University of Baghdad
College of Engineering
JE
JOURNAL OF ENGINEERING

# Journal of Engineering

journal homepage: www.jcoeng.edu.iq

Volume 32 Number 3 March 2026

# Bi-objective Machine Scheduling Using Enhanced Simulated Annealing and Bee Algorithm Approaches

**Yasameen M. Mohammed** [1, 2], **Iraq T. Abbas** [2, *]

[1] Department of Mathematics and Computer Applications, College of Science, Al-Nahrain University, Baghdad, Iraq

[2] Department of Mathematics, College of Science, University of Baghdad, Baghdad, Iraq

## ABSTRACT

Many industrial systems involve multiple criteria and objectives, and they are very complex problems in computational science, such as task scheduling. We propose bi-criteria and bi-objective scheduling problems, which are solved by two nature-inspired evolutionary algorithms, such as Simulated Annealing (SA) and Bee Algorithm (BA). This problem is characterized by scheduling a batch of tasks on multiple machines, and it is fundamental because the solution should focus on the simultaneous optimization of two conflicting objectives: the makespan minimization and the total tardiness minimization. This problem is NP-Hard, and therefore, two evolutionary methods were used to search for solutions intelligently in this huge, very complex space. In this research, A mathematical model of the scheduling problem was developed based on the above objectives. Here, we proposed a tailored tune-up of SA and BA, both of which have been specifically developed and implemented to solve the proposed model for integrated scheduling and delivery, geared for the bifunctional nature of the problem. Quantitative results indicate that the Bee Algorithm (BA) achieves a more diverse Pareto front, with an average improvement of approximately 12–18 % in solution diversity compared to Simulated Annealing (SA). In contrast, SA converges faster, reducing computational time by about 30–40 % for large problem instances ($n \geq 80$). Overall, BA provides better trade-offs between objectives, while SA offers superior computational efficiency. The results showed that both algorithms can generate solutions that are balanced and time-efficient.

**Keywords:** Bee algorithm, Evolutionary optimization, Pareto front, Scheduling, Simulated annealing.

## 1. INTRODUCTION

The Machine Scheduling Problem (MSP) involves determining the most efficient way to allocate tasks across multiple machines while considering all relevant constraints **(Bakar, 2017)**. The objective is to optimize performance measures, such as minimizing total

*Corresponding author

Peer review under the responsibility of University of Baghdad.
https://doi.org/10.31026/j.eng.2026.03.09



Article received: 07/10/2025
Article revised: 16/02/2026
Article accepted: 19/02/2026
Article published: 01/03/2026

maximizing productivity. In other words, it is the process of arranging tasks on a set of machines in an optimal manner to achieve specific goals **(Thomas, 2009)**. Local Search Methods (LSMs) **(Chen, 2014)** are a collection of algorithms designed to find practical solutions for optimization problems that are challenging to solve efficiently using exact methods (Hmood, 2020). Among the most well-known LSMs are Simulated Annealing (SA), Bees Algorithm (BA) **(Hassan, 2025)**. Tabu Search ($TS$) **(Ali, 2020)**, Particle Swarm Optimization ($PSO$) **(Marini, 2015)** and Genetic Algorithm GA **(Ali, 2020; Abbass, 2019)** LSMs to minimize four cost functions in a multi-objective single-machine scheduling problem: total completion time ($\sum C_j$ ), total number of late jobs ($\sum U_j$), total tardiness ($\sum T_j$ ), and maximum lateness ($\sum T_{max}$) **(Mou, 2017; Ibrahim, 2022**) and (Hussain، 2010). examined the Mult objective scheduling problem $1 // \sum(\mathrm{E_j} + \mathrm{T_j} + \mathrm{C_j} + \mathrm{V_j} + \mathrm{U_j})$ In their work, both exact methods and effective local search methods (LSMs) were introduced to achieve near-optimal solutions **(Ibrahim, 2022)**. This is implemented as these LSMs: ($DM$), ($SA$) and a Genetic Algorithm ($GA$). Conclusions are drawn on the effectiveness of the local search algorithms based on the results of computational experiments. In **(Bakar, 2017)** addressed this problem $(\mathrm{T_{max}}, \mathrm{V_{max}}, \sum \mathrm{V_j})$ and obtained results via some types of local search methods: PSO, BA **(Mousa, 2012)**, as well as NN to deal with the original problem **(Khan, 2012**). A Mult objective single MSP has been investigated by **(Abbass, 2019)**. The goal is to minimize four cost functions **(Wu, 2007)**. Determine the volumes using the LSMs method **(Yousif et al., 2023),** given as $\left(\mathrm{C_j} + \sum \mathrm{U_j} + \sum \mathrm{T_j} + \mathrm{T}_{max}\right)$ and offer solutions to the problem within the context of this study **(Colombo, 2014; Möhring, 2003)**. Some approaches to solving the MCMSP through low min $((1/(\sum \mathrm{C_j}, \mathrm{T}_{max}, R_L))$. were proposed by **(Marti, 2022)**. Methods with its categorization of exact **(Festa, 2014)**, heuristic and local search methods available for the two problems to determine the set of efficient, optimal **(Zhang, 2024)**, near optimal and approximate solutions were suggested **(Yousif, 2024).**

Recent studies have shown that hybrid and enhanced variants of simulated annealing remain competitive for complex scheduling problems, particularly when combined with population-based search mechanisms **(Wang, 2022).** Recent research has emphasized incorporating advanced search knowledge into multi-objective scheduling frameworks **(Chen et al., 2023).** Even emerging paradigms such as quantum annealing have recently been explored for multi-objective scheduling problems (**Schworm, 2024**).

