

A Comparative Framework for Job Shop Scheduling Between Exact Method and Genetic Algorithm

Hasan Abdulsahib Mezaal  *, Omar Yaseen Saeed  

Engineering Supporting & Services Department, Scientific Research Commission, Baghdad, Iraq

ABSTRACT

In modern industries optimizing production processes time consumption has become very important for enhancing competitiveness and reducing operation costs. This research explores two case studies of the job shop scheduling problem (JSSP) through a comparative study applied to real-world data from the Iraqi hydraulic industry to find the optimal makespan time by using two approaches: an exact method Via linear programming LP for a small-scale problem in the first case study and constraint programming (CP-SAT) applied to the larger problem to overcome the computational limitations of LP. Artificial intelligence (AI) techniques model as genetics algorithm (GA) applied to both case studies for comparison. A model is solved using Python to minimize makespan time. The results show a significant effect of job size on model efficiency. In the first case study has 5 jobs done on five machines; the optimal Exact LP Method makespan time is 234 minutes, and (GA) shows the same result. In the second case study, the 8*8 Exact CP model optimal makespan is 209, and (GA) optimal makespan time in the second case study is 212 minutes. This research achieved a 33.1% time reduction compared to the company's traditional scheduling, providing valuable insights into the strengths and limitations of each approach, offering guidance for industries seeking to optimize production processes and reduce work time.

Keywords: Constraint programming, Genetic algorithm, Job shop scheduling, Linear programming, Makespan.

1. INTRODUCTION

The availability of goods and services that's meets the people acceptance is crucial for the standard of living in most societies, so goods are divided into two classes: consumer goods and producer goods. Producer goods are manufactured for other companies to use for manufacturing goods for consumers, so most consumer goods come from multiple manufacturing processes. The more accurate the production operations, the less waste of materials and cost with shorter the time needed to deliver the goods to consumers; also, controlling the production time helps lower energy consumption, and that is very important for reducing emissions and environmental pollution. For economical manufacturing, the

*Corresponding author

Peer review under the responsibility of University of Baghdad.

<https://doi.org/10.31026/j.eng.2026.10.10>



This is an open access article under the CC BY 4 license (<http://creativecommons.org/licenses/by/4.0/>).

Article received: 25/05/2026

Article revised: 28/08/2026

Article accepted: 29/08/2026

Article published: 01/10/2026



planning should start at the design stage and continue to materials selection, tooling processing, materials handling and job shop scheduling, which is the subject of this paper **(DeGarmo, 1984)**. The success of operations planning from production level to wide supply chains has increasingly depends on the development of optimization methods like hyper-heuristics and advanced control strategies **(Burke et al., 2013; Olhager, 2013)**. Meta-heuristic approaches have been developed to address complex optimization problems that are difficult to solve using conventional techniques. Among these approaches, Genetic Algorithms (GAs) have gained significant attention because of their ability to efficiently explore large search spaces and identify near-optimal solutions **(Blum and Roli, 2003)**. Although Linear Programming (LP) has traditionally been applied to the Job Shop Scheduling Problem (JSSP), its effectiveness decreases as problem size grows due to the NP-hard nature of the problem **(Prata et al., 2024)**. As a result, researchers have increasingly focused on more advanced optimization methods.

Recent studies demonstrate this trend. For example, **(Magalhães et al., 2025)** proposed a knowledge-guided genetic algorithm for dual-resource job shop scheduling, while **(Zhao and Lu, 2026)** investigated reinforcement learning-enhanced genetic algorithms for flexible scheduling environments. Similarly, **(Momenikorbekandi and Kalganova, 2025; Valencia-Rodriguez et al., 2026)** highlighted the growing use of evolutionary optimization techniques as effective alternatives to traditional methods that often struggle to maintain solution quality in large and complex scheduling problems.

At the same time, exact optimization techniques continue to advance. **(Yuraszeck et al., 2024)** demonstrated that Constraint Programming (CP) can provide highly effective exact solutions and mathematical benchmarks for large-scale scheduling problems, outperforming several conventional formulations, including those examined by **(Buske and Jendle, 2024)**. In earlier applications, **(Abdullah, 2022)** successfully minimized makespan using a priority-based chromosome structure in GA. Furthermore, **(Al-Ashhab and Fadag, 2018)** demonstrated the practical adaptability of GA compared to MILP (mixed-integer linear programming) in multi-product scheduling, while **(Stanković et al., 2020)** proved GA's superior convergence velocity over Tabu Search and Ant Colony algorithms. **(Ismail, 2018)** used real-world data from the Iraqi Hydraulic Industry to compare traditional heuristic methods with Ant Colony Optimization, demonstrating the practical value of metaheuristic approaches in industrial settings. Recent studies highlight the critical role of job shop scheduling to enhance manufacturing efficiency and lowering energy use. While traditional exact methods like MILP provide optimal solutions, they quickly become computationally expensive as problem sizes grow. To overcome this, the literature has moved towards constraint programming (CP), prove its scalability for industrial-sized problems **(Lunardi et al., 2020; Da Col and Teppan, 2022)**.

Modern production planning also needs to account for dynamic energy costs and sustainable environment **(Ham et al., 2021; Park and Ham, 2022; Terbrack and Claus, 2025; Li and Chen, 2023)**. Because exact methods alone struggle to handle all these complex constraints at once, researchers are rely on advanced genetic algorithms and hybrid meta-heuristics to find practical solutions in a reasonable time **(Zhang et al., 2020; Vital-Soto et al., 2020; Park et al., 2021)**. Despite this progress it is essential in the industry 4 era where dynamic adapting scheduling system validated by real world industrial date are required to handle real -time distribution in modern manufacturing floor **(Leusin et al., 2018; Tarigan et al., 2025)**, deep reinforcement learning has resent emerged a strong design pattern to create a scheduling model capable of learning from complex shop floor environment **(Wang et al.,**



2025). As manufacturing moves toward industry 5.0, scheduling must consider both human and machine resources. To handle this, recent studies are combining reinforcement learning with algorithm evolution to find the optimal solution (**Chang et al., 2025**). Managing large-scale operations remains a significant challenge that requires continuous methodological improvements and hybrid frameworks (**Dauzère-Pérès et al., 2024; Xiong and Liu, 2026**). While there is rapid advancement toward Industry 5.0 and reinforcement learning, there is still a significant research gap in developing a strong digital foundation for complex, heavy traditional manufacturing environments, such as the hydraulic industry in Iraq. Jumping directly to autonomous learning models is impractical without first establishing tight benchmarks comparing the limits of exact mathematical methods to those of evolutionary algorithms in these specific real-world settings. To fill this gap, the following study presents a comparative framework that evaluates Linear Programming (LP), Constraint Programming (CP-SAT), and a tailored Genetic Algorithm (GA).

