOMMENDATIONS

Our approach adopts a two-stage methodology. During the initial stage, the primary objective of makespan is optimized under deterministic conditions, while in the second stage, the bi-objective function is optimized, taking into account expected machine breakdowns. We employ a predictive-reactive scheduling policy that generates an initial schedule using a Genetic Algorithm (GA), and rescheduling is performed using a multi-strategy technique called RCSGR. This technique integrates the Right Shift (RS), Route Changing (RC), and Shifted Gap Reduction (SGR) methods, providing a trade-off between performance deviation and schedule stability.

The main focus of our work is to mitigate disruptions and ensure accurate and timely rescheduling by integrating an IoT-enabled shop floor digitization framework with the dynamic scheduler which uses real-time delay feedback from Smart Sense Nodes (SSN). Experiments demonstrate that integrating multiple strategies yields better performance than using a single strategy. Given the comparable performance of our approach to RS, RC and IHA, we believe that integrating the SGR policy with route changing and right-shift policies enhances the robustness and stability of rescheduling.

The proposed framework can be expanded to address additional disruptions, including dynamic job arrivals, alterations in due dates, and the inclusion or removal of machines. Additional performance measures can also be explored to evaluate the proposed methodology, such as throughput, resource utilization, or energy efficiency. In future other modules of MES can also be considered as use case and explored for potential research gaps in addition to the production planning and monitoring module.

REFERENCES

[1] G. Zhang, X. Shao, P. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 56, no. 4, pp. 1309–1318, May 2009.

[2] M. Ghaleb, H. Zolfagharinia, and S. Taghipour, "Real-time production scheduling in the Industry-4.0 context: Addressing uncertainties in job arrivals and machine breakdowns," *Comput. Oper. Res.*, vol. 123, Nov. 2020, Art. no. 105031.

[3] K. Z. Gao, M. C. Zhou, and Y. X. Pan, "Jaya algorithm for rescheduling flexible job shop problem with machine recovery," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2019, pp. 3660–3664.

[4] M. A. Awad and H. M. Abd-Elaziz, "A new perspective for solving manufacturing scheduling based problems respecting new data considerations," *Processes*, vol. 9, no. 10, p. 1700, Sep. 2021.

[5] S. M. K. Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic algorithms for solving job-shop scheduling problems," *Memetic Comput.*, vol. 1, no. 1, pp. 69–83, Mar. 2009.

[6] M. Nouiri, A. Bekrar, A. Jemai, D. Trentesaux, A. C. Ammari, and S. Niar, "Two stage particle swarm optimization to solve the flexible job shop predictive scheduling problem considering possible machine breakdowns," *Comput. Ind. Eng.*, vol. 112, pp. 595–606, Oct. 2017.

[7] G. Mejía, C. Montoya, S. Bolívar, and D. A. Rossit, "Job shop rescheduling with rework and reconditioning in Industry 4.0: An event-driven approach," *Int. J. Adv. Manuf. Technol.*, vol. 119, nos. 5–6, pp. 3729–3745, Jan. 2022.