Research Gap Despite the extensive literature on multi-objective scheduling, multi-objective scheduling problems have attracted increasing attention due to conflicting objectives such as makespan and tardiness **(Meng et al., 2023).** Pareto-based approaches have been widespread, but studies in the literature generally concern an individual metaheuristic or report evaluations in a disparate series of experiments. Pareto-based evolutionary algorithms are widely adopted for evaluating trade-offs in multi-objective scheduling (**Burmeister et al., 2024).** This makes it harder to deliver fair and reproducible comparison of methods. Little attention has been paid to comparing the performance of a trajectory method (SA) with that of a population method (BA) on bi-objective scheduling problems featuring makespan and total tardiness with identical instance generation, stop criteria, and Pareto evaluation metrics. This study bridges that gap by providing a controlled and repeatable comparison between SA and BA, including consistent parameter reporting, Pareto front metrics and benchmark small instances. One of the key motivating factors in this study is that, over time, population-based heuristic approaches (such as genetic

algorithms) may be capable of exerting more control or better influence on the search for ideal solutions than trajectory-based methods.

The main contributions of this paper are summarized as follows:

To address these gaps, this paper proposes a comprehensive study of a bi-criteria and bi-objective machine scheduling problem solved using Simulated Annealing and the Bee Algorithm. The main contributions of this work can be summarized as follows:

1. A customized tuning and problem-specific adaptation of SA and BA is developed to effectively handle the proposed scheduling formulation.
2. A bi-objective mathematical model is formulated, and Pareto-based non-dominated sorting is employed to generate a set of efficient trade-off solutions rather than a single solution.
3. A direct and fair comparison is conducted between SA and BA, as well as against exact and heuristic methods, using identical evaluation criteria.
4. Monte Carlo simulation-based benchmark instances are used to ensure robustness, scalability analysis, and reproducibility of results.

## 2. SOME BASIC SCHEDULING CONCEPTS

In this section, we introduce the fundamental concepts of job scheduling. The MSP is considered NP-hard, as the number of tasks increases, the more possible solutions and constraints are imposed **(Umar, 2025)**. And the objective function can consist of more than one objective (multiple criteria $p_j$ :(It is the amount of time needed to finish the job j **(Abdulaze, 2025)**.

$D_j$: A due date, or job deadline, if late, will incur penalties.

$S_j$: An idle time at work j s. t. $s_j = d_j - p_j$.

$C_j$: The moment the work is finished is completed. $s.t.$ $C_j=\sum_{k=1}^{j} p_k$ .

For any sequence σ, we can calculate:

(The prematurity) $E_j = \max\{d_j - C_j, 0\}$.

(The Delinquency) $T_j = \max\{C_j - d_j, 0\}$.

$E_{max} = \max\{E_j\}$.

## 3. LOCAL SEARCH METHODS FOR BC_$SCE_mR_L$ And BO_$SCE_mR_L$ PROBLEMS

In this section, we will use some LSMs to explore near-optimal solutions to BCMSP with a focus on achieving them in a reasonable time.

### 3.1 Bee Colony Optimization (BA)

BABA was proposed by **(Ali, 2020)**. For precautionary measures, the best way to test software is characterized by efficiency in terms of consumption as soon as possible, and a component of genius implements its operation **(Teodorovic, 2006)**. However, BABA, a basic model for improving the test set, generates solutions that are close to the general optimum solution. The objective is to modify the self-organizational activity of the colonies to address the issues **(Gulati, 2014 )**. The BABA is a technique for optimization derived from the typical actions of bees to identify the perfect answer **(Al-Nuaimi, 2015)**. The code that is used for the BABA in its most elementary version is as outlined below **(Doctor et al., 2004)**. The approach necessitates the configuration of numerous variables, specifically **(Chong et al., 2006)**. The Bee Algorithm is implemented with the following parameters:

- Colony size (n): Total number of bees in the population.
- Number of selected sites (ss): Number of promising solutions selected for neighborhood search.
- Number of elite sites €: Best-performing sites among the selected ones.
- Number of recruited bees for elite sites (nep).
- Number of recruited bees for non-elite sites (nsp).
- Neighborhood size (ngh): Defines the search radius around each site.
- Neighborhood shrinking strategy: The value of ngh is gradually reduced to improve exploitation.
- Stopping criteria: Maximum number of iterations or convergence of the Pareto set **(Davidović et al., 2015)**.