Rather than introducing a new algorithm, this work is novel in providing a practical, scalable roadmap for the hydraulic industry. The goal of this framework is to minimize makespan, maximize machine utilization and provide the basis for a transitional scheduling approach whereby this work improves current traditional operations in what is expected to be a next-generation smart manufacturing environment by identifying the boundaries at which mathematical solvers stagnate and heuristic methods thrive.

2. METHODOLOGY AND INDUSTRIAL CONTEXT

Production scheduling is the process of determining when and where to perform a set of operations on available machines to achieve specific objectives (**Pinedo, 2016**). This study is an applied research study aiming to solve the Job Shop Scheduling Problem (JSSP) within the specific context of the General Company for Hydraulic Industries in Iraq.

2.1. Industrial Context and Problem Justification

The Iraqi hydraulic industry faces operational challenges when manufacturing multi-component products, like agricultural harvester parts and complex gearboxes. Traditionally, scheduling in these facilities relies on manual, experience-based dispatching rules (Shortest Processing Time). This manual approach cannot effectively handle the strict, multi-machine routing constraints required for complex components, which lead to severe machine bottlenecks, extended idle times, and prolonged makespan. The exact methods (LP and CP-SAT) and the Genetic Algorithm (GA) were chosen specifically for this study because they are mathematically proven to explore the exhaustive, NP-hard search space of operational sequences. These methods are well-suited to minimizing makespan and practically removing current manual inefficiencies through algorithmically preventing machine overlaps and optimizing start times. The methodology makes use of Python 3 for optimization and computations. This choice enriches the research, as Python is a highly effective tool for combining exact methods with genetic algorithms (**Christoforidis et al., 2025**). **Fig. 1** illustrates the flowchart of the proposed four-phase comparative optimization approach for both case studies.



Figure 1. Methodological flowchart of the proposed four-phase comparative optimization approach.

2.2. General Optimization Framework

First, the general theoretical framework of the methods used in this framework can be shown as follows before applying the models to the industrial data:

- **Exact Methods:** Linear programming (LP) formally formulates JSSP in terms of continuous and binary variables to reach the absolute optimum of having a schedule. But LP is NP-hard, and hence the time complexity of LP increases exponentially. In the case of larger problems, we mitigate this by using CP-SAT, which uses interval variables and global constraints (NoOverlap) to prune the search space efficiently.
- **Genetic Algorithm (GA):** The GA is a metaheuristic that aims to arrive at near-optimal solutions in a short time span. It encodes schedules as chromosomes (sequences of operations) and uses evolutionary operators, including selection, crossover, and mutation, in conjunction with a custom repair function to ensure that all generated schedules are still feasible (respecting machine precedence).



2.3. Implementation on Industrial Case Studies

Following the general framework, the methodology was practically implemented on two real-world datasets extracted from the Iraqi hydraulic factory (**Ismail, 2018**). This research uses real-life manufacturing data taken from the General Company for Hydraulic Industries in Iraq, as found in the thesis by (**Ismail, 2018**). The main goal for both case studies is minimize the total completion time as makespan C_{max} in Mathematics

$$C_{max} = \max(S_{ij} + t_{ij}) \quad (1)$$

Where S_{ij} is the start time and t_{ij} is the processing time of the operation j of job i

Two models were evaluated:

- 1- Case study 1 (5*5): a small (JSSP) (potato-harvester parts) with 5 jobs working on 5 machines, as listed in **Table 1**

Table 1. First case study job sequencing

Job	Job name	Sequence of machines and process time in minutes				
Job 1	PTO Yoke	M1(40) Lathe	M2(20) Milling	M3(10) Drilling	M4(20) Scraper	M5(15) Grinding
Job 2	Universal joint Yoke	M2(15)	M1(60)	M3(15)	M4(24)	M5(10)
Job 3	Harvester Yoke	M3(18)	M2(24)	M1(30)	M5(16)	M4(18)
Job 4	Sliding Arm	M1(40)	M4(12)	M3(14)	M5(16)	M2(24)
Job 5	Driving shaft	M2(20)	M3(18)	M1(30)	M4(23)	M5(16)

- 2- Case study 2 (8*8): Large scale (JSSP) (harvester gearbox parts) 8 jobs work at 8 machines, as listed in **Table 2**

Table 2. Second case study job sequencing

Job	Job name	Sequence of machine (process time mins.)							
Job 1	Main shaft gear	M1(11) Power hacksaw	M2(30) Lathe	M3(15) CNC Lathe	M4(18) Milling	M5(18) CNC Milling	M6(12) Drilling	M6(6) Scraper	M8(10) Grinding
Job 2	Second shaft gear	M1(24)	M4(26)	M5(14)	M2(15)	M3(12)	M7(12)	M6(6)	M8(7)
Job 3	Shaft holder	M1(6)	M2(10)	M3(8)	M7(10)	M4(12)	M6(8)	M5(7)	M8(6)
Job 4	Shaft bush	M2(25)	M1(4)	M3(8)	M6(10)	M4(7)	M5(4)	M7(6)	M8(4)
Job 5	Shaft holder lock	M2(28)	M1(6)	M6(9)	M4(16)	M3(14)	M5(10)	M8(6)	M7(4)
Job 6	Main frame	M1(20)	M2(30)	M6(19)	M4(17)	M3(14)	M5(12)	M8(12)	M7(8)
Job 7	Driver Bevel Gear	M1(16)	M6(15)	M2(14)	M3(12)	M4(8)	M5(6)	M7(4)	M8(6)
Job 8	Pinion bevel Gear	M2(20)	M1(10)	M6(25)	M3(15)	M4(10)	M5(10)	M7(12)	M8(8)

2.3.1 Mixed Integer Linear Programming (MILP) for Case Study 1 (5×5)

To evaluate the proposed AI algorithm, strict mathematical benchmarks were established. Due to the NP-hard nature of the JSSP, the exact methodology was divided based on the



problem size. Linear Programming (LP) is a mathematical optimization technique used to maximize or minimize a linear objective function subject to a set of linear constraints. LP is widely applied in economics, engineering, management, military science, and manufacturing due to its ability to solve resource allocation problems efficiently. Linear Programming is used in production scheduling to improve production efficiency by minimizing costs or maximizing profits, optimize resource utilization (raw materials, labor, time), meet demand while respecting operational constraints and minimize makespan (the total time required to complete all jobs (Pinedo, 2016; Birgin et al., 2014)).

