International Journal of Industrial Engineering Computations 16 (2025) 423–440

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Scheduling of jobs and autonomous mobile robots: Towards the realization of line-less assembly systems

Tarun Ramesh Gattu[a], Sachin Karadgi[a,b*], Chinmay S. Magi[a], Amit Kore[a], Lloyd Lawrence Noronha[a] and P. S. Hiremath[c]

[a]*KLE Technological University, Department of Automation & Robotics, Hubballi 580 031, India*
[b]*KLE Technological University, Center for Automation & Robotics Research (CARR), Hubballi 580 031, India*
[c]*KLE Technological University, Department of Computer Applications, Hubballi 580 031, India*

CHRONICLE

Article history:
Received October 12 2024
Received in Revised Format December 29 2024
Accepted January 9 2025
Available online January 9 2025

Keywords:
Industry 4.0
Job shop scheduling problem (JSSP)
Conveyor-less assembly
Mass personalization
Autonomous mobile robots (AMRs)
Genetic algorithm

ABSTRACT

As Industry 4.0 continues to transform the manufacturing domain, the focus is shifting towards mass personalization of products, enabling companies to efficiently produce customized goods that meet individual customers' unique needs and preferences. This requires manufacturing enterprises to be flexible and adaptable with their scheduling processes and manufacturing setup. Flexibility and subsequent realization of personalization of products can be realized by utilizing the notion of a Line-less Assembly System (LAS), which replaces a fixed conveyor system with a system in which the products move between machines, with products being fitted on Autonomous Mobile Robots (AMRs) to transport the products from one machine to another as per their production routing. This necessitates scheduling products as per their production routing on available AMRs to reap the benefits of LAS, which is viewed as a Job Shop Scheduling Problem (JSSP) to maximize resource utilization while adhering to constraints. The novelty of this approach is that, in addition to scheduling products, it also considers the scheduling of AMRs. A mathematical formulation to solve the deterministic JSSP is presented in the current work. The formulation is solved for various inputs using a mathematical solver. In general, JSSPs are NP-hard problems. Subsequently, a meta-heuristic-based Genetic Algorithm (GA) has been constructed to solve the JSSP. The solutions obtained through both GA and mathematical solver are compared, and it was found that GA performs well in computation and optimization efficiencies.

1. Introduction

The world in which we live and work has undergone a profound transformation over the past few decades following the IT revolution, with an impact comparable to that following the mechanization and electricity during the first and second industrial revolutions. A trend toward providing an increasing scale of IT infrastructure and services through smart networks has been observed in conjunction with the evolution of PCs (Personal Computers) into smart devices. This trend is leading to a world in which ubiquitous computing has become a reality, in conjunction with the relentless march of the internet and the ever-increasing miniaturization of edge devices. Powerful and autonomous microcomputers are found to be interconnected with each other, which results in the integration of physical systems and virtual systems (cyberspace) into Cyber-Physical Systems (CPS) (Kagermann et al., 2013; Fraser, 2016). In line with Industry 4.0, researchers and practitioners are turning to Smart Manufacturing to optimize materials, resources, and labor while adapting to market fluctuations to meet customer needs (Leiva, 2018; Lu et al., 2016). A smart manufacturing ecosystem is characterized by its quality, agility, productivity, and sustainability capabilities, incorporating functional dimensions (Leiva, 2018) or lifecycle (Lu et al., 2016). In the above-discussed changing scenarios, manufacturing paradigms have dramatically evolved, embracing mass production, mass customization, and mass individualization, as depicted in Fig. 1. Traditional mass production focuses on manufacturing vast amounts of uniform products, while mass customization allows for adjustments to fit specific customer preferences within

* Corresponding author
E-mail sachin.karadgi@kletech.ac.in (S. Karadgi)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2025 Growing Science Ltd.
doi: 10.5267/j.ijiec.2025.1.003

limited restrictions. Mass individualization takes this procedure a step further, enabling the manufacturing of customer-specific products, preferably near customers (Mathews et al., 2023). It has been observed that more advanced manufacturing systems are being designed to enable customer-centered production, i.e., mass individualization (Oh et al., 2022). This entails manufacturing enterprises to be flexible and adaptable in terms of layout and material flow (Mathews et al., 2023; Kaven et al., 2022; Schmidtke et al., 2021; Oh et al., 2022) and, at the same time, be optimal (Buckhorst et al., 2021).

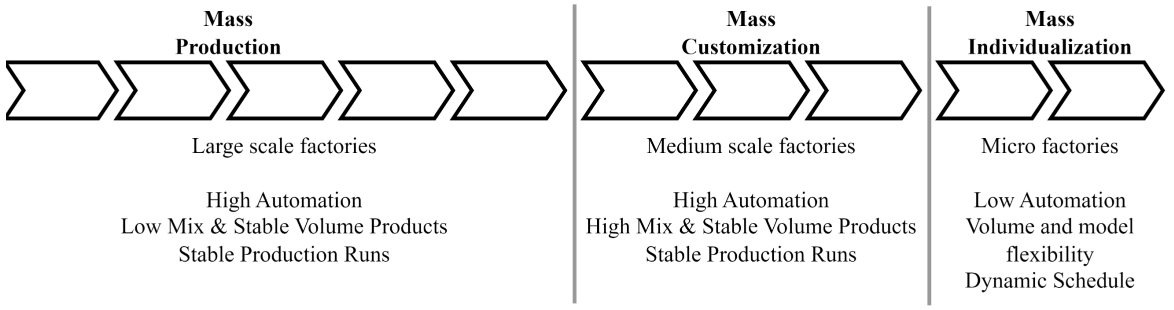


Fig. 1. Comparison of different types of production in a manufacturing paradigm

A novel solution facilitating mass individualization is a Line-less Assembly System (LAS) (Mathews et al., 2023; Kaven et al., 2022), and the more complex one is a Line-less Mobile Assembly System (LMAS) (Buckhorst et al., 2022a,2022b,2022c,2022d; Schmitt et al., 2021). LAS breaks away from typically fixed production lines with conveyor systems, allowing for a more dynamic and adaptable movement of products (or jobs) where Autonomous Mobile Robots (AMRs) carry products to the work centers (Mathews et al., 2023). The realization of LAS involves material transportation as per the production routing, integration of resources, and integrated production planning and control (Mathews et al., 2023), which can be further detailed into four critical stages as shown in Fig. 2: planning, preparation, execution, and cleaning.

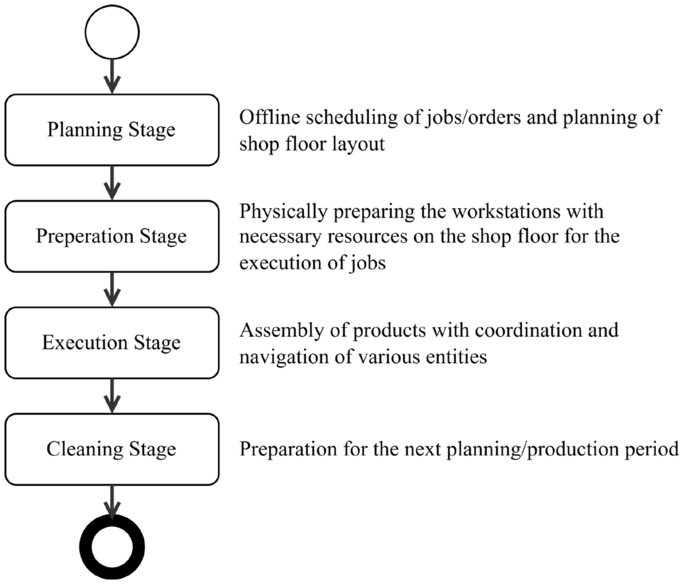

Planning Stage
Offline scheduling of jobs/orders and planning of shop floor layout
Preperation Stage
Physically preparing the workstations with necessary resources on the shop floor for the execution of jobs
Execution Stage
Assembly of products with coordination and navigation of various entities
Cleaning Stage
Preparation for the next planning/production period

Fig. 2. Stages for realization of LAS

In the planning stage, the shop floor layout is planned as per customer orders' (or jobs') requirements during every planning cycle, and corresponding offline job scheduling is critical. The shop floor layout is meticulously designed to support the dynamic nature of the LAS, determining optimal positions for workstations, material storage, pathways, and machines. The preparation stage focuses on physically setting up the shop floor for execution, which includes workstations, tools, and materials to fulfill jobs. During the execution stage, the actual product operations occur, which involves coordinating and navigating various entities, including AMRs and human workers. Finally, the cleaning stage prepares the shop floor for the next planning and production period. This involves resetting and re-configuring workstations, replenishing materials, and performing maintenance on equipment to ensure that everything is ready for the next cycle.

As mentioned above, production planning and scheduling are crucial in realizing LAS. Since products need to be transported by AMRs according to their production routing, the corresponding issue is classified as a Job Shop Scheduling Problem

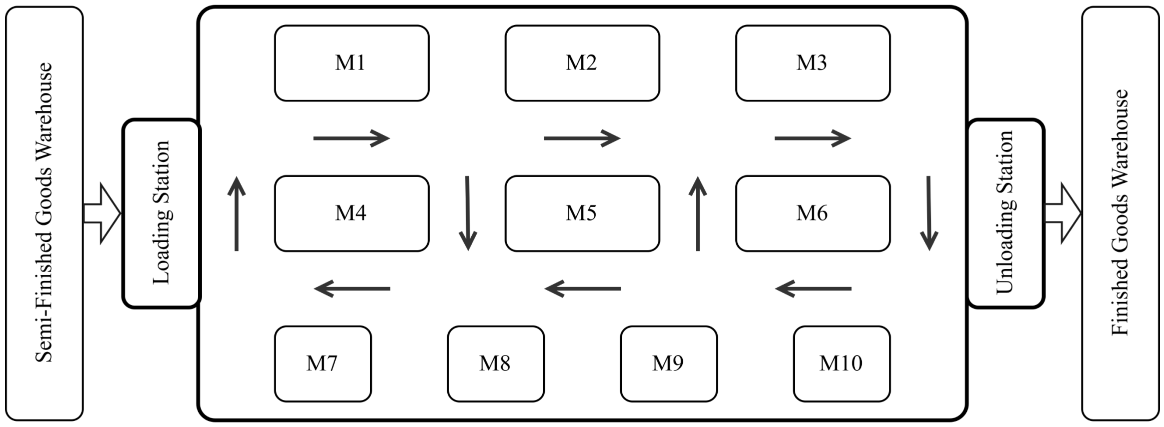
(JSSP). This article concentrates on the planning stage, particularly the offline JSSP, which must incorporate AMRs to maximize resource utilization. Thus, the highlights of this paper are as follows:

- To realize the planning stage of LAS, mathematical formulation as a Mixed Integer Linear Programming (MILP) is elaborated to schedule jobs and AMRs optimally,
- Design a novel meta-heuristic solution based on a Genetic Algorithm (GA) to determine a near-optimal schedule of jobs and AMRs,
- Carrying out computational experiments of proposed mathematical formulation and GA for different data sets and
- Simulating solution outcomes in ROS2 and GAZEBO platform.