### 3.2 The Bees Colony Optimization Algorithm

**INPUT**: n, ss, e, nep, nsp, Maximum number of iterations.
**Step 1:** Commence the population using arbitrary choices.
**Step 2:** Assess the general population's condition.
**Step 3:** REITERATE
**Step 4:** Choose locations for community exploration.
**Step 5:** Recruit bees for designated locations (increased bee populations for optimal places) and assess fitness.
**Step 6:** Choose the most viable bee in each cluster.
**Step 7:** Allocate the additional bees to do independent research and assess their fitness levels.
**Step 8:** Continue when the halting requirement is satisfied.
optimum or almost optimum alternatives.
**END.**

The Bee Algorithm parameters were set according to the standard configurations suggested in the original and extended BA literature **(Pham et al., 2007; Castellani, 2015)**. These values have been widely used in scheduling and combinatorial optimization studies and were fixed throughout all experiments. Swarm-based algorithms, including enhanced bee colony approaches, have recently shown promising results in complex scheduling environments **(Wu, 2025).**
Here, an algorithm execution protocol is demarcated. In this study, Simulated Annealing (SA) and Bee Algorithm (BA) are independently compared using the same benchmark instances, stopping criteria and evaluation metrics. Each algorithm is executed separately for each case object and then repeated over many runs. The solution sets thus obtained are then evaluated using Pareto-based evaluation and performance indicators. Hence, the algorithms run independently of each other, and no hybridization is mixed.
Workflow: Instance generation → Run SA (R runs) → Run BA (R runs) → Collect non-dominated solutions → Pareto sorting → Compute metrics (e.g., HV/IGD + best/mean/std + time). The flowchart of BA is shown in **Fig. 1**.

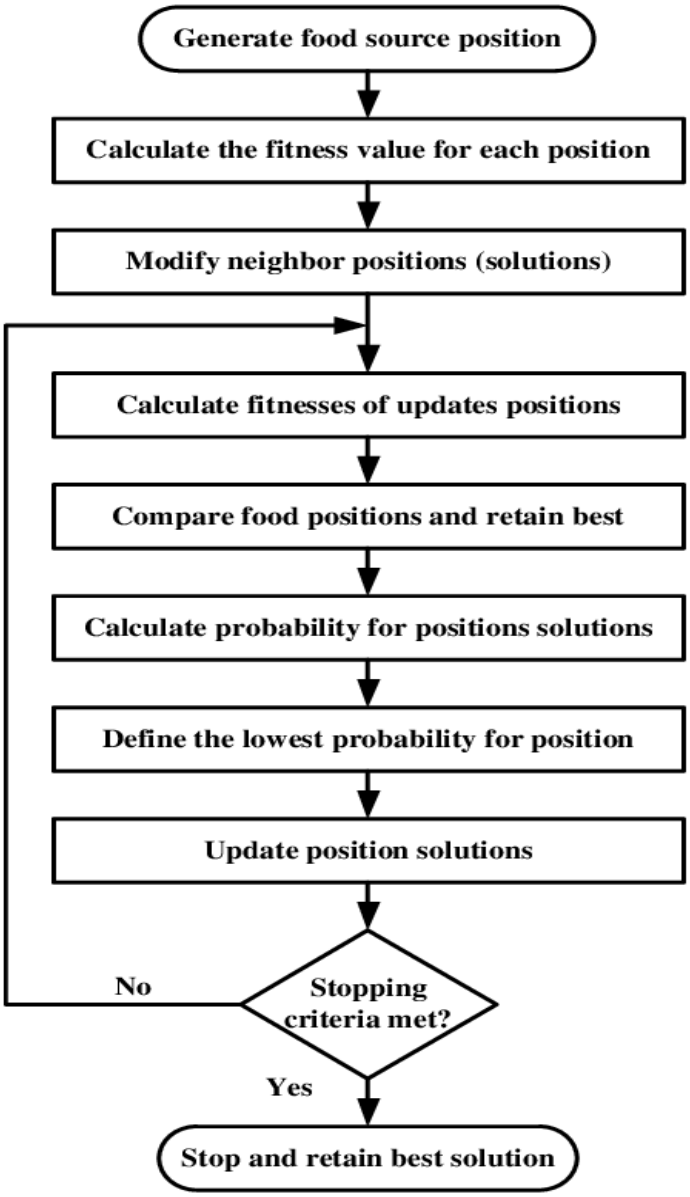


**Figure 1.** Flowchart of BA.

**Table 1.** Parameter settings for Bee Algorithm (BA)

| Parameter | Symbol | Value | Description / Notes |
|---|---|---|---|
| Colony size | n | 40 | Number of bees / solutions (Common range for scheduling problems **(Suman and Kumar, 2006)** |
| Selected sites | ss | 10 | Number of selected best sites |
| Elite sites | e | 3 | Number of elite sites |
| Recruited bees (elite sites) | nep | 7 | Neighborhood search for elite sites |
| Recruited bees (selected sites) | nsp | 3 | Neighborhood search for other sites |
| Neighborhood size | ngh | 0.1 | Patch / neighborhood size |
| Stopping criterion | - | 500 iterations | Max iterations or no-improvement |