A job shop scheduling problem is defined where:

- Jobs: J_1, J_2, J_3, J_4, J_5
- Machines: M_1, M_2, M_3, M_4, M_5
- Operations: Each job J_i consists of 5 sequential operations O_{ij} , where $j = 1, 2, 3, 4, 5$

Table 3 shows and defines the variables that will be used in the formulation

Table 3. Decision Variables

Symbol	Description
S_{ij}	Start time of operation j of job i
C_{max}	Makespan (the total completion time of the schedule)
$y_{iljm} \in \{0,1\}$	Binary variable: 1 if operation j of job i precedes operation m of job l on the same machine
t_{ij}	Processing time of operation j of job i

Minimize the total completion time (makespan):

- **min** C_{max}

-Constraints

1. Operation Precedence (within job):

$$S_{ij+1} \geq S_{ij} + t_{ij} \quad (2)$$

2. Machine Capacity Constraint (no overlapping):

$$S_{ij} \geq S_{lm} + t_{lm} - M(1 - y_{iljm}) \quad (3)$$

$$S_{lm} \geq S_{ij} + t_{ij} - M(y_{iljm}) \quad (4)$$

3. Makespan Definition:

$$C_{max} \geq S_{ij} + t_{ij}, \forall i \quad (5)$$

4. Start Times Non-Negative:

$$S_{ij} \geq 0, \forall i, j \quad (6)$$

2.3.2 Constraint Programming (CP-SAT) for Case Study 2 (8×8)

Because of the increasing problem size 8*8, the MILP and big-M formulation requires an impractical number of binary precedence variables, making standard solvers stall without reaching optimal solution. To overcome that challenge, a constraint programming (CP) model was formulated using the Google Or-Tool (CP-SAT solver). Unlike MILP, which relies on continuous variables and binary choices to represent scheduling decisions, this makes



the search space very large and complex a Cp use interval variable to directly represent the start and end times of tasks. It also applies global constraints to reduce the search space through active propagated constraint like tasks cannot overlap on the same machine. To clarify, the mathematical formulation for the proposed Cp model is defined as follows **(Pinedo, 2016; Prata et al., 2024; Yuraszeck et al., 2024)**:

1. interval variables: For each operation O_{ij} (operation j of job i), interval variable V_{ij} is defined. It encapsulates a start time S_{ij} , a fixed processing time t_{ij} , and an end time E_{ij} , such that:

$$E_{ij} = S_{ij} + t_{ij} \quad (7)$$

The domain of both S_{ij} and E_{ij} is bounded by $[0, H]$, where H is a sufficiently large upper bound Time Horizon representing the worst-case makespan.

2. Precede Constrains: to make sure that routing of each job is strict, the time of ending an operation must precede or equal the start time of its subsequent operation:

$$E_{ij} \leq S_{i,j+1} \forall i, \forall j \in \{1, \dots, n_i - 1\} \quad (8)$$

3. Machine capacity no overlapping Global Constraints: instead of formulating complex Big-M binary equations, CP enforce machine capacity using a global constraint. Let M_k be the subset of all interval variables assigned to a specific machine k . The Google OR-Tools solver makes sure no two intervals within this subset overlap in time:

$$NoOverlap(V_{ij} \in M_k) \forall k \in \{1, \dots, m\} \quad (9)$$

This constraint filters out logically infeasible start times from the variable domains simultaneously, making it highly efficient for the JSSP. So the objective is to minimize the makespan time, which is defined as the maximum end time among the final operations of all scheduled jobs:

$$Minimize C_{max} = \max_i(E_{i,last}) \quad (10)$$

2.3.3. The Proposed Genetic Algorithm (GA) Formulation

To provide a fast, flexible, and scalable solution, a customized Genetic Algorithm (GA) was proposed. A GA model was built where chromosomes represent the operation sequences. The fitness function was designed based on the total Makespan. Genetic operations like crossover, mutation, and selection were applied using Python and the DEAP library **(Stanković et al., 2020; Shi, 1997; Gonçalves et al., 2002)**.

- Chromosome Represent and Fitness Evaluation

To solve the JSSP the schedule needs to be represented as a mathematical array chromosome. If job has 5 operations, its ID appears 5 times in the array. Mathematically, a chromosome C is an array of length L , where L is the total number of operations (Total jobs * Machines):

$$C = [g1, g2, \dots, gL] \quad (11)$$

Where each generation represents a job ID

$$gk \in [1, 2, \dots, n] \quad (12)$$



The algorithm starts by generating an initial population of N random chromosomes to explore the different scheduling possibilities.

For each chromosome, the algorithm reads the array from left to right, assigns the operations to their required machines, and calculate the total makespan time. The fitness function $f(C)$ simply equals the calculated makespan:

$$f(C) = C_{max}(C) \quad (13)$$

The objective of the algorithm is to minimize $f(C)$. A lower makespan means better fitness score

- The Evolutionary Loop (operations and Repair)

if the terminate condition isn't met, the population evolves through the following sequential steps, as listed below and shown in **Fig.1**

1. Selection operator:

Tournament selection is used. The algorithm randomly picks a small group of chromosomes (tournament size k) from the population. The one with the lowest makespan wins and is selected as a parent. In mathematics, the selected parent p from random subset s is

$$P = \arg \min_{i \in s} f(c_i) \quad (14)$$

2. Crossover Operator

To create a new child schedule (o_1) from two selected parents (p_1 and p_2) precedence-preserving crossover (POX) is applied with a probability p_c . The logic is simple, as listed:

- Randomly select a subset of jobs j_s
- if gen in p_1 belongs to j_s it copies directly to the same index in the child o_1
- fill the remaining empty spots in o_1 using the remaining job IDs from p_2 to keep their original sequence.

This mixes the traits of both parents while trying to maintain valid operation routing.

3. Mutation Operator

To avoid the algorithm from getting stuck in a local optimum, a small random change is introduced, called small swap mutation, with low probability p_m . In this step, the algorithm randomly picks two positions in the chromosome (lets call them id_{x1} and id_{x2}) in chromosome array and swap their value $C[id_{x1}] \leftrightarrow C[id_{x2}]$

To prove the industrial value of the proposed GA method, the results of GA are compared against exact benchmarks (LP and CP-SAT solvers) and the ant colony method results taken from **(Ismail, 2018)** and company's traditional heuristic. The mathematical error percentage (optimality GAP) was calculated to show if GA is close to the absolute optimal solution

$$Gap(\%) = \left((C_{max}(GA) - C_{max}(Exact)) / C_{max}(Exact) \right) \times 100 \quad (15)$$

To clearly present these findings, all Gantt charts, makespan comparisons, and convergence curves were generated using the Matplotlib library in Python, ensuring accurate and high-quality engineering data visualization.