Following is the structure of the current work. Section 2 elaborates on the offline JSSP associated with LAS. Section 3 presents a state-of-the-art literature review on the JSSP. Section 4 comprehensively presents MILP formulation and the corresponding GA approach. Section 5 presents data sources and summarizes the computational results of both these approaches by comparing them across various data groups and parameters. Finally, Section 6 discusses the proposed offline scheduling problem of LAS and identifies possible future work.

2. Problem Description

LAS focuses on moving material from one production resource to another as per the production routing without relying on fixed conveyor systems (Buckhorst et al., 2022c). There will be quick adaptation to changes in customer demands, and the assembly systems will be set up as per the incoming customer orders. These assembly systems will support a more comprehensive customization range of products, but the resources will have lower mobility compared to LMAS (Buckhorst et al., 2022a, 2022b, 2022c, 2022d; Schmitt et al., 2021; Hüttemann et al., 2019). However, LMAS makes the resources required to complete an operation available dynamically for a given combination of jobs and operations (Schmitt et al., 2021). Hence, it places stringent requirements on the mobility of resources. Once the operation is completed, the resources are released to form a new coalition to support another job and its operation. This article will refer to the assembly workstations in LAS as machines and products as jobs to mirror the terminologies of scheduling.



Note:
(1) Not to scale; (2) Each machine/station has an IN and OUT location and has not been shown;
(3) 'M*x*' denotes machine *x*; and (4) Arrow indicates the direction of AMR movement.

Fig. 3. Illustration of a possible shop floor

Fig. 3 illustrates a typical shop floor setting, which shows the location of machines, loading and unloading stations, the direction of AMR movement, and so forth. In the future, LAS will be critical in fulfilling customers' wishes as part of mass individualization. Also, as introduced in Section 1, LAS is a broad framework encompassing multiple stages. The current article focuses on the planning stage. Due to individualization, each job has specific requirements as indicated by its production routing, and this type of scheduling problem is known as JSSP (Nouiri et al., 2018; Tavakkoli-Moghaddam et al., 2011; Lin et al., 2010; Zhang et al., 2020; Zhang et al., 2009; Mesghouni et al., 2004; Sha & Lin, 2010). The focus of the JSSP is makespan minimization while adhering to the following constraints and assumptions:

- There exist n jobs that must be processed on maximum m machines.
- Every job has a predefined production routing. This production routing can consist of a sequence of operations, which must be assigned to maximum m machines.
- A job is processed on a given machine only once and will not revisit that machine, i.e., no recirculation.
- There are a_n number of AMRs, which are homogeneous and seamlessly transport the jobs from one location (e.g., machine) to another.

- Due to AMRs, there are loading and unloading stations. Also, an AMR must return to the loading station after unloading to transport the next waiting job as per its production routing.
- A job is not interrupted during its processing at a machine.
- At time *t = 0*, all jobs, machines, and AMRs are available for scheduling.
- Distance between machines, loading and unloading stations are known. Given the velocity of AMRs, it will be possible to determine the time taken to travel from one machine, loading station, or unloading station to another machine, loading station, or unloading station.
- Jobs are independent and can be scheduled in any sequence, i.e., no precedence constraints among jobs.
- Setup, loading, and unloading times are negligible and are not considered.
- Jobs do not have any priorities.
- At any given time, a machine can handle only one job, which can be processed by only one machine. Also, an AMR will move only one job between machines.
- Breakdowns of AMRs or machines are not considered.
- AMR efficiency remains the same throughout the planning horizon.

3. Literature Review

Scheduling is critical for the successful realization of LAS (Mathews et al., 2023), which involves the simultaneous assignment of products/jobs to AMRs for material transportation and further assignment to resources/machines as per the jobs' production routing (Buckhorst et al., 2022b; Tchernev et al., 2013). The scheduling problem elaborated in Section 2 is to assign jobs to machines, as per their production routing (i.e., considered as JSSP), and AMRs to minimize makespan. Research has been conducted on JSSP and more generalized JSSP, i.e., Flexible Job-shop Scheduling Problem (FJSP).

Table 1
Summary of literature review of JSSP and FJSP

Article	Scheduling Problem	Key Research	Objective	Algorithm	Data Instance
Salido et al. (2016)	JSSP	Consideration of energy efficiency	Makespan, Energy consumption	GA	Taillard(1993); Agnetis et al. (2011); Watson et al. (2005)
Lin et al. (2010)	JSSP	Improve scheduling efficiency using the Simulated Annealing technique	Makespan	MPSO	Lawrence (1984); Muth and Thompson (1963)
Liu (2007)	JSSP		Makespan, Adjustment Parameters		
Park et al. (2003)	JSSP	Improvement in quality of initial population	Makespan	Hybrid GA	Muth and Thompson (1963)
Bierwirth (1995)	JSSP	Generalized permutation approach	Makespan	GA	Muth and Thompson (1963); Lawrence (1984)
Falkenauer and Bouffouix (1991)	JSSP	Optimization of schedule	Tardiness	GA	
Liu et al. (2024)	FJSP	Fixture-Pallet combinatorial optimization	Makespan	GA with neighborhood search	
Govi et al. (2021)	FJSP	Sequence within work-centers	Makespan	Two-Stage GA	Barnes and Chambers (1995)
Göppert et al. (2021)	FJSP	Online prediction of the consequence of available actions		ANN	
Ren et al. (2021)	Dynamic FJSP	Considering transportation time and resource constraints	Makespan	GA with Genetic Operations	
Gu et al. (2020)	FJSP	Multi-Objective optimization	Makespan, Bottleneck machine workload, Total machine workload	DPSO-AIW	Flexible Job Shop Problem (2024); Kacem et al. (2002a, 2002b)
Zarrouk et al. (2016)	FJSP	Improvement of CPU time	Makespan	PSO	Zhang et al. (2009)
Jamrus et al. (2018)	FJSP	Fuzzy, Handles uncertain processing time	Makespan	Hybrid PSO and GA	Jia and Hu (2014)
Wu et al. (2018)	FJISP	Multi-objective optimization with decomposition for inverse scheduling	Makespan, Adjustment Parameters	MOEA/D-PSO	Lawrence (1984); Brandimarte Data Set (2024); Agnetis et al. (2011)
Kamble et al. (2015)	FJSP	Multi-objective optimization with rescheduling strategy	Makespan, Machine workload, Total workload, Total idle time, Tardiness	Hybrid MOPSO and SA	Brandimarte Data Set (2024)
Zhang et al. (2009)	FJSP	Multi-Objective optimization	Makespan, max working time, total workload	Hybrid PSO	Kacem et al. (2002a)
Pezzella et al. (2008)	FJSP	Demonstrates effectiveness of GA for FJSP	Makespan	GA	Brandimarte Data Set (2024)
Yamada and Nakano (2000)	FJSP	Guide to GA for FJSP		GA	Muth and Thompson (1963)

Jobs	Production Routing	Processing Times
1	1 (O_{11}) → 3 (O_{21}) → 2 (O_{31})	p_{11}=5; p_{31}=6; p_{21}=3;
2	4 (O_{12}) → 1 (O_{22}) → 2 (O_{32}) → 3(O_{42})	p_{42}=8; p_{12}=8; p_{22}=3; p_{32}=6;
3	2 (O_{13}) → 3 (O_{23}) → 4 (O_{33})	p_{23}=2; p_{33}=3; p_{43}=1;

Number of machines, m = 4
Number of jobs, n = 3
Number of operations = 10
Number of AMRs, a_n = 2

p_{ij}: Processing time of job j on machine i
O_{ij}: i^{th} operation of job j

	Loading	1	2	3	4	Unloading
Loading	0	6	6	11	11	10
1	6	0	5	10	10	9
2	6	5	0	10	10	9
3	11	10	10	0	5	6
4	11	10	10	5	0	6
Unloading	10	9	9	6	6	0

Distance Matrix

Fig. 4. Sample JSSP data, partially adapted from (Pinedo, 2016)

Table 1 summarizes some of these works in scheduling problem domains. The researchers provide basic mathematical formulation. The JSSP and FJSP are NP-hard problems and require utilizing meta-heuristic algorithms to solve these problems (El Ashhab et al., 2017; Yan et al., 2018; Martin and Shmoys, 1996). Researchers employ Particle Swarm Optimization (PSO) or Genetic Algorithm (GA). In some cases, there might be a combination of multiple meta-heuristic approaches, e.g., Discrete PSO with Adaptive Inertia Weight (DPSO-AIW) (Gu et al., 2020) and Multi-Objective Evolutionary Algorithm based on decomposition and PSO (MOEA/D-PSO) (Wu et al., 2018).

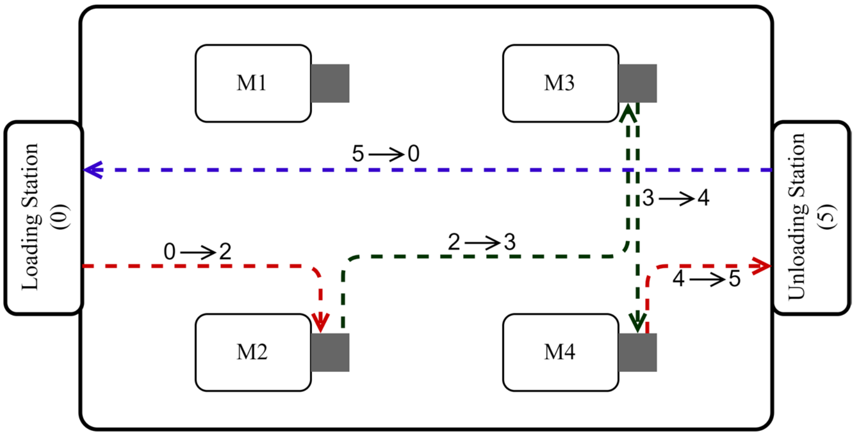
Numerous meta-heuristic algorithms are employed to address JSSP and obtain near-optimal solutions. Evolutionary algorithms (e.g., GA (Falkenauer and Bouffouix, 1991) and PSO (Liu, 2007)) are very commonly used (Mesghouni et al., 2004). GA is often used to solve many scheduling problems of higher complexity and different job shop environments, such as FJSP. For example, the GA approach has been utilized to address the FJSP, thoroughly detailing the solution representation and various GA operations (Pezzella et al., 2008). Binary representation, permutation representation, and genetic enumeration methods are introduced to represent the solution (Bierwirth, 1995). JSSP has been solved using a random critical representation of the solution, a permutation representation, implementing a hybrid swarm intelligence algorithm using PSO, and simulated annealing (Lin et al., 2010). Significant research has been conducted to tackle the JSSP and FJSP. However, significantly less research work is reported by factors other than machines. For instance, researchers have focused on the FJSP, considering transportation time and resource constraints (Ren et al., 2021). A novel algorithm is proposed for the FJSP with a tri-resource constraint (Liu et al., 2024). This enhanced version of the genetic algorithm is hybridized with a feasibility correction strategy and a self-learning variable neighborhood search. Also, a three-string approach is proposed to store the various combinations of the fixtures and buffering of setups between systems or machines (Liu et al., 2018). Handling multiple constraints will become crucial for scheduling. The on-time delivery of products is crucial when realizing LAS (Andrade et al., 2020). Researchers have provided valuable insights for handling a limited number of AMRs (Liu et al., 2024; Ren et al., 2021). Research has not been addressed to include the availability of a limited number of AMRs for job transportation and its corresponding constraints. For instance, the AMR must unload the job at the unloading station and then travel to the loading station. This article focuses on optimizing the job schedule while considering the limited number of AMRs for job transportation and the constraints it imposes, like the travel time of AMR between different machines.