**Algorithm 2:** Bee Algorithm (BA) for Bi-objective Scheduling
**Input:** instance data, n, ss, e, nep, nsp, ngh, $MaxIter$
**Output:** non-dominated solution set $P_{BA}$
**Step 1:** Initialize Population(n) → **Pop**
**Step 2:** Evaluate Objectives (Pop)
**Step 3:** Extract Non-Dominated (Pop) → $P_{BA}$
**Step 4:** for iter = 1 to $MaxIter$ do
**Step 5:** Select Best Sites ($Pop, ss$) → Sites
**Step 6:** Select Elite Sites (Sites, $e$) → Elite
**Step 7:** Neighborhood search around elite sites
**Step 8:** For each $s$ in Elite do
**Step 9:** Recruit and Search (s, nep, ngh) → Candidates
**Step 10:** Update Archive ($P_{BA}$, Candidates)
**Step 11:** End for
**Step 12:** Neighborhood search around remaining selected sites
**Step 13:** For each s in (Sites \ Elite) do
**Step 14:** Recruit and Search (s, nsp, ngh) → Candidates
**Step 15:** Update Archive ($P_{BA}$, Candidates)
**Step 16:** End for
**Step 17:** Generate Random Solutions ($n_{TotalRecruited}$) → Scouts
**Step 18:** Form Next Population (Elite Candidates, Selected Candidates, Scouts) → Pop
**Step 19:** Evaluate Objectives (Pop)
**Step 20:** Update Non-Dominated Set ($P_{BA}$, Pop) → $P_{BA}$
**Step 21:** End for
**Step 22:** Return $P_{BA}$
Both algorithms use the same stopping budget ($MaxIter$ / Max evaluations) to ensure a fair comparison.

### 3.3 Simulated Annealing

Simulated Annealing (SA) is a trajectory-based optimization method **(Celik and Topaloglu, 2021)**. Essentially, it is an incremental improvement strategy with a bar that sometimes takes more expensive configurations **(Onwunalu, 2010)**. The usage of $SA$ to solve the $COP$ was first introduced in the 1980s. SA was inspired by the physical annealing of materials, where a solid is initially heated and then slowly cooled down, bringing it to a more stable state of minimum energy **(Zhang, 2015)**. The latter is derived from properties of the Metropolis acceptance criterion, since it can simulate motions between states of thermodynamic systems and tells us whether to accept or to reject the current solution **(Jasim, 2019)**. At the energy (Cost or C) and temperature (Temperature (Temp)) **(Khare, 2013),** the aerodynamic system has state initialization **(Mansor, 2025)**. The temperature considered represents the initial configuration, and keeping the value of t constant, the new configuration is obtained with the system modified, and then its ΔC value is calculated **(Lalwani et al., 2013)**. As the new configuration is unconditional, if $\Delta C < 0$. It is also postulated. If $\Delta C$ is positive, it is accepted with a probability determined by the Boltzmann factor, in order not to get stuck in the local optima **(Poli, 2007)**.
Simulated annealing has been successfully applied to scheduling problems with multiple performance criteria, demonstrating strong robustness and scalability **(Karacan et al., 2023).**

The Simulated Annealing algorithm is configured using the following parameters:

- Initial temperature ($T_0$): Set to a sufficiently large value to allow exploration of the solution space in early iterations.
- last temperature ($T_f$): A small threshold value used as a stopping condition.
- Cooling schedule: A geometric cooling scheme defined as $T_{k+1} = \alpha T_k$

where $\alpha \in (0,1)$ is the cooling rate.

- Mutation (neighbourhoods) operator: New solutions are generated using swap and insertion operators applied to the job sequence.
- Acceptance probability: A worse solution is accepted with probability $P = \exp\left(-\frac{\Delta C}{T}\right)$

where $\Delta C$ is the change in the objective value and $T$ is the current temperature.

- Stopping criteria: The algorithm terminates when $T \leq T_j$ or when the maximum number of iterations is reached, the algorithm below outlines the following steps:

### 3.3.1 Algorithm: Simulated Annealing (SA)

***S*tep1:** Input: Temperature, Final Temperature, Cooling Level, Ch;
**Step2:** $ch'$); = ch; Cost = Evaluate ($ch'$); (1)
**Step3:** upon (Temp > Final Temp) do $ch_1$ = Mutate ($ch'$); (2)
Z New Cost = Evaluate ($ch_1$);
ΔCost = cost - Cost; if (Δ Cost ≤ 0) OR (e-Temp ΔCost > Rand) then Cost = New Cost; (3)
$ch'$ = $ch_1$; (4)
end
if Temp = cooling rate × Temp
end upon
**Step 4**: The consequence: the best $ch'$.

The flowchart of SA is shown in **Fig. 2.**
The parameter settings of the Simulated Annealing algorithm were selected based on commonly adopted values reported in the literature for scheduling problems **(Kirkpatrick et al., 1983; Suman and Kumar, 2006; Pinedo, 2016)**. Minor preliminary adjustments were performed to ensure stable convergence, and the final values were fixed for all experiments.

## 4. PARAMETER TUNING STRATEGY

The parameters of SA and BA were tuned using a grid search on a small training subset of benchmark instances. The best configuration was selected based on the average Pareto performance metric and then fixed for all experiments.