3. RESULTS AND DISCUSSION

All models, including the exact solvers (LP and CP-SAT) and the proposed Genetic Algorithm (GA), are programmed and executed using Python 3. The exact models utilized Google OR-Tools, while the GA is developed using the DEAP computational library. This section presents the evaluated results of applying these implemented methods to the industrial case studies. To prove the effectiveness of the framework. They will be compared against exact mathematical bounds and previously documented heuristic approaches.

3.1 Results of case study 1 (5*5 problem)

The first case study is a manufacturing potato-harvester: 5 jobs on 5 machines, back to **Table 1**.

3.1.1 Exact Method Results

The problem is formulated using Mixed integer Linear programming (MILP). The solver finds that the absolute optimal makespan for this problem is exactly 234 minutes, as shown in Gantt chart **Fig. 2**.

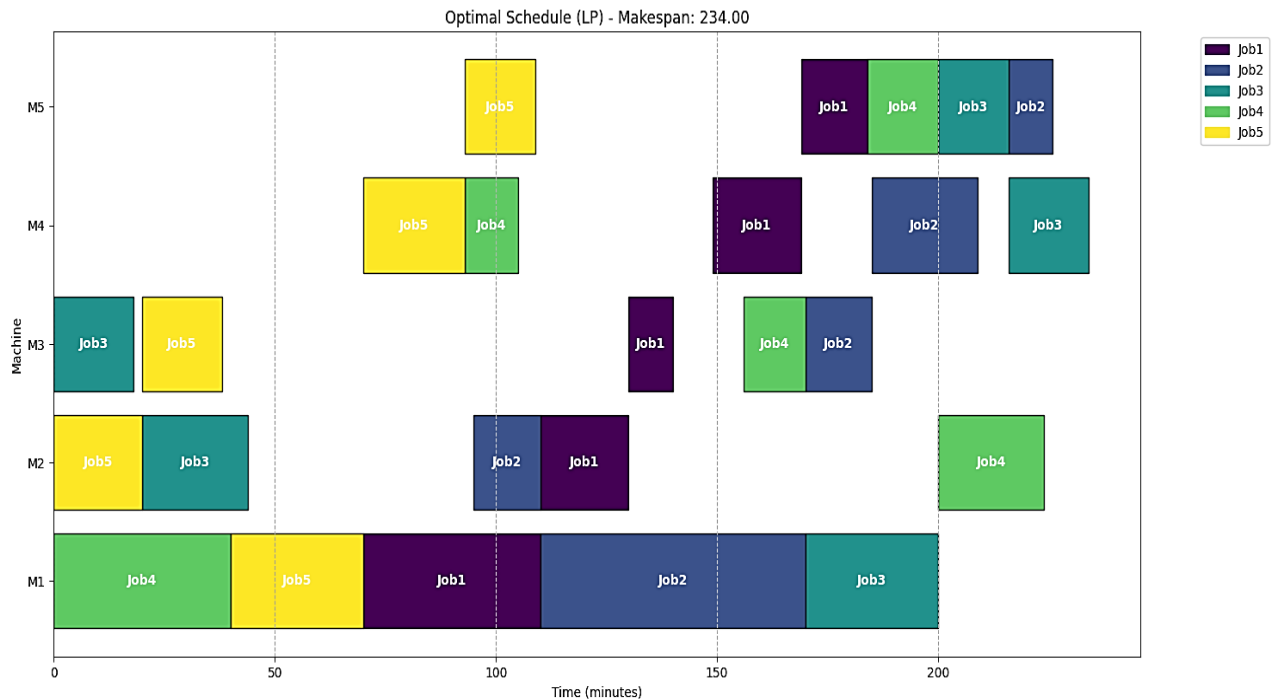


Figure 2. Gantt chart of the optimal schedule for Case Study 1 (5×5) using the exact MILP method.

3.1.2 Genetic Algorithm Results

When the proposed GA was applied to this problem, the algorithm successfully found a schedule with makespan of 234 minutes, that match the exact mathematical benchmark as shown in the Gantt chart **Fig. 3**.

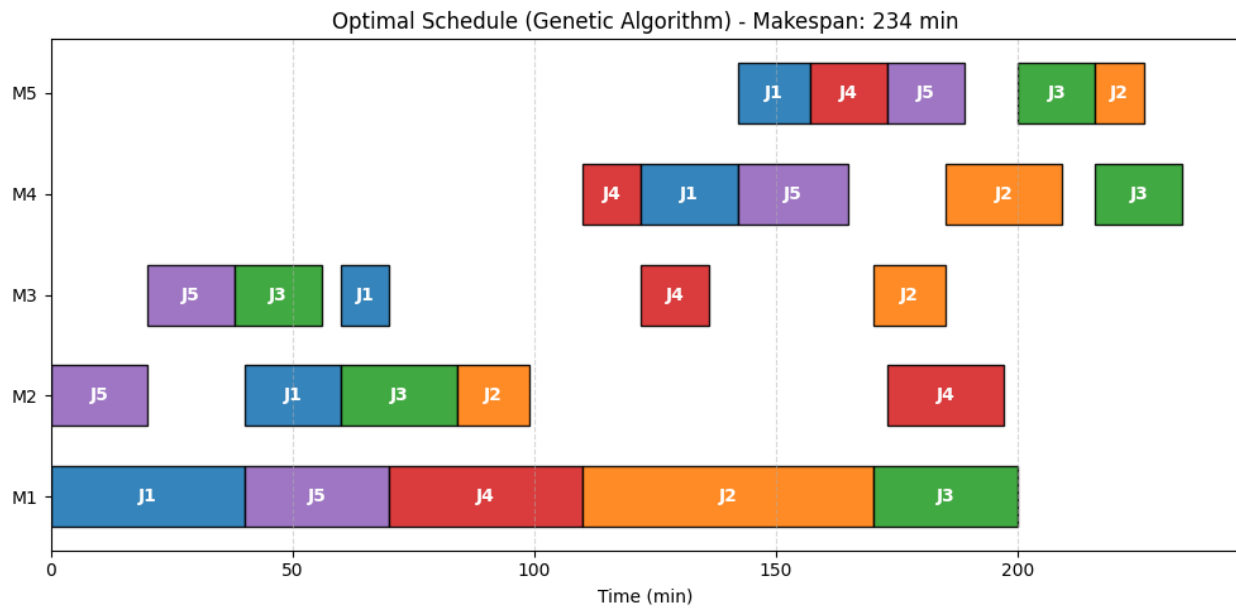


Figure 3. Gantt chart of the optimal schedule for Case Study 1 (5×5) using the proposed Genetic Algorithm.