4. Methodology

A methodology is presented in the following paragraphs. First, a mathematical formulation is presented based on JSSP, and the formulation is extended to include the AMR constraints described in Section 2. Second, a GA is elaborated as the mathematical formulation is NP-hard to solve.

4.1 Mathematical Formulation Approach

The scheduling problem elaborated in Section 2 can be represented as $J_m \mid\mid C_{max}$, with a limited number of AMRs and other constraints. Fig. 4 presents a sample job shop data with *4* machines and *3* jobs. Processing time, distance among machines, and loading and unloading stations are also given. Also, Fig. 5 illustrates the movement of job *3* as per the production routing mentioned in Fig. 4, including the movement of AMR from the loading station *0* to the machine *2* of the first operation (i.e., 0→2), the machine *4* of the last operation to the unloading station *5* (i.e., 4→5), and the unloading station *5* to the loading station *0* (i.e., 5→0).



Note:
Movement of Job *3*, including moving to Machine 2 from loading station, and AMR moving from unloading station to loading station

Fig. 5. Movement of AMR with Job 3 for the sample data given in Fig. 4, starting from the loading station to various machines as per its production routing, unloading at the unloading station, and finally, the AMR travels back to the loading station

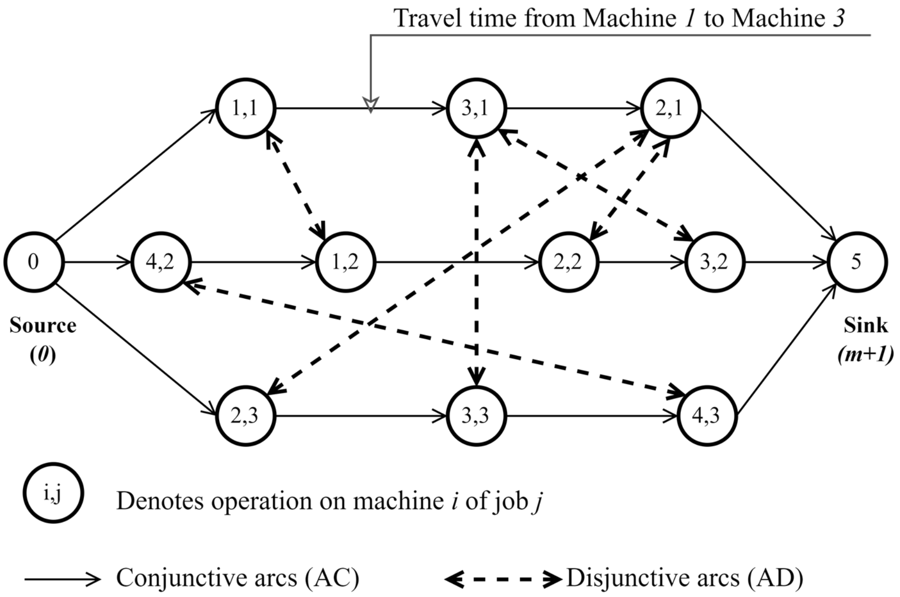


Fig. 6. Directed graph representing the production routing (or conjunctive arcs) and machine dependencies among jobs (or disjunctive arcs)

In general, two dummy machines, 0 and m+1, are introduced that can be considered as loading and unloading stations, respectively. Here, each job starts with processing on machine *0* with $p_{0j} = 0$ and ends with processing on machine $m+1$ with $p_{(m+1)j} = 0$. Additionally, each job's production routing, i.e., operations precedence constraints, must be strictly adhered to and represented as conjunctive arcs (AC) (Pinedo, 2016). Likewise, a machine might process one or more jobs represented as disjunctive arcs (*AD*) (Pinedo, 2016). These conjunctive and disjunctive arcs are depicted in Fig. 6.

The MILP formulation employs many decision variables, parameters, and indices, which are listed as follows.

- Indices

i, l	–	Machine index
j, k	–	Job index
a	–	AMR index

- Parameters

m	–	Number of available machines
n	–	Number of available jobs
a_n	–	Number of available AMRs

M – Machine set, $M = \{i \mid i = 0, 1, 2, \dots m, m+1\}$
J_N – Job set, $J_N = \{j \mid j = 1, 2, \dots n\}$
A – AMR set, $A = \{a \mid a = 1, 2, \dots a_n\}$
AJ – Set of all possible combinations of machine and jobs (i, j)
AC – Conjunctive arcs, $(i, j) \rightarrow (l, j)$
AD – Disjunctive arcs, $(i, j) \rightarrow (i, k)$
p_{ij} – Processing time of job j on machine i
D_{il} – Distance between machine i and machine l
V_a – Velocity of AMR a

- Continuous Variables

C_{ij} – Completion time of job j on machine i
C_{max} – Makespan

- Decision Variables

$$x_{ijk} = \begin{cases} 1, \text{ if job } j \text{ precedes job } k\ (j \prec k) \text{ on machine } i \\ 0, \text{ otherwise} \end{cases}$$

$$\delta_{aj} = \begin{cases} 1, \text{ if AMR } a \text{ is assigned to job } j \\ 0, \text{ otherwise} \end{cases}$$

$$\gamma_{jk} = \begin{cases} 1, \text{ if job } j \text{ precedes job } k\ (j \prec k) \text{ directly or indirectly for AMR assignment} \\ 0, \text{ otherwise} \end{cases}$$

Given the above notations, the MILP formulation is proposed as follows:

$$\text{Minimize } C_{max} \tag{1}$$

subjected to

$$H(1 - x_{ijk}) + C_{ik} \geq C_{ij} + p_{ik}, \qquad \forall (i,j) \rightarrow (i,k) \in AD \tag{2}$$

$$H \cdot x_{ikj} + C_{ij} \geq C_{ik} + p_{ij}, \qquad \forall (i,j) \rightarrow (i,k) \in AD \tag{3}$$

$$C_{lj} \geq C_{ij} + p_{lj} + \sum_{a \in A} (\delta_{aj} \cdot D_{0i} \div V_a) \qquad \forall (i,j) \rightarrow (l,j) \in AC \tag{4}$$

$$H(1 - \delta_{aj}) + H(1 - \delta_{ak}) + H(1 - \gamma_{jk}) + C_{ik} \geq C_{lj} + p_{ik} + \textstyle\sum_{a \in A}(\delta_{ak} \cdot D_{(m+1)0} \div V_a) + \textstyle\sum_{a \in A}(\delta_{aj} \cdot D_{0i} \div V_a), \qquad \forall j,k \in J_N, j \neq k, \; i = \text{first operation of job } k, \; l = \text{last operation of job } j \tag{5}$$

$$H(1 - \delta_{aj}) + H(1 - \delta_{ak}) + H \cdot \gamma_{jk} + C_{lj} \geq C_{ik} + p_{lj} + \textstyle\sum_{a \in A}(\delta_{aj} \cdot D_{(m+1)0} \div V_a) + \textstyle\sum_{a \in A}(\delta_{aj} \cdot D_{0i} \div V_a), \qquad \forall j,k \in J_N, j \neq k, \; i = \text{first operation of job } j, \; l = \text{last operation of job } k \tag{6}$$

$$C_{ij} \geq p_{ij} + \textstyle\sum_{a \in A}(\delta_{aj} \cdot D_{oi} \div V_a), \qquad \forall j \in J_N, \; i = \text{first operation of job } j \tag{7}$$

$$C_{(m+1)j} \geq C_{lj} + \textstyle\sum_{a \in A}(\delta_{aj} \cdot D_{l(m+1)} \div V_a), \qquad \forall j \in J_N, \; l = \text{last operation of job } j \tag{8}$$

$$x_{ijk} + x_{ikj} \leq 1, \qquad \forall (i,j) \rightarrow (i,k) \in AD \tag{9}$$

$$\gamma_{jk} + \gamma_{kj} = 1, \qquad \forall j,k \in J_N, j \neq k \tag{10}$$

$$C_{ij} \geq p_{ij}, \qquad \forall i \in M, \; \forall j \in J_N \tag{11}$$

$$C_{max} \geq C_{(m+1)j}, \qquad \forall j \in J_N \tag{12}$$

$$\sum_{a \in A} \delta_{aj} = 1, \qquad \forall j \in J_N \tag{13}$$

$$\sum_{k \in J_N} x_{i0k} = 1, \qquad \forall i \in M \tag{14}$$

$$\gamma_{jj} = 0, \qquad \forall j \in J_N \tag{15}$$

$$x_{ijk} \in \{0,1\}, \qquad \forall i \in M, \; \forall j,k \in J_N \tag{16}$$

$$\delta_{aj} \in \{0,1\}, \qquad \forall a \in A, \; \forall j \in J_N \tag{17}$$

$$\gamma_{jk} \in \{0,1\}, \qquad \forall j,k \in J_N \tag{18}$$

Objective (1) illustrates the makespan C_{max} minimization, i.e., maximizing the utilization of machines. The disjunctive arcs imply that no operations/processing of two jobs start simultaneously on a machine i. This is ensured by Constraints (2) and (3), which form either-or constraints. Constraint (4) assures the operations are sequenced as per the production routing with no two operations starting simultaneously, and also accounts for the time to transport job j on AMR a from the previous machine i to the current machine l. Once job j is assigned to AMR a, the AMR a cannot be assigned to job k until all the operations of job j are completed on AMR a. Either or Constraints (5) and (6) assure that the last operation of job j and the first operation of a different job k do not overlap in case of both jobs j and k are being assigned AMR a. These constraints also include the time AMR takes to travel from operation on the last machine of a job *j* to an unloading station $m + 1$ and from an unloading station $m + 1$ to loading station *0,* where a job *k* is loaded on the AMR *a*.

As part of LAS, job *j* must be transported from loading station *0* to the machine processing the first operation of the job. Likewise, job *j* must be transported from the machine processing the last operation of the job to the unloading station $m + 1$. These requirements are fulfilled by Constraints (7) and (8), respectively. Constraint (9) ensures that job *j* precedes job *k* or job *k* precedes job *j* on machine *i* directly or indirectly. Likewise, Constraint (10) ensures the precedence constraints among jobs *j* and *k* across the planning horizon and directly impacts AMR assignment. Constraint (11) establishes that the C_{ij} is more than the p_{ij}. The makespan C_{max} is computed by Constraint (12) as indicated by completion time at the unloading station $m + 1$, which also includes the travel time of AMR to the unloading station. Constraint (13) ensures that a job *j* is assigned to an AMR *a*. Constraint (14) ensures that a job originates at a loading station and then moves to a machine processing the first operation. Finally, the non-negative constraints are indicated by Constraints (15), (16), (17), and (18).