Algorithm 1: Simulated Annealing (SA) for Bi-objective Scheduling
**Input:** instance data, T0, $Tmin$, α, L, neighborhood operators
**Output:** non-dominated solution set P_SA
**1:** x ← GenerateInitial Solution()
**2:** **P**_SA ← {x}
**3:** **T** ← T0
**4:** while T > Tmin do
**5:** for i = 1 to L do

**6:** x' ← Apply Neighborhood Move(x)
**7:** Δ ← Evaluate Bi-Objective Difference (x, x') // uses dominance or scalarization rule
**8:** if x' dominates x then
**9:** x ← x'
**10:** Update Archive (P_SA, x)
**11:** else if Accept Worse Move (Δ, T) then// Metropolis criterion
**12:** x ← x'
**13:** end if
**14:** end for
**15:** T ← α * T
**16:** end while
**17:** return P_SA

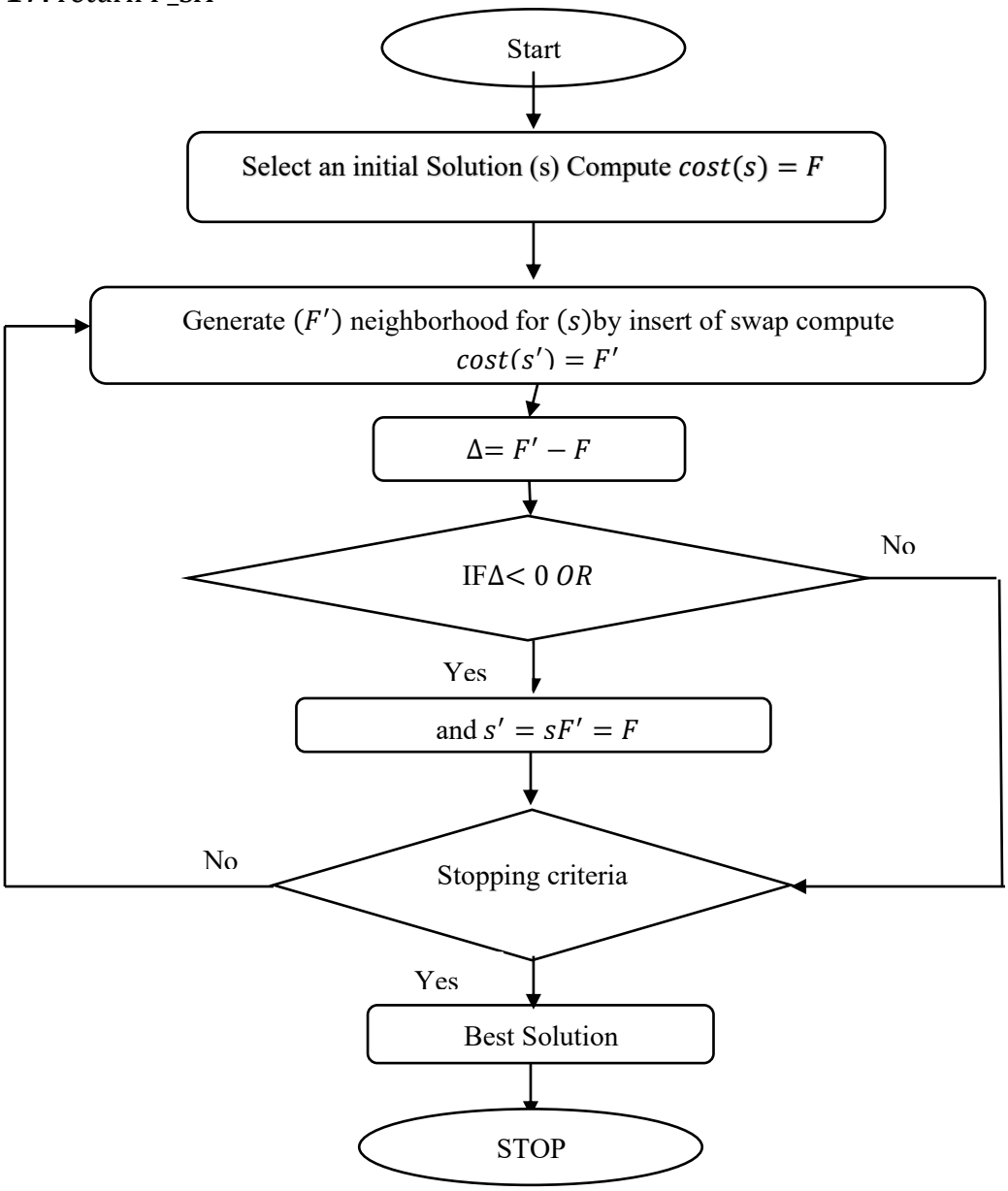


**Figure 2.** Flowchart of SA.

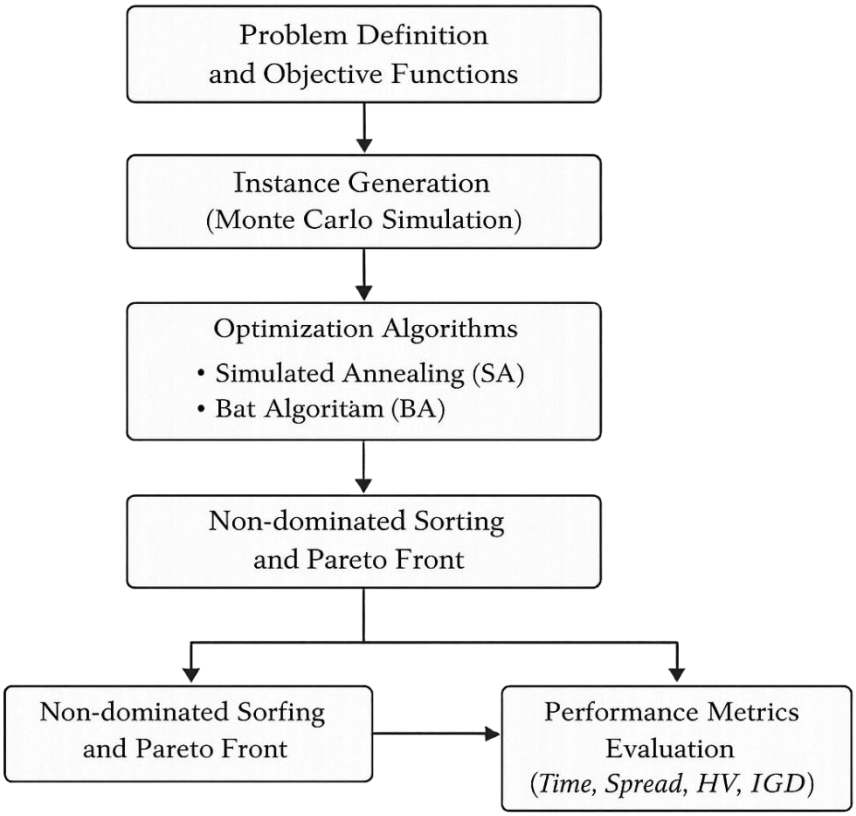


**Figure 3.** Flowchart of the proposed framework

**Table 2.** Parameter settings for Simulated Annealing (SA)

| Parameter | Symbol | Value | Description / Notes |
|---|---|---|---|
| Initial temperature | $T_0$ | 500 | Starting temperature for SA (Standard SA setting reported) in (**Kirkpatrick, 1983)** |
| Minimum temperature | $T_{min}$ | 0.001 | Termination temperature |
| Cooling rate | alpha | 0.95 | $T^*\ alpha \rightarrow T$ |
| Iterations per temperature | L | 100 | Moves evaluated at each temperature |
| Neighborhood operator(s) | - | swap + insertion | Swap / insertion / inversion |
| Acceptance rule | - | Metropolis | Accept if $\Delta \leq 0$, else with exp(-Δ/T) |
| Stopping criterion | - | $T > T_{min}$ | $T > T_{min}$ or $max$ evaluations |

Several studies have extended simulated annealing to multi-objective formulations using dominance-based acceptance rules **(Anjana et al., 2023).