It is worth noting that the convergence curve for this 5*5 case is not explicitly plotted because GA reached the optimal solution almost instantaneously in the initial generations because of relatively small search space. A close visual analysis of the generated Gantt charts (**Figs. 2 and 3**) reveals how both the MILP and the GA effectively managed the machine capacities. The schedules demonstrate a highly compact sequence, significantly minimizing the white spaces (idle times) between operations while strictly adhering to the sequential routing of each harvester part.

3.2 Results of Case Study 2 (8*8)

The second case study (harvester gearbox, 8 jobs on 8 machines) is more complex than the first one. The standard LP method could not handle the complex 8*8 problem due to the combinatorial explosion of variables, making the execution time in Python impractical. Therefore, the constraint programming (CP-SAT) solver was used. The CP solver successfully bypassed the mathematical bottleneck and established that the exact absolute optimal makespan is 209 minutes, as shown in Gantt Chart **Fig. 4**. The proposed GA with a repair function showed good performance in this complex scenario. As shown in the convergence curve **Fig. 5**, the algorithm improved the schedules rapidly over generations and reached a final makespan of 212 minutes. The optimal Gantt chart generated by the GA is shown in **Fig. 6**



Figure 4. Gantt chart of the optimal schedule for Case Study 2 (8×8) using the exact CP-SAT method.

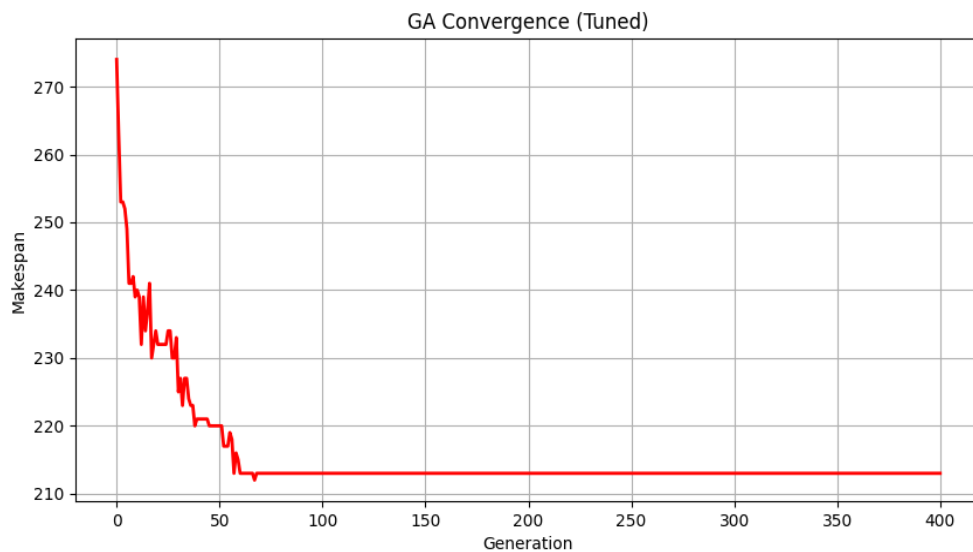


Figure 5. Convergence curve of the Genetic Algorithm for Case Study 2

Examining the Gantt charts for the 8×8 complex gearbox (**Figs. 4 and 6**), it can be seen that serious potential bottlenecks exist, unlike the smaller dataset. The produced charts show that the algorithms did. In fact, interleave operations occur concurrently on the eight machines, without overlap on the timeline. This basically confirms the robustness of the global No Overlap constraint used in the CP-SAT model and the efficiency of the custom repair function implemented within the GA to strengthen the strict job precedence.

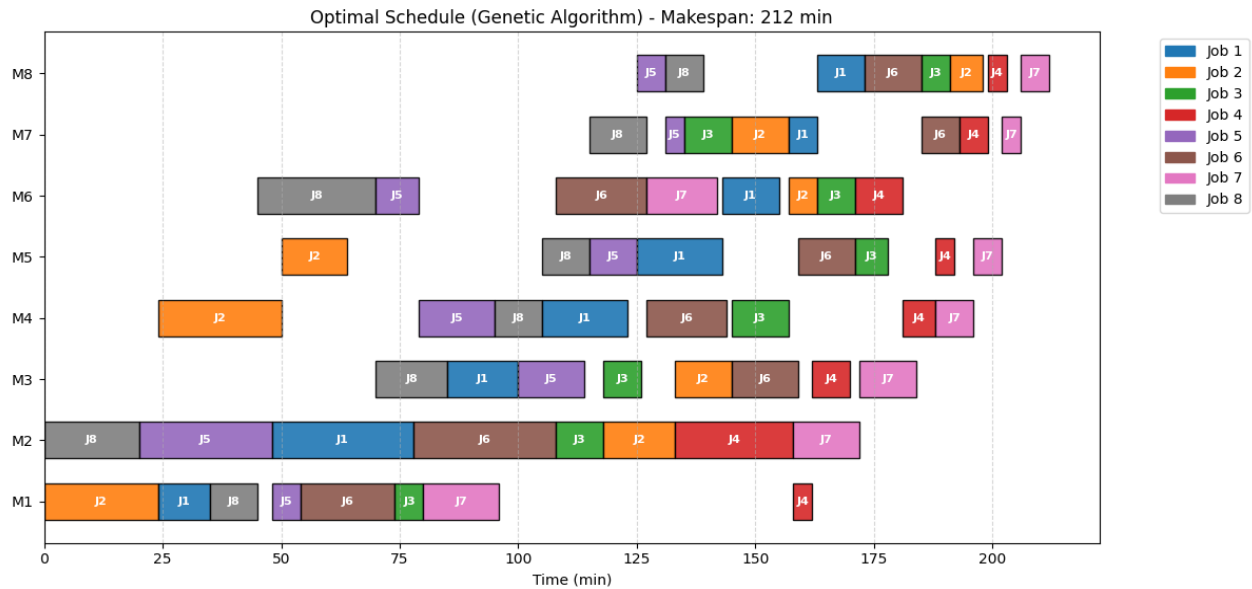


Figure 6. Gantt chart of the optimal schedule for Case Study 2 (8×8) using the proposed Genetic Algorithm.