4.2 Genetic Algorithm Approach

The mathematical formulation elaborated in Section 4.1 is NP-hard to solve. As a choice of meta-heuristics, GA is designed per the steps mentioned in Fig. 7.

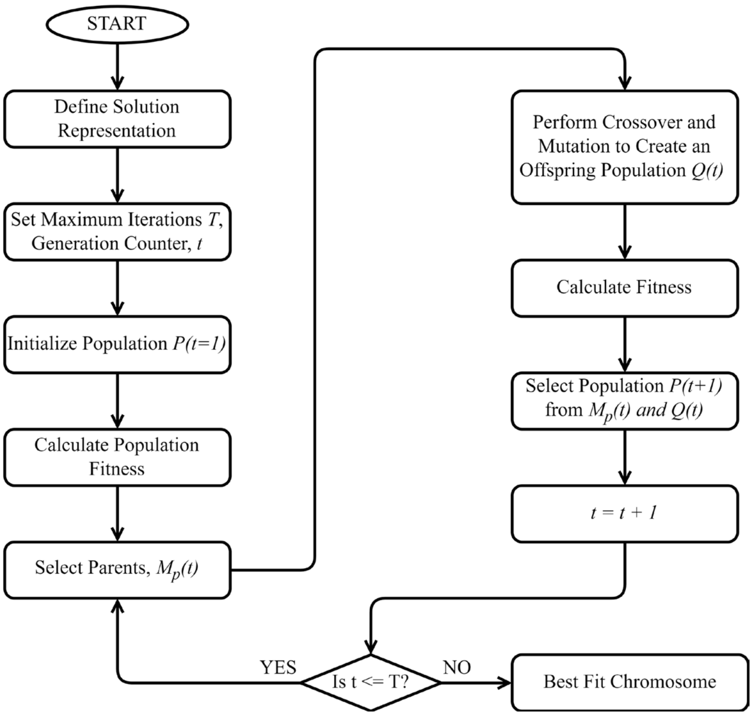
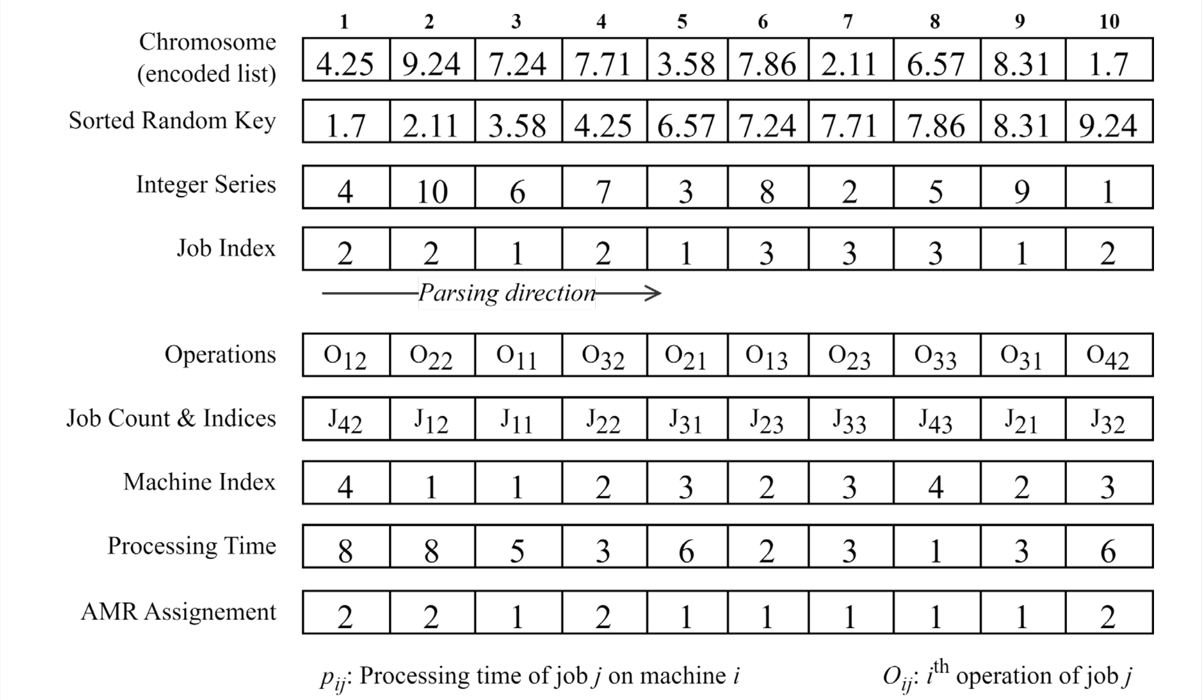

START
Define Solution Representation
Set Maximum Iterations T, Generation Counter, t
Initialize Population P(t=1)
Calculate Population Fitness
Select Parents, M_p(t)
Perform Crossover and Mutation to Create an Offspring Population Q(t)
Calculate Fitness
Select Population P(t+1) from M_p(t) and Q(t)
t = t + 1
YES
Is t <= T?
NO
Best Fit Chromosome

Fig. 7. Pseudocode of Genetic Algorithm (GA)

4.2.1. Solution Representation

The solution representation is crucial as it influences the quality of the GA outcome and the processing complexity of the remaining GA steps. A solution representation, illustrated in Fig. 8, has been adapted from Lin et al. (2010) and extended to include AMR assignments, which is the novel idea introduced in the present work. The solution representation is populated with the data from Fig. 4, and the possible decoding of the solution representation into a schedule is illustrated in Fig. 9. The quality of the fitness value and convergence of the fitness value to near-optimal value depend on randomness. The solution representation relies on random key encoding, wherein a series of strings denote the individual operations (Lin et al., 2010). The solution representation consists of a list of elements with a length equal to the total number of operations, and each element corresponds to an operation. The value of each element is composed of whole numbers and fractional values, where a whole number is randomly generated between 1 and the total number of operations and a fractional value between 0 and 0.99. These random numbers are sorted in ascending order, denoted by a sorted random key. At the same time, an integer number identifying the location of the number before sorting is noted as an integer series. This integer series undergoes permutation using the formula *operation = (number mod n) + 1* to determine the job index. When parsed from left to right, each occurrence of a job number in the job index list is mapped onto an operation of that job. The solution representation is extended to include the random assignment of AMRs to jobs, which is the novelty of this work. The travel time constraint requires accurate knowledge of the shop floor layout to construct a distance matrix containing distances between machines and other locations, as well as the speed (or velocity) of the AMRs, which influences the determination of C_{max}.



	1	2	3	4	5	6	7	8	9	10
Chromosome (encoded list)	4.25	9.24	7.24	7.71	3.58	7.86	2.11	6.57	8.31	1.7
Sorted Random Key	1.7	2.11	3.58	4.25	6.57	7.24	7.71	7.86	8.31	9.24
Integer Series	4	10	6	7	3	8	2	5	9	1
Job Index	2	2	1	2	1	3	3	3	1	2
Operations	O_{12}	O_{22}	O_{11}	O_{32}	O_{21}	O_{13}	O_{23}	O_{33}	O_{31}	O_{42}
Job Count & Indices	J_{42}	J_{12}	J_{11}	J_{22}	J_{31}	J_{23}	J_{33}	J_{43}	J_{21}	J_{32}
Machine Index	4	1	1	2	3	2	3	4	2	3
Processing Time	8	8	5	3	6	2	3	1	3	6
AMR Assignement	2	2	1	2	1	1	1	1	1	2

p_{ij}: Processing time of job j on machine i O_{ij}: i^{th} operation of job j

Fig. 8. Solution representation for the problem data defined in Fig. 4

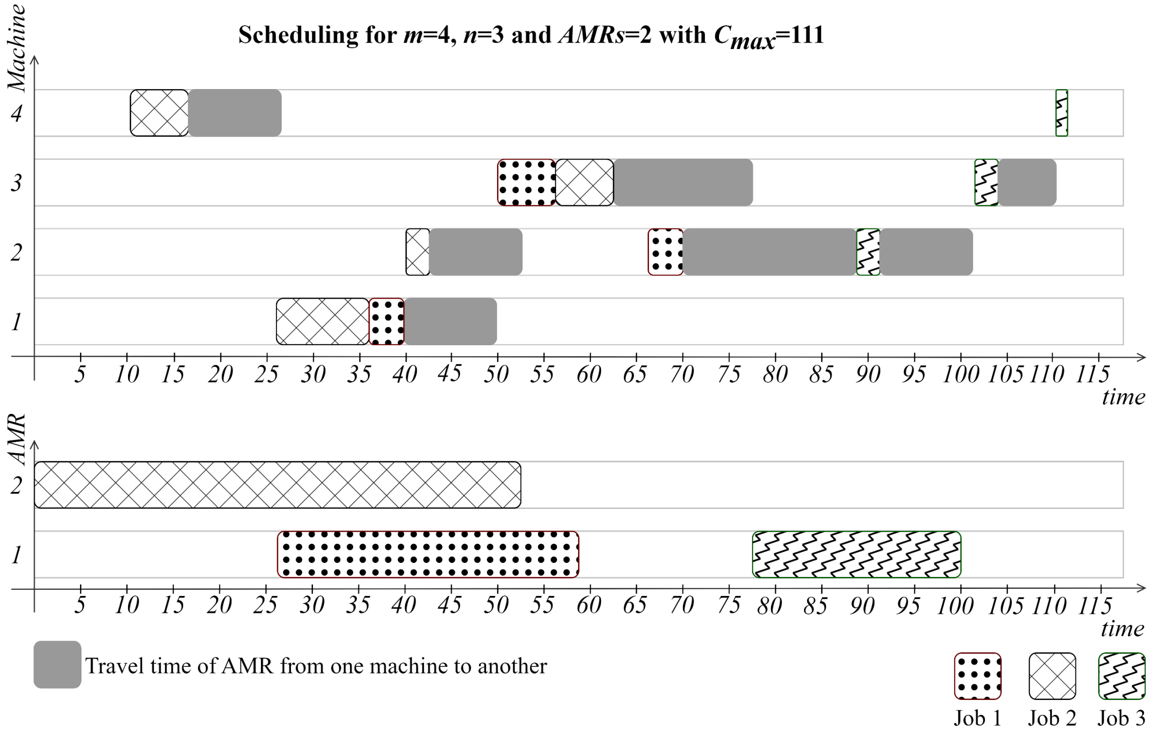


Fig. 9. Schedule represented as a Gantt chart after manually decoding the solution representation illustrated in Fig. 8

4.2.2. Generating Initial Population

The population *P(t=1)* contains *N′* solutions, and the fitness of each solution is computed according to the objectives and constraints. The initial solutions can be generated using multiple ways, including random generation and heuristic methods.