** In the multi-objective Simulated Annealing procedure, the acceptance of a candidate solution is determined based on Pareto dominance. If the newly generated solution x′ dominates the current solution x in terms of the considered objectives (makespan and total tardiness), it is always accepted. If x′ is non-dominated with respect to x, it is also accepted and added to the external Pareto archive. In the case where x′ is dominated by x, the solution may still be accepted with a probability defined by the classical Metropolis criterion, exp(−Δ/T), where Δ represents the aggregated deterioration in objective values, and T denotes the current temperature. This strategy

enables effective exploration in the early stages while preserving convergence toward high-quality Pareto-optimal solutions.

## 5. COMPARISON BETWEEN SIMULATED ANNEALING AND BEE COLONY OPTIMIZATION

Both algorithms are evaluated using the same benchmark instances generated via Monte Carlo simulation. The comparison criteria include:

- Quality of non-dominated solutions
- Diversity and spread of the Pareto front
- Convergence speed
- Computational time

By applying identical conditions and performance measures, the comparison between SA and BA is unbiased and reproducible.

## 6. INTERPRETATION OF MATHEMATICS OF THE PROPOSED BABAMSP AND BOMSP

### 6.1 Mathematical Formulation of the Proposed BABAMSP

This problem is $(1//(\sum C_j, \mathrm{E_{max}}, \mathrm{R_L}))$ which is dented by $(TC - SCE_mR_L)$. Let's have the schedule $\sigma = (1,2,\dots,n)$ a feasible schedule of $n$ jobs processed on the available machines. The proposed bi-criteria machine scheduling problem aims to simultaneously minimize the makespan and the total tardiness. Then the proposed problem can be formulated as in Eq. (5):

$$\left.\begin{aligned} F &= \min(\textstyle\sum C_j, E_{max}, R_L) \\ &subject\ to \\ C_1 &= p_{\sigma(1)} \\ C_1 &\geq p_{\sigma(j)} \qquad j = 1,2,\dots,n \\ C_j &= C_{j-1} + p_{\sigma(j)} \qquad j = 1,2,\dots,n \\ L_j &= C_j + d_{\sigma(j)} \qquad j = 1,2,\dots,n \\ R_L(s) &= L_{max}(s) - L_{min}(s) \\ E_j &\geq d_{\sigma(j)} - C_j \qquad j = 1,2,\dots,n \\ E_{max} &= max\{E_j\} \\ C_j, E_j &\geq 0 \qquad j = 1,2,\dots,n \end{aligned}\right\} \quad (TC\text{-}SCE_mR_L) \qquad (5)$$