3.3 Comparative Analysis and Industrial Impact

The value of this work appears when the results are compared with the real-word factory data, so a comprehensive comparison of all methods across both case studies is illustrated in **Fig.7**. The company's heuristic rule, Shortest Process Time (SPT) take 249 minutes to finish these jobs (**Ismail, 2018**). Both the exact LP model and the proposed GA reduced this time to 234 minutes. This result matches the 234 minutes achieved by the previous Ant Colony study (**Ismail, 2018**).

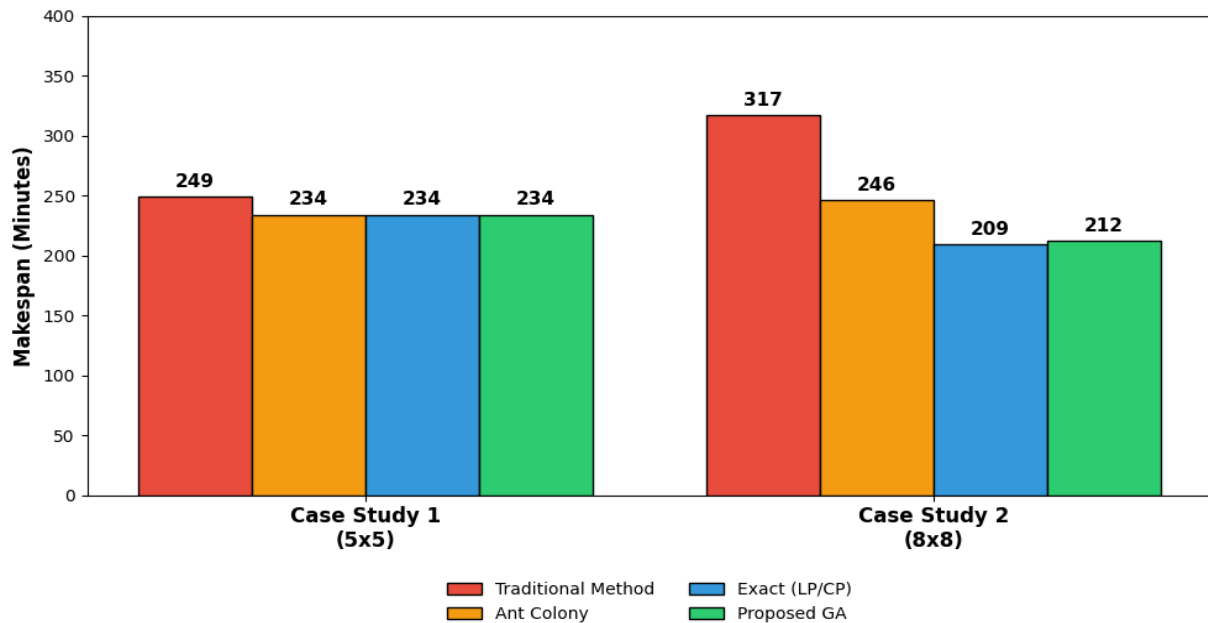


Figure 7. Comparison of the optimal makespan results across traditional, exact, and proposed heuristic methods for both case studies



Now, looking at the second case study comparison with the company and Ant Colony results, the following can be observed:

- 1- The company heuristic results using the Most Work Remaining (MWR) rule have a makespan of 317 minutes
- 2- The Ant Colony (ACO) algorithm has a makespan of 246 minutes
- 3- The customized GA dropped the time significantly to 212 minutes of makespan. This 34 - minute advantage over ACO is justified analytically. While ACO depends on probabilistic pheromone updates that may get trapped in local optima when there is complex multi-machine working. The customized GA preserved the population diversity through its crossover and mutation operators. This mechanical advantage means the GA can effectively get around the severe machine bottlenecks that are inherent in the 8x8 gearbox dataset.
- 4- The exact cp method has 209 minutes of make span

3.4 Calculating the Optimality Gap

To measure how close our methods are to each others we calculate the optimality Gap between the GA (212 min) and the exact CP solver (209 min) using Eq. (13). Finding that the error Gap is only 1.43% proves that the GA is highly reliable and provides near-optimal solutions to the exact solvers

3.5 Industrial Time Reduction

To compare the GA result (212 minutes) to the traditional method that the Iraqi hydraulic factory uses (317 minutes), the total time saved is 105 minutes per production cycle. So, Eq. (13) shows that the time reduction percentage is 33.1. In a real-world factory. By using the proposed GA schedule, the General Company for Hydraulic Industries will not only shorten the makespan by 33.1% but also analytically remove the overlap of idle times of bottleneck machines (Lathes, CNCs). The proposed model directly enhances the overall equipment effectiveness and reduces the daily energy and operational overhead costs by tightening the operation sequence without requiring any physical hardware upgrades to the Iraqi hydraulic factory. There is a difference between the methods in this study. The MILP method always finds the perfect answer, but it takes too long to calculate large problems due to the combinatorial explosion of binary precedence variables, to make the search space more effective. The CP-SAT method is faster because it uses interval variables and active global constraints (**Yurazeck et al., 2024**). However, the customized GA achieved an optimality gap of 1.43%, which demonstrates that evolutionary algorithms can reach the absolute lower bounds in the mathematical sense, without requiring the exhaustive computational overheads that are typical of exact solvers. This is in line with recent studies (**Tarigan et al., 2025; Chang et al., 2025**) confirming GA as a very robust tool for practical industrial floors.

4. LIMITATIONS AND FUTURE WORK

This study works under the assumption that processing times are deterministic and there is never a machine breakdown. In reality, unexpected events happen. Advanced artificial intelligence (AI) methods, such as deep reinforcement learning, could be used to handle sporadic machine failures and make the factory floor even smarter



5.CONCLUSIONS

The main conclusions drawn from applying exact models and the proposed GA to real industrial data are:

1. Mathematical methods (MILP) work perfectly for small tasks (5×5), and they found the optimal time of 234 minutes.
2. For larger tasks (8×8), the Constraint Programming (CP-SAT) method was able to overcome mathematical limitations to arrive at an optimal solution time of 209 minutes.
3. Our genetic algorithm (GA) solved the large task with a nearly optimal runtime of 212 minutes. That is a large improvement on both the factory's old baseline (317 minutes) and previous works by ACO methods themselves (246 minutes).
4. In a real factory setting, our method reduces production time by 33.1% (that saves 105 minutes per cycle). This directly reduces daily operational costs, increases machine efficiency, and reduces wear and tear.