Some heuristic methods include the Shifting Bottleneck method (Pinedo, 2016), shortest processing time first method (Falkenauer & Bouffouix, 1991), least slack time first method (Falkenauer and Bouffouix, 1991), global minimum and random assignment rules (Pezzella et al., 2008) and many more (Wu et al., 2018; Mesghouni et al., 2004). In the current work, the initial population generation has been carried out by generating the population with individual solutions from *3* different heuristic methods and the remaining solutions by random number generation. The heuristics used in this implementation are Shortest Remaining Time (SRT) (Akhtar et al., 2015; Satapathy et al., 2017), Shortest Processing Time (SPT) (Falkenauer & Bouffouix, 1991; Pinedo, 2016; Mesghouni et al., 2004), and Longest Processing Time (LPT) (Pinedo, 2016; Mesghouni et al., 2004).

4.2.3. Parent Selection Criteria

Researchers have proposed numerous methods for selecting the parents from the population for the mating pool $M_p(t)$ who are fit for generating offspring, such as fitness proportional, ranking, roulette wheel, tournament, and uniform parent selections, among others (Eiben & Smith, 2015; Pinedo, 2016; Pezzella et al., 2008). The two-way tournament selection method is used in the present work as it is an overall balanced method for maintaining selection pressure (Pinedo, 2016; Pezzella et al., 2008).

4.2.4. Crossover and Mutation Operations

GA operations, i.e., crossover and mutation, are performed on the previously selected parent solutions to generate variations leading to a (near) optimal fitness value.

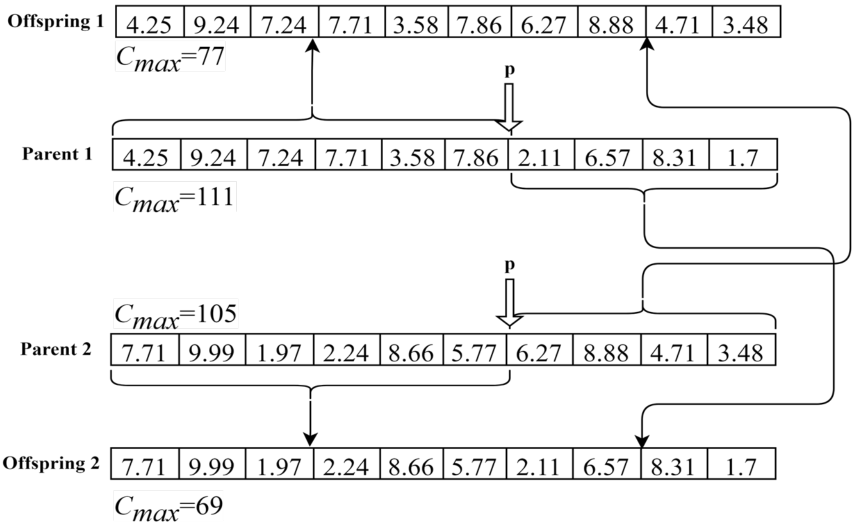


Fig. 10. Single-point crossover operation, other details of the solution not shown

These genetically modified solutions will form the offspring population *Q(t)*. There are multiple crossover operators, like single-point crossover (Mendes, 2013; Lin et al., 2010; Falkenauer & Bouffouix, 1991; Karadgi & Hiremath, 2023; Eiben & Smith, 2015), two-point crossover (Mendes, 2013; Eiben & Smith, 2015), and many others. In the current work, when two parents are chosen from $M_p(t)$, a single-point crossover operation is carried out if random r_c is less than the crossover probability P_c, where a random length p is selected for splitting and combining the two selected parent solutions into two offspring, as illustrated in Fig. 10, otherwise, two chosen parent solutions are treated as two offspring. Additionally, mutation operation is carried out on the offspring population to enhance the solution space (Eiben & Smith, 2015; Lin et al., 2010; Falkenauer and Bouffouix, 1991). There are multiple mutation operators, and the current work performs single-point (Falkenauer and Bouffouix, 1991; Karadgi and Hiremath, 2023) (see Fig. 11), swapping (Lin et al., 2010) (see Fig. 12) and inversion (Lin et al., 2010; Falkenauer and Bouffouix, 1991; Govi et al., 2021) (see Fig. 13) mutation types if random r_m is less than the mutation probability P_m. The various strings (e.g., sorted random key, job index) of solution representation are recalculated after GA operations.

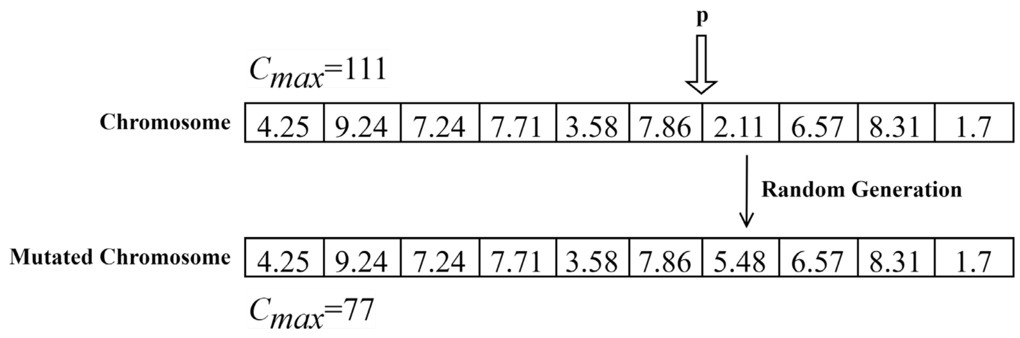


Fig. 11. Single-point mutation operation, other details of the solution not shown

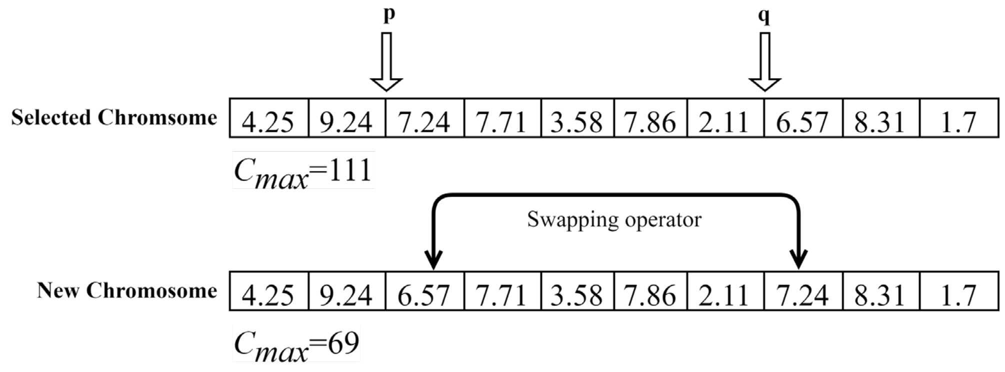


Fig. 12. Swapping mutation operation, other details of the solution not shown

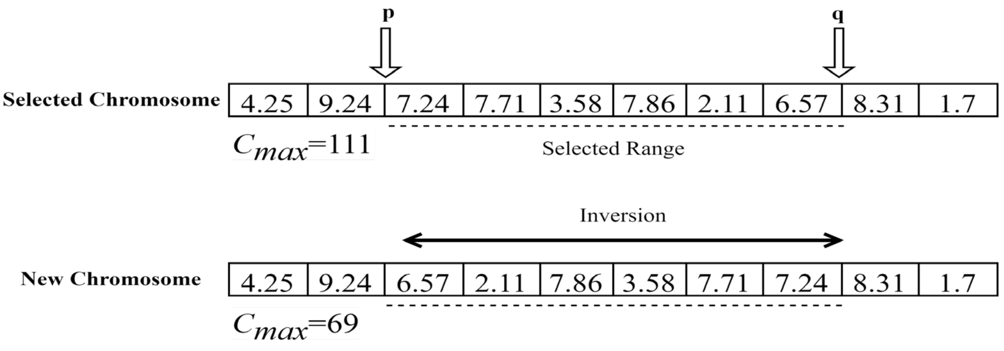


Fig. 13. Inversion mutation operation, other details of the solution not shown

4.2.5. Fitness Evaluation and Survivor Selection Strategy

The fitness is evaluated for every member of the offspring population by the fitness function, which is the makespan C_{max}. A smaller makespan implies better fitness and higher utilization of machines. The older population *P(t)* (i.e., parents), along with newly generated offspring *O(t)*, must form the next parent generation *P(t+1)* of size *N'*, by selecting the survivors of both populations based on a selection strategy (Eiben & Smith, 2015). There are multiple strategies to identify new populations. In the age-based replacement strategy, the replacement is based on the age of the solutions, and it does not prioritize fitness; instead, it prioritizes the offspring population (Pinedo, 2016; Karadgi & Hiremath, 2023). In fitness-based replacement, the previous parent population is replaced based on the fitness values of both the parent and offspring populations (Pinedo, 2016; Govi et al., 2021). In the current work, the Replace-Worst fitness-based selection strategy is employed, which involves sorting the total *P(t)* and *O(t)* population based on their fitness and replacing the current population with the solutions with better fitness (Govi et al., 2021; Pinedo, 2016). This implies that only the fittest members will show their presence in the future generations, and all unfit solutions, be parents or offspring, will be left behind if these solutions are not better than the competitors.

4.2.6. Termination

The termination condition for the GA implementation includes two possible choices. One condition for the termination is stagnancy (Karadgi and Hiremath, 2023), i.e., the number of consecutive generations in which the best fitness does not change. The stagnancy counter will be reset if the fitness changes at least once during any consecutive generation. In the implementation, the empirical default value for termination due to stagnancy varies based on the N' and T. The second condition for termination is reaching the maximum number of generations T. The current work employs both these conditions for termination.

5. Computational Experiments

Section 4.1 presented the mathematical formulation of MILP to solve the scheduling of AMRs associated with LAS, and the corresponding meta-heuristic approach GA was described in Section 4.2. These solutions are solved for various data sets, and the following paragraphs present the results.