[8] M. Nouiri, A. Jemai, A. C. Ammari, A. Bekrar, and S. Niar, "An effective particle swarm optimization algorithm for flexible job-shop scheduling problem," in *Proc. Int. Conf. Ind. Eng. Syst. Manag. (IESM)*, Oct. 2013, pp. 1–6.
[9] C. Wang and P. Jiang, "Manifold learning based rescheduling decision mechanism for recessive disturbances in RFID-driven job shops," *J. Intell. Manuf.*, vol. 29, no. 7, pp. 1485–1500, Oct. 2018.
[10] L. Atzori, I. A. Iera, and M. Giacomo, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, pp. 2787–2805, May 2010.
[11] V. Subramaniam and A. S. Raheja, "MAOR: A heuristic-based reactive repair mechanism for job shop schedules," *Int. J. Adv. Manuf. Technol.*, vol. 22, nos. 9–10, pp. 669–680, Nov. 2003.
[12] M. K. Amjad, S. I. Butt, R. Kousar, R. Ahmad, M. H. Agha, Z. Faping, N. Anjum, and U. Asgher, "Recent research trends in genetic algorithm based flexible job shop scheduling problems," *Math. Problems Eng.*, vol. 2018, 2018, Art. no. e9270802.
[13] D. Lei, "A genetic algorithm for flexible job shop scheduling with fuzzy processing time," *Int. J. Prod. Res.*, vol. 48, no. 10, pp. 2995–3013, May 2010.
[14] S. Vaghefinezhad and K. Y. Wong, "A genetic algorithm approach for solving a flexible job shop scheduling problem," Jul. 2012, *arXiv:1207.2253*.
[15] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 10, pp. 3202–3212, 2008.
[16] J. Liangxiao and D. Zhongjun, "An improved genetic algorithm for flexible job shop scheduling problem," in *Proc. 2nd Int. Conf. Inf. Sci. Control Eng.*, Apr. 2015, pp. 127–131.
[17] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Exp. Syst. Appl.*, vol. 38, no. 4, pp. 3563–3573, Apr. 2011.
[18] L. De Giovanni and F. Pezzella, "An improved genetic algorithm for the distributed and flexible job-shop scheduling problem," *Eur. J. Oper. Res.*, vol. 200, no. 2, pp. 395–408, Jan. 2010.
[19] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling," *IEEE Trans. Evol. Comput.*, early access, Feb. 14, 2023, doi: 10.1109/TEVC.2023.3244607.
[20] K. Lei, P. Guo, Y. Wang, J. Zhang, X. Meng, and L. Qian, "Large-scale dynamic scheduling for flexible job-shop with random arrivals of new jobs by hierarchical reinforcement learning," *IEEE Trans. Ind. Informat.*, vol. 20, no. 1, pp. 1007–1018, Jan. 2024.
[21] X. Li, X. Guo, H. Tang, R. Wu, L. Wang, S. Pang, Z. Liu, W. Xu, and X. Li, "Survey of integrated flexible job shop scheduling problems," *Comput. Ind. Eng.*, vol. 174, Dec. 2022, Art. no. 108786.
[22] E. Kutanoglu and I. Sabuncuoglu, "Routing-based reactive scheduling policies for machine failures in dynamic job shops," *Int. J. Prod. Res.*, vol. 39, no. 14, pp. 3141–3158, Jan. 2001.
[23] N. H. Al-Hinai, *Optimizing the Flexible Job-Shop Scheduling Problem Using Hybridized Genetic Algorithms*. Manitoba, Canada: Department of Mechanical and Manufacturing Engineering University of Manitoba Winnipeg, 2012, p. 156.
[24] Y. Wang and J. Han, "A FJSSP method based on dynamic multi-objective squirrel search algorithm," *Int. J. Antennas Propag.*, vol. 2021, pp. 1–19, Oct. 2021.
[25] J. Stastny, V. Skorpil, Z. Balogh, and R. Klein, "Job shop scheduling problem optimization by means of graph-based algorithm," *Appl. Sci.*, vol. 11, no. 4, p. 1921, Feb. 2021.
[26] B. Xu, Y. Mei, Y. Wang, Z. Ji, and M. Zhang, "Genetic programming with delayed routing for multiobjective dynamic flexible job shop scheduling," *Evol. Comput.*, vol. 29, no. 1, pp. 75–105, Mar. 2021.
[27] M. Wang, P. Zhang, P. Zheng, J. He, J. Zhang, and J. Bao, "An improved genetic algorithm with local search for dynamic job shop scheduling problem," in *Proc. IEEE 16th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2020, pp. 766–771.
[28] K. Gao, F. Yang, J. Li, H. Sang, and J. Luo, "Improved Jaya algorithm for flexible job shop rescheduling problem," *IEEE Access*, vol. 8, pp. 86915–86922, 2020.
[29] C. Wang, Z. Ji, and Y. Wang, "A novel memetic algorithm based on decomposition for multiobjective flexible job shop scheduling problem," *Math. Problems Eng.*, vol. 2017, pp. 1–20, Jan. 2017.
[30] C. Li, P. Zheng, Y. Yin, B. Wang, and L. Wang, "Deep reinforcement learning in smart manufacturing: A review and prospects," *CIRP J. Manuf. Sci. Technol.*, vol. 40, pp. 75–101, Feb. 2023.
[31] Y. Gui, D. Tang, H. Zhu, Y. Zhang, and Z. Zhang, "Dynamic scheduling for flexible job shop using a deep reinforcement learning approach," *Comput. Ind. Eng.*, vol. 180, Jun. 2023, Art. no. 109255.
[32] S. Yang, J. Wang, L. Xin, and Z. Xu, "Real-time and concurrent optimization of scheduling and reconfiguration for dynamic reconfigurable flow shop using deep reinforcement learning," *CIRP J. Manuf. Sci. Technol.*, vol. 40, pp. 243–252, Feb. 2023.
[33] K. Lei, P. Guo, W. Zhao, Y. Wang, L. Qian, X. Meng, and L. Tang, "A multi-action deep reinforcement learning framework for flexible job-shop scheduling problem," *Expert Syst. Appl.*, vol. 205, Nov. 2022, Art. no. 117796.
[34] P. Sharma and A. Jain, "Performance analysis of dispatching rules in a stochastic dynamic job shop manufacturing system with sequence-dependent setup times: Simulation approach," *CIRP J. Manuf. Sci. Technol.*, vol. 10, pp. 110–119, Aug. 2015.
[35] L. Wei, J. He, Z. Guo, and Z. Hu, "A multi-objective migrating birds optimization algorithm based on game theory for dynamic flexible job shop scheduling problem," *Exp. Syst. Appl.*, vol. 227, Oct. 2023, Art. no. 120268.
[36] R. J. Abumaizar and J. A. Svestka, "Rescheduling job shops under random disruptions," *Int. J. Prod. Res.*, vol. 35, no. 7, pp. 2065–2082, Jul. 1997.
[37] W. He and D.-H. Sun, "Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies," *Int. J. Adv. Manuf. Technol.*, vol. 66, nos. 1–4, pp. 501–514, Apr. 2013.
[38] L. M. Thi, T. T. Mai Anh, and N. Van Hop, "An improved hybrid meta-heuristics and rule-based approach for flexible job-shop scheduling subject to machine breakdowns," *Eng. Optim.*, vol. 55, no. 9, pp. 1535–1555, Sep. 2023.
[39] S. M. K. Hasan, R. Sarker, and D. Essam, "Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns," *Int. J. Prod. Res.*, vol. 49, no. 16, pp. 4999–5015, Aug. 2011.
[40] S. M. K. Hasan, "Evolutionary algorithms for solving job-shop scheduling problems in the presence of process interruptions," Doctoral dissertatio, Austral. Defence Force Acad., School Inf. Technol. Elect. Eng., Univ. New South Wales, 2009.
[41] N. Al-Hinai and T. Y. ElMekkawy, "Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm," *Int. J. Prod. Econ.*, vol. 132, no. 2, pp. 279–291, Aug. 2011.
[42] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Trans. Syst., Man Cybern., C*, vol. 32, no. 1, pp. 1–13, Feb. 2002.
[43] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic," *Math. Comput. Simul.*, vol. 60, nos. 3–5, pp. 245–276, Sep. 2002.
[44] D. Behnke and M. J. Geiger, "Test instances for the flexible job shop scheduling problem with work centers," Dept. Logistics Manag., Helmut-Schmidt-Univ., Hamburg, Germany, Arbeitspapier/Res. Rep. RR-12-01-01, 2012.
[45] P. Brandimarte, "Routing and scheduling in a flexible job shop by Tabu search," *Ann. Oper. Res.*, vol. 41, no. 3, pp. 157–183, Sep. 1993.