NOMENCLATURE

Symbol	Description	Symbol	Description
C	Chromosome array, dimensionless.	O_{ij}	Fitness function score, dimensionless.
C_{max}	Makespan (total completion time), min.	n	Total number of jobs, dimensionless.
E_{ij}	End time of operation j of job i, min.	pc	Crossover probability, dimensionless.
$f(C)$	Fitness function score, dimensionless.	pm	Mutation probability, dimensionless.
gk	Gene representing a job ID, dimensionless.	St_{ij}	Start time of operation j of job i, min.
H	Upper bound Time Horizon, min.	t_{ij}	Processing time of operation j of job i, min.
M	A large positive number, dimensionless.	V_{ij}	Interval variable for operation j of job i.
m	Total number of machines, dimensionless.	$yljm$	Binary precedence variable, dimensionless.
		$Gap\%$	Optimality gap, %.

Acknowledgements

This work was supported by the Scientific Research Commission, Baghdad, Iraq. The authors also extend their appreciation to the General Company for Hydraulic Industries for providing the industrial real-world data, and special thanks to researcher Ahmed Hameed for his valuable efforts in collecting and documenting the foundational dataset used in this comparative study.

Credit Authorship Contribution Statement

Hasan Abdulsahib Mezaal: Conceptualization, Methodology, Software, Validation, Writing – original draft, Visualization. Omar Yaseen Saeed: Data curation, Investigation, Validation, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.



REFERENCES

- Abdullah, W.N., 2022. Solving job-shop scheduling problem using genetic algorithm approach. *Journal of the College of Basic Education*, 17(70), pp. 241–253. <https://doi.org/10.35950/cbej.vi.8491>.
- Al-Ashhab, M.S. and Fadag, H., 2018. Multi-product master production scheduling optimization modelling using mixed integer linear programming and genetic algorithms. *International Journal of Research–Granthaalayah*, 6(5), pp. 78–92. <https://doi.org/10.29121/granthaalayah.v6.i5.2018.1429>.
- Birgin, E.G., Feofiloff, P., Fernandes, C.G., de Melo, E.L., Oshiro, M.T.I. and Ronconi, D.P., 2014. A MILP model for an extended version of the flexible job shop scheduling problem. *Optimization Letters*, 8(4), pp. 1417–1431. <https://doi.org/10.1007/s11590-013-0669-7>.
- Blum, C. and Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3), pp. 268–308. <https://www.ac.tuwien.ac.at/files/pub/blum-11>.
- Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E. and Woodward, J.R., 2013. Hyperheuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12), pp. 1695–1724. <https://doi.org/10.1057/jors.2013.71>.
- Buske, J. and Jendle, T., 2024. Integer Linear Programming Applied to Production Planning. Master's thesis, Chalmers University of Technology. <http://hdl.handle.net/20.500.12380/308396>.
- Chang, X., Jia, X. and Ren, J., 2025. A reinforcement learning enhanced memetic algorithm for multi-objective flexible job shop scheduling toward Industry 5.0. *International Journal of Production Research*, 63(1), pp. 119–147. <https://doi.org/10.1080/00207543.2024.2357740>
- Christoforidis, S., Titopoulos, E., Mihaylova, B., Kromitoglou, E. and Intzes, S., 2025. Solving the job shop scheduling problem - Different techniques and programming languages. *Vocational Education*, 27(3-4), pp. 177–194. <https://doi.org/10.53656/voc25-3-4-12>
- Da Col, G. and Teppan, E.C., 2022. Industrial-size job shop scheduling with constraint programming. *Operations Research Perspectives*, 9, article 100249. <https://doi.org/10.1016/j.orp.2022.100249>.
- Dauzère-Pérès, S., Ding, J., Shen, L. and Tamssaouet, K., 2024. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2), pp. 409–432. <https://doi.org/10.1016/j.ejor.2023.05.017>.
- DeGarmo, E.P., 1984. *Materials and processes in manufacturing*. 6th ed. Macmillan publishing.
- Gonçalves, J.F., de Magalhães Mendes, J.J. and Resende, M.G.C., 2002. A hybrid genetic algorithm for the job shop scheduling problem (Technical Report TD-5EAL6J). AT&T Labs Research.
- Ham, A., Park, M.-J. and Kim, K.M., 2021. Energy-aware flexible job shop scheduling using mixed integer programming and constraint programming. *Mathematical Problems in Engineering*, 2021, article 8035806. <https://doi.org/10.1155/2021/8035806>.



Ismail, A.H., 2018. Multiple Ant Colony Algorithm for Job Shop Scheduling problem. Master's thesis, College of Engineering, University of Baghdad, Baghdad, Iraq.

Leusin, M.E., Frazzon, E.M., Uriona Maldonado, M., Kück, M. and Freitag, M., 2018. Solving the job-shop scheduling problem in the industry 4.0 era. *Technologies*, 6(4), P. 107. <https://doi.org/10.3390/technologies6040107>.

Li, Z. and Chen, Y., 2023. Minimizing the makespan and carbon emissions in the green flexible job shop scheduling problem with learning effects. *Scientific Reports*, 13, article 6369. <https://doi.org/10.1038/s41598-023-33615-z>.

Lunardi, W.T., Birgin, E.G., Laborie, P., Ronconi, D.P. and Voos, H., 2020. Mixed integer linear programming and constraint programming models for the online printing shop scheduling problem. *Computers and Operations Research*, 123, article 105020. <https://doi.org/10.1016/j.cor.2020.105020>.

Magalhães, R., Sousa, J.M.C. and Vieira, S.M., 2025. A heuristic guided genetic algorithm applied to dual resource job shop scheduling. *Mathematics*, 13(19), p.3116.

Momenikorbekandi, A. and Kalganova, T., 2025. Intelligent scheduling methods for optimisation of job shop scheduling problems in the manufacturing sector: A Systematic Review. *Electronics*, 14(8), p. 1663. <https://doi.org/10.3390/electronics14081663>.

Olhager, J., 2013. Evolution of operations planning and control: From production to supply chains. *International Journal of Production Research*, 51(23-24), pp. 6836-6843. <https://doi.org/10.1080/00207543.2012.761363>.