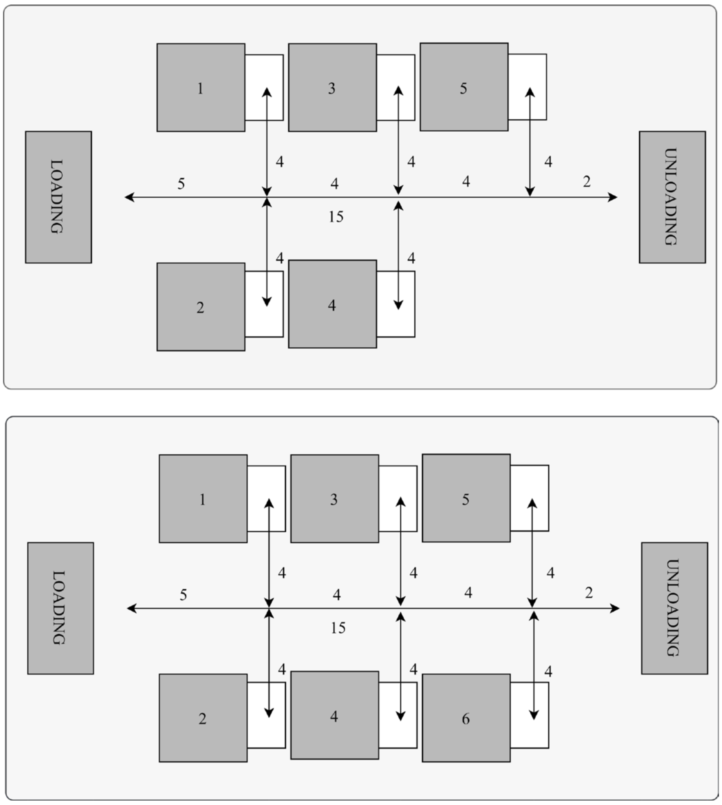


Fig. 14. Schematic shop floor layout with 5-machine and 6-machine environment (note: the numbers on the arrow indicate the distance)

There exists numerous benchmark data sets associated with the JSSP (e.g., Taillard (TA) (Taillard, 1993), Fisher Thompson (FT) (Muth and Thompson, 1963), and Lawrence (LA) (Lawrence, 1984). However, these data sets need to be extended to include the number of AMRs, distance matrix, velocity vector, and so forth corresponding to LAS, which has been done as a novel contribution to this work. The efficient arrangement of machines on the shop floor significantly impacts resource utilization. However, this aspect is beyond the scope of this article. Fig. 14 and Fig. 15 illustrate the shop floor layout with different numbers of machines, which assist in calculating the distance between machines and loading and unloading stations. The distance matrix and the speed or velocity vector of AMR will assist in determining the time taken to travel between machines and stations. The computations of MILP and GA approaches towards solving the LAS problem are performed on Intel Xeon CPU E5-2420 V2 2.20GHz with Windows 10 64-bit OS with 64 GB RAM using Python programming language. Python-MIP package is employed to solve the MILP with an underlying CBC (COIN-OR Branch-and-Cut) solver (PYTHON-MIP, 2024). Further, GA is implemented using Python 3.9.16. Different benchmark data sets are chosen randomly and extended to suit the LAS scenario. The results of the computational experiments for these data sets are displayed in Table 2.

Table 2
Comparison of C_{max} from mathematical formulation and corresponding fitness of GA implementation

Instance	Problem Size *mxnxa*	#Integer Decision Variables (MILP)	#Binary Decision Variables (MILP)	#Constraints (MILP)	CPU Time (s, MILP)	Objective C_{max} (MILP)	N (GA)	T (GA)	P_c (GA)	P_m (GA)	#Similar Fitness (GA)	CPU Time (s, GA)	Fitness (GA)
FT06	6×6×2	65	592	524	120.81	**177**	30	300	0.7	0.5	40	2.2	**177**
FT06	6×6×3	65	600	584	315.46	**123**	30	350	0.8	0.5	40	2.99	127
LA06	5×15×3	120	2363	3262	86400.06[§]	2188	50	450	0.7	0.5	40	10.9	**1821**
LA06	5×15×4	120	2380	3682	86401.02[§]	**1540**	50	500	0.7	0.5	40	8.19	1628
FT10	10×10×2	145	1896	2052	86401.48[§]	**3040**	100	600	0.7	0.5	60	23.41	3281
FT10	10×10×3	145	1908	2232	86401.41[§]	**2118**	100	700	0.7	0.5	60	25.29	2605
TA21	20×20×3	485	11198	14962	86401.41[§]	18927	100	700	0.7	0.5	60	236.31	**9006**
TA21	20×20×4	485	11220	15722	86401.51[§]	19920	100	700	0.7	0.5	60	76.39	**7282**
TA21	20×20×5	485	11242	16482	86401.65[§]	19458	100	700	0.7	0.5	60	174.98	**6846**
TA41	20×30×4	705	23680	35272	86400.00[§]	-[@]	100	1000	0.7	0.5	60	446.49	**11731**
TA41	20×30×5	705	23712	37012	86400.62[§]	28415	100	1000	0.7	0.5	60	150.11	**10580**
TA41	20×30×6	705	23744	38752	86400.00[§]	-[@]	100	1000	0.7	0.5	60	388.23	**9516**
TA61	20×50×5	1145	62452	102672	86400.00[§]	-[@]	100	1200	0.7	0.5	60	374.72	**17183**
TA61	20×50×6	1145	62504	107572	86415.57[§]	-[@]	100	1200	0.8	0.5	60	448.56	**15148**
TA61	20×50×7	1145	62556	112472	86477.22[§]	43347	100	1200	0.7	0.5	60	281.91	**14298**
TA71	20×100×8	2245	240108	469722	86400.66[§]	38554	120	1300	0.7	0.5	80	4237.87	**26153**
TA71	20×100×9	2245	240210	489522	86400.00[§]	-[@]	120	1300	0.7	0.5	80	1801.78	**25187**
TA71	20×100×10	2245	240312	509322	86400.00[§]	-[@]	120	1300	0.7	0.5	80	2973.69	**23822**

[§] The solver was terminated after ≈24 hours (or ≈86,400 s) of computational time.
[@] No valid solution was found after ≈24 hours (or ≈86,400 s) of computational time.
Time units of C_{max} depend on the time units of the machine processing times and AMR travel times on the shop floor.

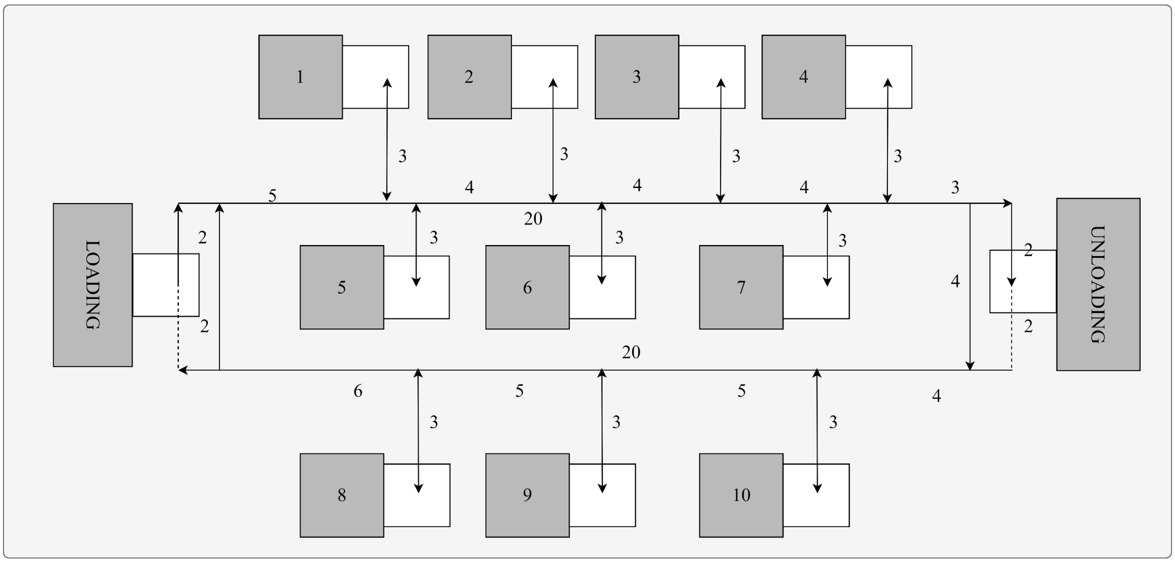


Fig. 15. Schematic shopfloor layout with 10-machine environment (note: the numbers on the arrow indicate

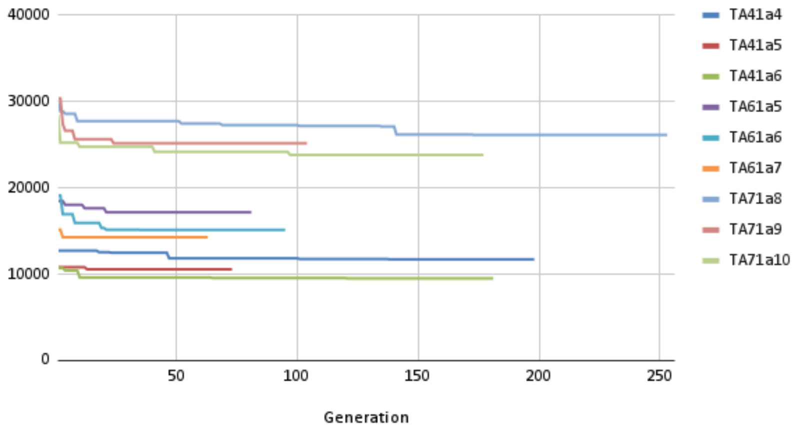


Fig. 16. Evolution of fitness value for some sample TA instances

MILP was solved for significantly bigger problem sizes (e.g., TA61, TA71) using the Python-MIP package. These bigger problem sizes were also solved using commercial Gurobi Solver (Gurobi, 2021). In both cases, the solver ran without any upper limit on the solver runtime. These solvers could not converge to the optimal solution even after 4 days. Subsequently, the solver was run for 24 hours for all the instances listed in Table 2. It is observed that for the smaller data sets (e.g., FT06), obtaining the solution by solving the MILP is advantageous. However, as the problem size increases, MILP loses its advantages as it is unable to determine optimally C_{max}, and in some instances, the solver could not even determine feasible solutions. It is observed that, for bigger problem sizes, the GA implementation could provide a feasible solution in a few minutes, and the % Gap was not comparable due to a lack of optimal/feasible values. The evolution of fitness of the GA implementation for various TA instances is illustrated in Fig. 16. A sample output of the GA implementation is presented as a Gantt chart in Fig. 17. Here, it is essential to note that the travel time between the machines and different stations is also considered, which is close to real-world scenarios. Also, an AMR will travel to the loading station after unloading the job at the unloading station.