ANUM TARIQ received the bachelor's degree in software engineering from The University of Azad Jammu and Kashmir and the master's degree in software engineering from the National University of Science and Technology (NUST), Pakistan, where she is currently pursuing the Ph.D. degree with the College of Electrical and Mechanical Engineering. She is a Lecturer with the Department of Software Engineering, Mirpur University of Science and Technology, Pakistan. Her research interests include digital transformation, Industry 4.0, business process re-engineering and optimization, cloud computing, global software development, software process improvement, requirements engineering, and capability maturity model integration (CMMI).

SHOAB AHMED KHAN received the Ph.D. degree in electrical and computer engineering from Georgia Institute of Technology, Atlanta, GA, USA. He is currently a Professor of computer and software engineering with the College of Electrical and Mechanical Engineering, National University of Sciences and Technology (NUST). He is an inventor of five awarded U.S. patents and has more than 260 international publications. His book on digital design was published by John Wiley and Sons and is being followed in national and international universities. He has more than 22 years of industrial experience in companies in USA and Pakistan. He is the Founder of the Center for Advanced Studies in Engineering (CASE) and the Center for Advanced Research in Engineering (CARE). CASE is a primer engineering institution that runs one of the largest postgraduate engineering programs in the country and has already graduated 50 Ph.D. students and more than 1800 M.S. students in different disciplines in engineering, whereas CARE, under his leadership, has risen to be one of the most profound high technology engineering organizations in Pakistan developing critical technologies worth millions of dollars for organizations in Pakistan. CARE has made history by winning 13 PASHA ICT awards and 11 Asia–Pacific ICT Alliance Silver and Gold Merit Awards while competing with the best products from advanced countries, such as Australia, Singapore, Hong Kong, and Malaysia. He has served as a member of the National Computing Council and the National Curriculum Review Committee. He received the Tamgh-e-Imtiaz Award (civil); the Highest National Civil Award in Pakistan; the National Education Award, in 2001; and the NCR National Excellence Award in Engineering Education. He has also served as the Chairperson of Pakistan Association of Software Houses (PASHA) and a member of the Board of Governance of many entities in the Ministry of IT and Commerce.

WASI HAIDER BUT received the Ph.D. degree in software engineering from the National University of Sciences and Technology (NUST), Pakistan. He is currently a tenured Associate Professor with the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering, NUST. He also led the Automated Software Engineering Research Group. His research interests include software engineering, automated software engineering, requirements engineering, and software quality.

ASIM JAVAID received the B.Sc. degree in software engineering from Foundation University, Pakistan, and the M.S. degree from Bahria University, Pakistan. He is currently pursuing the Ph.D. degree in software engineering with the National University of Computer and Emerging Sciences, Islamabad. He is an accomplished Software Engineer. He is also serving as a Lecturer with the Mirpur University of Science and Technology, Pakistan. His research interests include cyber security and industrial control systems security. As an Intermediate Researcher in this field, he is committed to advancing innovative solutions to tackle evolving cyber threats, reflecting his dedication to fortifying digital landscapes.

TEHMINA SHEHRYAR received the bachelor's degree in computer science from The University of Azad Jammu and Kashmir, Mirpur, Pakistan, in 2006, and the M.S. and Ph.D. degrees in software engineering from Bahria University, Islamabad, Pakistan, in 2009 and 2019, respectively. Previously, she was a Programmer with LMKR, Islamabad. She is currently a Lecturer with Mirpur University of Science and Technology, Mirpur. Her primary research interests include biomedical image analysis and image processing.