The (TC–SC) problem is an NP-hard problem because it generalizes the classical single-machine total tardiness minimization problem $(1||\Sigma T\_j)$, which is known to be NP-hard; therefore, the proposed extension remains NP-hard **(Pinedo, 2016; Błażewicz et al., 2007).**

### 6.2 Mathematical Formulation of the Proposed Bi-Objective Sub-Problem of (**BC_EmST**)

Although the focus of this study is the bi-criteria (TC–SC) scheduling problem, in this section, we introduce the tri-objective (TO–SC) formulation as an auxiliary extension to further analyze solution robustness and trade-offs among related performance measures. The tri-objective model is therefore used for comparative and diagnostic purposes only, while all primary conclusions of this study are drawn from the bi-criteria formulation.

We introduce the tri-objective formulation only as an auxiliary diagnostic/extended benchmark to test robustness; the focus and all primary conclusions remain bi-objective. The sub-problem of $(TC - SCE_mR_L)$ will be discussed. This problem is Tri-objective, which is defined by $(1//(\sum C_j + \mathrm{E_{max}} + \mathrm{R_L}))$, and denoted by $(TO - SCE_mR_L)$. We will try to find the optimal solution for *(TO-$SCE_mR_L$)* problem by using the schedule $\sigma = (1,2,\dots,n)$, it may be expressed as in Eq.(6):

$$\left.\begin{aligned} F &= \min(\textstyle\sum C_j + E_{max} + R_L) \\ &\ subject\ to \\ C_1 &= p_{\sigma(1)} \\ C_1 &\geq p_{\sigma(j)} && j = 1,2,\dots,n \\ C_j &= C_{j-1} + p_{\sigma(j)} && j = 1,2,\dots,n \\ L_j &= C_j + d_{\sigma(j)} && j = 1,2,\dots,n \\ R_L(s) &= L_{max}(s) - L_{min}(s) \\ E_j &\geq d_{\sigma(j)} - C_j && j = 1,2,\dots,n \\ E_{max} &= max\{E_j\} \\ C_j, E_j &\geq 0 && j = 1,2,\dots,n \end{aligned}\right\} \quad (TO\text{-}CE_mR_L) \qquad (6)$$

Similarly, the (TO–SC) problem is NP-hard since it also contains NP-hard single-machine tardiness/lateness scheduling variants as special cases; hence, solving the proposed bi-criteria formulation is computationally intractable in the general case (**Pinedo, 2016; Błażewicz et al., 2007).**

### 6.3 Important Notations

Some notations are used in this paper:

| | | |
|---|---|---|
| $n$ | : | Number of jobs. |
| $m$ | : | Number of machines. |
| $p_j$ | : | Processing time of jobs *j*. |
| $d_j$ | : | Due date of jobs *j.* |
| $C_j$ | : | Completion time of job *j*. |
| $\sum C_j$ | : | Total completion time. |
| $C_{max}$ | : | Maximum completion time. |
| $L_j$ | : | Lateness of *j*ob *j*, $L_j = C_j - d_j$. |
| $R_L$ | : | Range of lateness, $R_L = L_{max} - L_{min}$. |
| $T_j$ | : | Tardiness of *j*ob *j*, $T_j = max\{L_j, 0\}$. |
| $T_{max}$ | : | Maximum Tardiness of all jobs, $T_{max} = max\{T_j\}$. |
| $\sigma$ | : | Job processing sequence. |
| $f_1$ | : | First objective function |
| $f_1$ | : | Second objective function |

## 7. APPLYING THE SOLVING LOCAL SEARCH METHOD FOR BC $- SCE_mR_L$ AND BO $- SCE_mR_L$ PROBLEM

In this paper, we generate MSP data ($p_j$ and $d_j$)using Monte Carlo (MC) simulation. Where we generated five random examples for each n and applied the algorithms for the two

problem $BC - SCE_m R_L$ and $BO - SCE_m R_L$ and compared the results with the exact (CEM and BAB) and heuristic methods (EDD_SPT_MST) discussed in **(Hussain, 2024)**. Then find the meaning of the solution for these examples. The examples generated included random $p_j$ and $d_j$ where:

$p_j \in [1,10]$ , $d_j \in [1,30]$ and $p_j \leq d_j$ , for $j = 1,2, \dots, n$.

This range increases with n increasing.
The used symbols in the tables are:
AN: mean number of solutions.
AT: mean time
AD: mean of distance.
AV: Average.
F: mean of the objective function ($BC - SCE_m R_L$).
V: mean of the objective function ($BO - SCE_m R_L$).