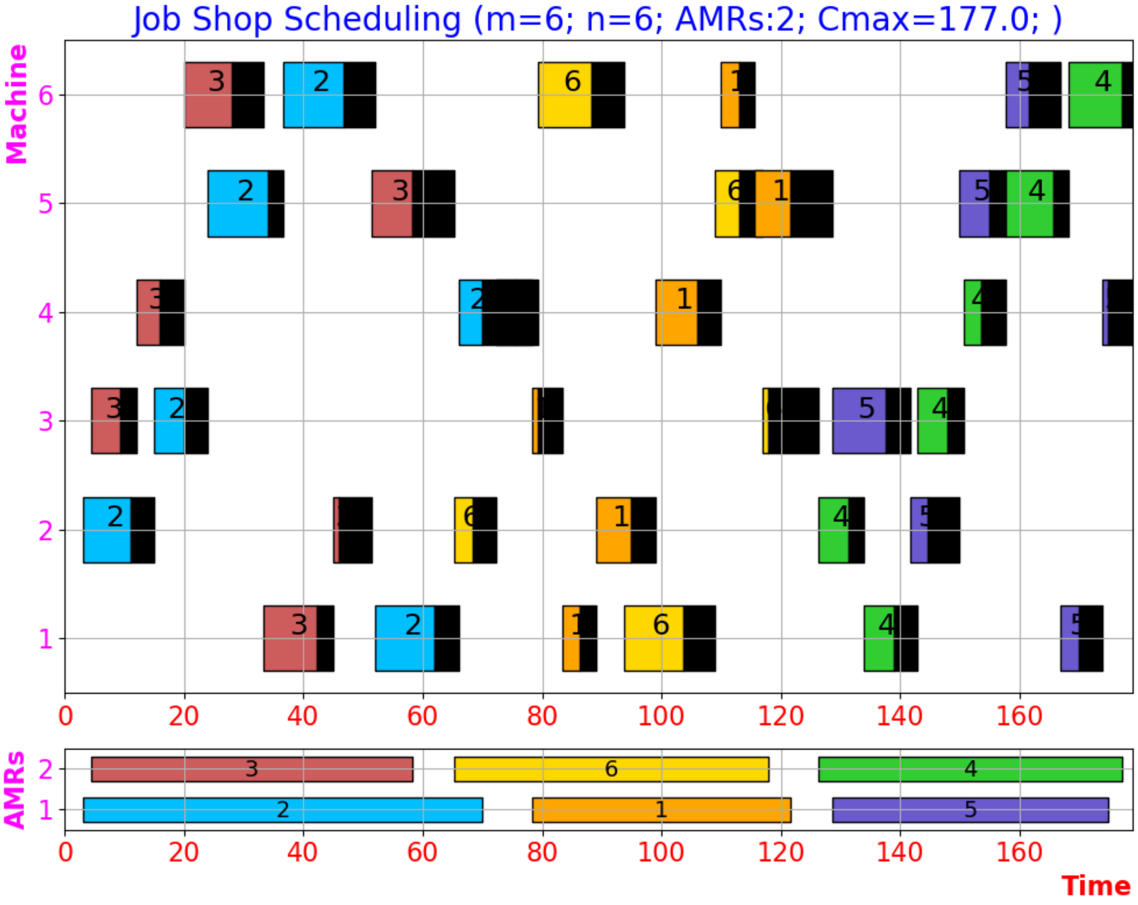


Fig. 17. Schedule of jobs and AMRs as Gantt chart for GA implementation with m=6, n=6, a=2, and C_{max}=177

Considering the instance TA71 with problem size 20x100x10 in Table 2, it is observed that GA has yielded the near-optimal solution C_{max} = 23822 in 2673.69 seconds, i.e., 0.74 hours (CPU time), while the MILP solver could not converge to a solution even after 24 hours (CPU time). Thus, the computational results in Table 2 demonstrate the utility and relevance of metaheuristics GA in optimizing LAS when it scales up.

Apart from the Gantt chart, the GAZEBO simulation has been conducted in a virtual machine on AMD Ryzen 5 5600H with Radeon Graphics@3.30 GHz CPU with 16GB RAM and Windows 11 23H, and RTX3050 having 4GB dedicated video memory. The virtual machine runs Ubuntu 22.04 with 8GB RAM and 60GB of storage allocated to it. ROS2 Humble is used to speed up the development of the simulation. A four-machine shop-floor layout is set up for the simulation experiment using the same data from Fig. 4. The shop-floor setup can be viewed in Figs. 18-20. The optimized output is decoded into a JSON file from which the goal poses and other details are read. The AMR description and model used are custom-made in SolidWorks. However, TurtleBot3 can also be used for the simulation (Amsters & Slaets, 2020). The movement of AMRs is displayed in Figure 19. This parsed data is transmitted to the ROS2 navigation framework (Macenski et al., 2020), controlling the AMR movement. The navigation stack uses this data to instruct the simulated robots within Gazebo on executing the scheduled tasks using the functions from the simple commander API. The keep-out filter plugin has ensured that the AMRs only move in the permitted work zone, as seen in Fig. 21.

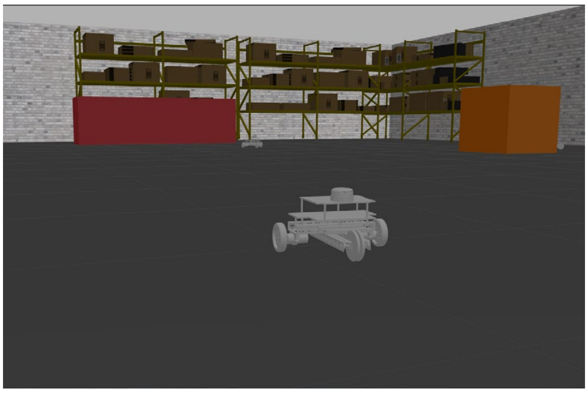


Fig. 18. AMR on the shop floor

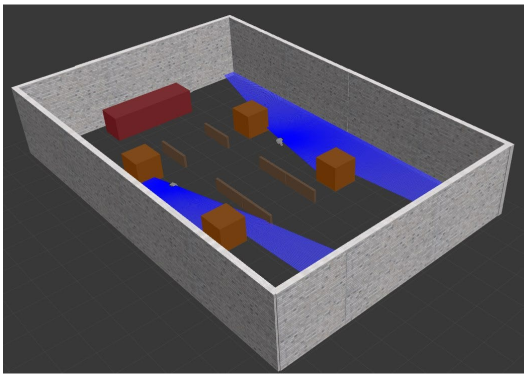


Fig. 19. Isometric view of the shop floor layout for a 4-machine environment, blue color indicating LIDAR sensor scan

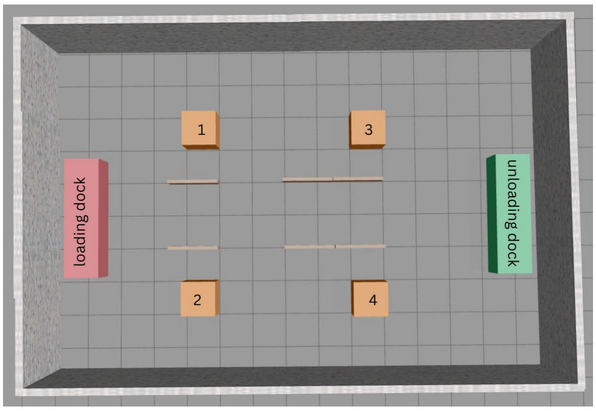


Fig. 20. Top view of the shop floor layout for a 4-machine environment

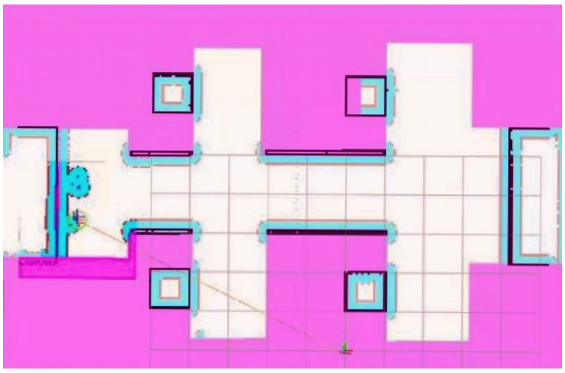


Fig. 21. Rviz view with keep-out filter in ROS platform of single AMR

6. Conclusions and Future Work

This paper elaborated on a scheduling algorithm that creates a near-optimal schedule for the LAS system by accounting for the number of available AMRs and the travel time between different machines. The details of the problem at hand are discussed in Section 2. The problem is addressed by designing a GA and a MILP formulation, which is described in depth in Section 4. Section 5 presents the results of data on which the experiments have been conducted (see Table 2). Details about the simulation experiment are also discussed. Thus, the algorithms developed are feasible for solving LAS's JSSP, especially when LAS scales up. Future work on this article will focus on some different areas of research, such as deterministic scheduling for a Flexible Job shop setup (Pinedo, 2016) for addressing the availability of similar machine types in the shop-floor layout, stochastic scheduling of the Job Shop to handle uncertainties in the shop-floor (e.g., movement of AMRs), designing of a software architecture for decentralized control of agents in a multi-agent system, targeting specific use case scenarios commonly occurring surrounding LAS.

References

Agnetis, A., Flamini, M., Nicosia, G., & Pacifici, A. (2011). A job-shop problem with one additional resource type. *Journal of Scheduling 14*, 225–237. doi:10.1007/s10951-010-0162-4.

Akhtar, M., Hamid, B., Ur-Rehman, I., Humayun, M., Hamayun, M., & Khurshid, H. (2015). An optimized shortest job first scheduling algorithm for CPU scheduling. *Journal of Applied Environent and Biological Science, 5*(12), 42-46, 2015 5, 42–46.

Amsters, R., & Slaets, P. (2020). *Turtlebot 3 as a robotics education platform, in: Robotics in Education: Current Research and Innovations 10*, Springer. pp. 170–181. doi:10.1007/978-3-030-26945-6_16.

Andrade, J., Canca, D., González-R, P., & Calle, M. (2020). Scheduling a dual-resource flexible job shop with makespan and due date-related criteria. *Annals of Operations Research 291*. doi:10.1007/s10479-019-03196-0.

Barnes, J.W., & Chambers, J.B. (1995). Solving the job shop scheduling problem with tabu search. *IIE Transactions 27*, 257–263. doi:10.1080/07408179508936739.

Bierwirth, C. (1995). A generalized permutation approach to job shop scheduling with genetic algorithms. *Operations-Research-Spektrum*, *17*(2), 87-92.

Brandimarte Data Set (2024). Brandimarte Data Set. https://shorturl.at/AITm5. Accessed on 2024-10-06.

Buckhorst, A., do Canto, M., Rabelo, R., & Schmitt, R. (2022a). A holonic control system approach for line-less mobile assembly system operations. *Procedia CIRP, 107*, 1449–1454. doi:10.1016/j.procir.2022.05.173. leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022.

Buckhorst, A., do Canto, M., & Schmitt, R. (2022b). The line-less mobile assembly system simultaneous scheduling and location problem. *Procedia CIRP, 106*, 203–208. doi:10.1016/j.procir.2022.02.179. 9th CIRP Conference on Assembly Technology and Systems.

Buckhorst, A.F., Grahn, L., & Schmitt, R.H. (2022c). Decentralized holonic control system model for line-less mobile assembly systems. *Robotics and Computer-Integrated Manufacturing 75*, 102301. doi:10.1016/j.rcim.2021.102301.

Buckhorst, A.F., Grahn, L., Schmitt, R.H. (2022d). Decentralized holonic control system model for line-less mobile assembly systems. *Robotics and Computer-Integrated Manufacturing, 75*, 102301. doi:10.1016/j.rcim.2021.102301.

Buckhorst, A.F., Montavon, B., Wolfschläger, D., Buchsbaum, M., Shahidi, A., Petruck, H., Kunze, I., Pennekamp, J., Brecher, C., Hüsing, M., Corves, B., Nitsch, V., Wehrle, K., & Schmitt, R.H. (2021). Holarchy for line-less mobile assembly systems operation in the context of the internet of production. *Procedia CIRP, 99*, 448–453. doi:10.1016/j.procir.2021.03.064. 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020.

Eiben, A.E., & Smith, J.E. (2015). *Introduction to Evolutionary Computing*. 2nd ed., Springer Publishing Company, Incorporated.