Park, J.-S., Ng, H.-Y., Chua, T.-J., Ng, Y.-T. and Kim, J.-W., 2021. Unified genetic algorithm approach for solving flexible job-shop scheduling problem. *Applied Sciences*, 11(14), article 6454. <https://doi.org/10.3390/app11146454>.

Park, M.-J. and Ham, A., 2022. Energy-aware flexible job shop scheduling under time-of-use pricing. *International Journal of Production Economics*, 248, article 108507. <https://doi.org/10.1016/j.ijpe.2022.108507>.

Pinedo, M.L., 2016. *Scheduling: Theory, algorithms, and systems*. 5th ed. Springer. <https://doi.org/10.1007/978-3-319-26531-1>.

Prata, B.A., Abreu, L.R. and Nagano, M.S., 2024. Applications of constraint programming in production scheduling problems: A descriptive bibliometric analysis. *Results in Control and Optimization*, 14, p. 100350. <https://doi.org/10.1016/j.rico.2023.100350>.

Shi, G.Y., 1997. A genetic algorithm applied to a classic job-shop scheduling problem. *International Journal of Systems Science*, 28(1), pp. 25-32. <https://doi.org/10.1080/00207729708929359>.

Stanković, A., Petrović, G., Čojbašić, Ž. and Marković, D., 2020. An application of metaheuristic optimization algorithms for solving the flexible job-shop scheduling problem. *Operational Research in Engineering Sciences: Theory and Applications*, 3(3), pp. 13-28. <https://doi.org/10.31181/oresta20303013s>.



- Tarigan, M., Gaol, F.L., Gunawan, A.A.S. and Budiharto, W., 2025. Advancing dynamic flexible job shop scheduling with a real-world dataset and genetic adaptive scheduling system. *IEEE Access*, 13, pp. 193020–193031. <https://doi.org/10.1109/ACCESS.2025.3630184>
- Terbrack, H. and Claus, T., 2025. The generalized energy-aware flexible job shop scheduling model: A constraint programming approach. *Computers and Industrial Engineering*, 204, article 111065. <https://doi.org/10.1016/j.cie.2025.111065>.
- Valencia-Rodriguez, A., et al., 2026. Evolutionary optimization for job shop scheduling with blocking: A genetic algorithm approach. *Algorithms*, 19(2), p.115.
- Vital-Soto, A., Azab, A. and Baki, M.F., 2020. Mathematical modeling and a hybridized bacterial foraging optimization algorithm for the flexible job-shop scheduling problem with sequencing flexibility. *Journal of Manufacturing Systems*, 54, pp. 74–93. <https://doi.org/10.1016/j.jmsy.2019.11.010>.
- Wang, S., Li, J., Jiao, Q. and Ma, F., 2025. Design patterns of deep reinforcement learning models for job shop scheduling problems. *Journal of Intelligent Manufacturing*, 36, pp. 3741–3759. <https://doi.org/10.1007/s10845-024-02454-8>
- Xiong, F. and Liu, H., 2026. Logic-based Benders decomposition methods for the distributed flexible job shop scheduling problem. *European Journal of Operational Research*, 329(3), pp. 778–797. <https://doi.org/10.1016/j.ejor.2025.08.039>.
- Yuraszeck, F., Mejía, G., Rossit, D.A. and Lüer-Villagra, A., 2024. A constraint programming-based lower bounding procedure for the job shop scheduling problem. *Computers & Operations Research*, 161, p. 106362. <https://doi.org/10.1016/j.cor.2024.106964>.
- Zhang, G., Hu, Y., Sun, J. and Zhang, W., 2020. An improved genetic algorithm for the flexible job shop scheduling problem with multiple time constraints. *Swarm and Evolutionary Computation*, 54, article 100664. <https://doi.org/10.1016/j.swevo.2020.100664>.
- Zhao, G. and Lu, J., 2026. Application of a reinforcement learning-based improved genetic algorithm in flexible job-shop scheduling problems. *Mathematics*, 14(2), P. 307.

إطار مقارنة لجدولة ورشة العمل بين الطريقة الدقيقة والخوارزمية الجينية

حسن عبد الصاحب مزعل*, عمر ياسين سعيد

قسم الاسناد الهندسي والخدمات , هيئة البحث العلمي، بغداد، العراق

الخلاصة

في الصناعات الحديثة، أصبح تحسين عمليات الإنتاج وتقليل استهلاك الوقت أمراً بالغ الأهمية لتعزيز التنافسية وتقليل تكاليف التشغيل. يستكشف هذا البحث دراستي حالة لمشكلة جدولة ورشة العمل (JSSP) من خلال دراسة مقارنة طُبقت على بيانات حقيقية من الشركة العامة للصناعات الهيدروليكية في العراق، بهدف إيجاد وقت الإنجاز الأمثل (Makespan) باستخدام نهجين: طريقة دقيقة عبر البرمجة الخطية (LP) للمشكلة صغيرة النطاق في دراسة الحالة الأولى، بينما تم تطبيق البرمجة المقيدة (CP-SAT) على المشكلة الأكبر لتجاوز القيود الحسابية للبرمجة الخطية. كما تم تطبيق نموذج تقنيات الذكاء الاصطناعي (AI) المتمثل بالخوارزمية الجينية (GA) على كلتا الحالتين لغرض المقارنة. تم حل النموذج باستخدام لغة بايثون (Python) لتقليل وقت الإنجاز. أظهرت النتائج تأثيراً كبيراً لحجم العمل على كفاءة النماذج؛ ففي دراسة الحالة الأولى (5 أعمال على خمس مكائن)، كان وقت الإنجاز الأمثل بالطريقة الدقيقة (LP) هو 234 دقيقة، وأظهرت الخوارزمية الجينية (GA) النتيجة ذاتها. وفي دراسة الحالة الثانية، حقق نموذج (CP) الدقيق (8×8) وقت إنجاز أمثل بلغ 209 دقيقة، في حين كان وقت الإنجاز الأمثل للخوارزمية الجينية (GA) في دراسة الحالة الثانية 212 دقيقة. حقق هذا البحث تقليلاً في الوقت بنسبة 33.1% مقارنة بالجدولة التقليدية للشركة، مما يوفر رؤية قيمة حول نقاط القوة والقيود لكل نهج، ويقدم إرشاداً للصناعات التي تسعى لتحسين عمليات الإنتاج وتقليل وقت العمل.

الكلمات المفتاحية: جدولة ورشة العمل، الخوارزمية الجينية، البرمجة المقيدة، البرمجة الخطية، وقت الإنجاز.