El-Ashhab, M. S., Munshi, S., Oreijah, M., & Ghulman, H. A. (2017). Job shop scheduling using mixed integer programming. *International Journal of Modern Engineering Research*, *7*(3), 2.

Falkenauer, E., & Bouffouix, S. (1991). A genetic algorithm for job shop, in: *Proceedings. 1991 IEEE International Conference on Robotics and Automation, pp. 824–829* vol.1. doi:10.1109/ROBOT.1991.131689.

Flexible Job Shop Problem (2024). Flexible Job Shop Problem. https://people.idsia.ch/ monaldo/fjsp.html. Accessed on 2024-10-06.

Fraser, J. (2016). Manufacturing Software for Industry 4.0 - Embracing Change and Decentralization for Success. Technical Report. Iyno Advisors Inc.

Göppert, A., Mohring, L., & Schmitt, R.H. (2021). Predicting performance indicators with anns for ai-based online scheduling in dynamically interconnected assembly systems. *Production Engineering 15*, 619–633.

Govi, D., Rizzuto, A., Schipani, F., & Lazzeri, A. (2021). A two-stage genetic algorithm for a novel fjsp with working centers in a real-world industrial application, in: *International Conference on Innovative Intelligent Industrial Production and Logistics,* pp. 75–83. doi:10.5220/0010654900003062.

Gu, X.L., Huang, M., & Liang, X. (2020). A discrete particle swarm optimization algorithm with adaptive inertia weight for solving multiobjective flexible job-shop scheduling problem. *IEEE Access 8*, 33125–33136. doi:10.1109/ACCESS.2020.2974014.

Gurobi (2021). Gurobi optimization. https://www.gurobi.com/. Accessed on 2024-10-27.

Hüttemann, G., Buckhorst, A.F., & Schmitt, R.H. (2019). Modelling and assessing line-less mobile assembly systems. Procedia CIRP 81, 724–729. doi:10.1016/j.procir.2019.03.184. *52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia*, June 12-14, 2019.

Jamrus, T., Chien, C.F., Gen, M., & Sethanan, K. (2018). Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing 31*, 32–41. doi:10.1109/TSM.2017.2758380.

Jia, S., Hu, & Z.H. (2014). Path-relinking tabu search for the multi-objective flexible job shop scheduling problem. *Computers & Operations Research 47*, 11–26. doi:10.1016/j.cor.2014.01.010.

Kacem, I., Hammadi, S., & Borne, P. (2002a). Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 32*, 1–13.

Kacem, I., Hammadi, S., & Borne, P. (2002b). Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation, 60*, 245–276. doi:10.1016/S0378-4754(02)00019-8.

Kagermann, H., Wahlster, W., & Helbig, J. (2013). Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. Technical Report April. Acatech.

Kamble, S., Mane, S., & Umbarkar, A. (2015). Hybrid multi-objective particle swarm optimization for flexible job shop scheduling problem. *International Journal of Intelligent Systems and Applications, 7*, 54–61. doi:10.5815/ijisa.2015.04.08.

Karadgi, S., & Hiremath, P. S. (2023). Job scheduling on parallel machines with precedence constraints using mathematical formulation and genetic algorithm, in: Sharma, S., Subudhi, B., Sahu, U.K. (Eds.), Intelligent Control, Robotics, and Industrial Automation, Springer Nature Singapore, Singapore. pp. 835–847.

Kaven, L., Rachner, J., Schmid, T., Göppert, A., & Schmitt, R.H. (2022). Reactive online scheduling of mobile resources for adaptive layout evolution in line-less assembly system. *Procedia CIRP 107, 270–275. doi:10.1016/j.procir.2022.04.044. leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022.*

Lawrence, S. (1984). Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques (supplement). Graduate School of Industrial Administration, Carnegie-Mellon University.

Leiva, C. (2018). Three Functional Dimensions Converge on Smart Manufacturing. NIST.

Lin, T.L., Horng, S.J., Kao, T.W., Chen, Y.H., Run, R.S., Chen, R.J., Lai, J.L., & Kuo, I.H. (2010). An efficient job-shop scheduling algorithm based on particle swarm optimization. *Expert Systems with Applications, 37*, 2629–2636. doi:10.1016/j.eswa.2009.08.015.

Liu, M., Lv, J., Du, S., Deng, Y., Shen, X., & Zhou, Y. (2024). Multi-resource constrained flexible job shop scheduling problem with fixture-pallet combinatorial optimisation. *Computers & Industrial Engineering, 188*, 109903. doi:10.1016/j.cie.2024.109903.

Liu, S.Q., Kozan, E., Masoud, M., Zhang, Y., & Chan, F. (2018). Job shop scheduling with a combination of four buffering constraints. *International Journal of Production Research, 56*, 3274–3293. doi:10.1080/00207543.2017.1401240.

Liu, Z. (2007). Investigation of particle swarm optimization for job shop scheduling problem, in: Third International Conference on Natural Computation (ICNC 2007), pp. 799–803. doi:10.1109/ICNC.2007.453.

Lu, Y., Morris, K., & Frechette, S. (2016). Current Standards Landscape for Smart Manufacturing Systems. volume 8107. National Institute of Standards and Technology. doi:10.6028/NIST.IR.8107.

Macenski, S., Martín, F., White, R., & Clavero, J.G. (2020). The marathon 2: A navigation system, *in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 2718–2725. doi:10.1109/IROS45743.2020.9341207.

Martin, P., & Shmoys, D.B. (1996). A new approach to computing optimal schedules for the job-shop scheduling problem, in: Cunningham, W.H., McCormick, S.T., Queyranne, M. (Eds.), Integer Programming and Combinatorial Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 389–403.

Mathews, J.B., Rachner, J., Kaven, L., Grunert, D., Göppert, A., & Schmitt, R.H. (2023). Industrial applications of a modular software architecture for line-less assembly systems based on interoperable digital twins. *Frontiers in Mechanical Engineering, 9*, 1113933. doi:10.3389/fmech.2023.1113933.

Mendes, J. (2013). A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Transactions on Computers, 12*, 164–173.

Mesghouni, K., Hammadi, S., & Borne, P. (2004). Evolutionary algorithms for job-shop scheduling. *International Journal of Applied Mathematics and Computer Science, 14*, 91–103.

Muth, J.F., & Thompson, G.L. (1963). *Industrial Scheduling. Prentice-Hall International series in management*. Prentice-Hall.

Nouiri, M., Bekrar, A., Jemai, A., Niar, S., & Ammari, A.C. (2018). An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing, 29*, 603–615. doi:10.1007/s10845-015-1039-3.

Oh, S.C., Wells, J.W., & Arinez, J. (2022). Conveyor-less urban-car assembly factory with vaac and matrix system. *Smart Cities, 5,* 947–963. doi:10.3390/smartcities5030047.

Park, B.J., Choi, H.R., Kim, H.S., (2003). A hybrid genetic algorithm for the job shop scheduling problems. *Computers & Industrial Engineering, 45*, 597–613. doi:10.1016/S0360-8352(03)00077-9.

Pezzella, F., Morganti, G., & Ciaschetti, G. (2008). A genetic algorithm for the flexible job-shop scheduling problem. Computers & Operations Research 35, 3202–3212. doi:10.1016/j.cor.2007.02.014. part Special Issue: Search-based Software Engineering.

Pinedo, M. (2016). *Scheduling Theory, Algorithms, and Systems.* 5th ed. [Pinedo 2016-02-11]. Springer.

PYTHON-MIP (2024). PYTHON-MIP. https://www.python-mip.com/. Accessed on 2024-10-27.

Ren, W., Yan, Y., Hu, Y., & Guan, Y. (2021). Joint optimisation for dynamic flexible job-shop scheduling problem with transportation time and resource constraints. *International Journal of Production Research 60*, 1–22. doi:10.1080/00207543.2021.1968526.

Salido, M., Escamilla, J., Giret, A., & Barber, F. (2016). A genetic algorithm for energy-efficiency in job-shop scheduling. *The International Journal of Advanced Manufacturing Technology, 85*. doi:10.1007/s00170-015-7987-0.

Satapathy, S., Prasad, V., Rani, B., Udgata, S., & Raju, S. (2017). Proceedings of the First International Conference on Computational Intelligence and Informatics: ICCII 2016. volume 507. doi:10.1007/978-981-10-2471-9.

Schmidtke, N., Rettmann, A., & Behrendt, F. (2021). Matrix production systems - requirements and influences on logistics planning for decentralized production structures, in: *Proceedings of the 54th Hawaii International Conference on System Sciences,* pp. 1665–1674. doi:10.24251/HICSS.2021.201.

Schmitt, R.H., Hüttemann, G., & Münker, S. (2021). A priori performance assessment of line-less mobile assembly systems. *CIRP Annals, 70*, 389–392. doi:10.1016/j.cirp.2021.04.059.

Sha, D., & Lin, H.H. (2010). A multi-objective pso for job-shop scheduling problems. *Expert Systems with Applications, 37*, 1065–1070. doi:10.1016/j.eswa.2009.06.041.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*, 278–285. doi:10.1016/0377-2217(93)90182-M.

Tavakkoli-Moghaddam, R., Azarkish, M., & Sadeghnejad-Barkousaraie, A. (2011). A new hybrid multi-objective pareto archive pso algorithm for a bi-objective job shop scheduling problem. *Expert Systems with Applications, 38*, 10812–10821. doi:10.1016/j.eswa.2011.02.050.

Tchernev, N., Lacomme, P., & Larabi, M. (2013). Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. *International Journal of Production Economics 143*, 24–34. doi:10.1016/j.ijpe.2010.07.012.

Watson, J.P., Whitley, L.D., & Howe, A.E. (2005). A dynamic model of tabu search for the job-shop scheduling problem, in: Kendall, G., Burke, E.K., Petrovic, S., Gendreau, M. (Eds.), Multidisciplinary Scheduling: Theory and Applications, Springer US, Boston, MA. pp. 247–266.

Wu, R., Li, Y., Guo, S., & Li, X. (2018). An efficient meta-heuristic for multi-objective flexible job shop inverse scheduling problem. *IEEE Access, 6*, 59515–59527. doi:10.1109/ACCESS.2018.2875176.

Yamada, T., & Nakano, R. (1997). Genetic algorithms for job-shop scheduling problems. *Proceedings of modern heuristic for decision support*, *6781*.

Yan, B., Bragin, M. A., & Luh, P. B. (2018). Novel formulation and resolution of job-shop scheduling problems. *IEEE Robotics and Automation Letters*, *3*(4), 3387-3393. doi:10.1109/LRA.2018.2850056.

Zarrouk, R., Bennour, I.E., & Jemai, A. (2016). Improving the runtime of the serial pso for the flexible job shop problem, in: 2016 *7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, pp. 381–385. doi:10.1109/SETIT.2016.7939899.

Zhang, G., Shao, X., Li, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering 56*, 1309–1318. doi:10.1016/j.cie.2008.07.021.

Zhang, Y., Zhu, H., & Tang, D. (2020). An improved hybrid particle swarm optimization for multi-objective flexible job-shop scheduling problem. *Kybernetes, 49*, 2873–2892. doi:10.1108/K-06-2019-0430.