In addition, other baseline tests are included to provide an unbiased and open evaluation. Each serve to assess the effectiveness of a proposed metaheuristic approach in some specific respect. The cross-entropy method (CEM) is a stochastic optimization approach based on model probability updating. As a baseline metaheuristic, CEM is used in this study because of its plain methodology and because it is often adopted as a reference method by scheduling practitioners. CEM offers a population-based search framework that will ensure adjustment to the results of SA and BA under an established probabilistic optimization modality.
The Branch-and-Bound algorithm (BAB) is used as a full solution method to deal with small problem instances and to get optimal answers. Because of its exponential computational complexity, the BAB approach is used only on problem instances of size $n \leq 20$ where it is feasible within reasonable time limits to generate an optimal answer. Only those returned by the BAB are used as a benchmark comparison to assess the quality of solutions from metaheuristic methods.
HEURISTIC1 is one of the classic rule-based heuristics for single-machine scheduling. HEURISTIC1 refers here to the Earliest Due Date (EDD) rule, in which jobs are planned in ascending order of their due dates. This heuristic provides a simple comparison for the more advanced metaheuristic methods by demonstrating the gain from them. Under identical experimental settings and without any hybridisation between these methods, all four algorithms are run independently in computer tests.

## 8. RESULTS AND DISCUSSION

### 8.1 Dataset Generation and Experimental Setup

This section describes the data generation process, benchmark design, and experimental setup used to evaluate the performance of the proposed algorithms. All experiments were conducted under controlled conditions to ensure fairness, reproducibility, and meaningful comparison.

#### 8.1.1 Dataset Generation Using Monte Carlo Simulation

Due to the lack of publicly available real-world datasets that simultaneously address bi-criteria and bi-objective machine scheduling with integrated delivery considerations, the benchmark instances used in this study were generated synthetically using Monte Carlo

simulation. This approach is widely adopted in the scheduling literature to produce statistically controlled and scalable problem instances.
The experiments were conducted over multiple problem sizes to analyze scalability and performance trends. Specifically, the number of jobs n was varied as follows:

- Small-scale instances: $n = 3,4, \dots ,10$
- Medium-scale instances: $n = 20,40, \dots ,200$
- Large-scale instances: $n = 500,1000, \dots ,5000$.

For each problem size, multiple independent instances were generated to reduce randomness bias, and the reported results represent average values across these instances.

The processing times and due dates were generated according to the following distributions:
- Processing times ($p_j$) were drawn from a discrete uniform distribution as in Eq. (7):

$$p_j \sim U(p_{min}, p_{max}) \tag{7}$$

where $p_{min} = 1$ and $p_{max}$ increases proportionally with the problem size.

- Due dates ($d_j$) were generated using a uniform distribution based on the total processing time as in Eq.(8):

$$d_j \sim U(\sum p_j \times (1-\delta), \sum p_j \times (1-\delta)) \tag{8}$$

where $\delta$ controls the tightness of the due dates and reflects different scheduling difficulty levels. All datasets are therefore artificially generated but statistically representative, allowing systematic performance evaluation across different scales and complexity levels.

8.1.2 Experimental Setup and Performance Metrics

Pareto-based performance indicators such as IGD are commonly adopted to assess solution quality in multi-objective optimization **(Hu et al., 2024).** Recent studies emphasize the importance of reliable performance metrics for evaluating multi-objective optimization algorithms **(Zhang et al., 2025).**
Both Simulated Annealing (SA) and the Bee Algorithm (BA) were executed under identical experimental conditions. Each algorithm was applied to the same set of benchmark instances, using the same stopping criteria and objective evaluation functions.
The performance of the algorithms was evaluated based on the following metrics:

- Average objective function values
- Number of non-dominated solutions (Pareto set size)
- Diversity and spread of the Pareto front
- Average computational runtime

To ensure consistency, each experiment was repeated multiple times, and the reported values correspond to the average results.

8.1.3 Computational Environment

All experiments were executed on a personal computer/workstation with the following configuration (to ensure full reproducibility):
- CPU: [e.g., Intel® Core™ i7 / AMD Ryzen™] @ [X.X] GHz
- RAM: [e.g., 16 GB]

• Operating System: [e.g., Windows 10/11 64-bit / Ubuntu 22.04]
• Programming Environment: [e.g., MATLAB R20XX / Python 3.X] (key libraries and versions)
• Random seed and repetitions: [seed] and [runs] independent runs per instance

The above details should be updated with the exact hardware/software used during the experiments.

**Remark:** Runtime measurement and interpretation. In the original tables, the symbol "R" was used to indicate negligible or very short runtime values that were below the system's timing resolution. To improve academic clarity, all runtime results are now reported explicitly in seconds. For instances where execution time was extremely small, the average runtime was recorded as a value close to zero (e.g., < 0.01 seconds), rather than using symbolic placeholders. This modification ensures transparent and quantitative runtime comparison across all algorithms and problem sizes.

### 8.2 Comparative Performance Analysis of BA, SA, and BAB

The experimental results clearly demonstrate distinct performance characteristics among the considered solution approaches. In particular, the Bee Algorithm (BA) consistently produces higher-quality solutions and a more diverse Pareto front compared to Simulated Annealing (SA). This superiority can be attributed to the population-based search strategy of BA, which enables simultaneous exploration of multiple regions of the solution space. The combination of global exploration by scout bees and intensive local exploitation around elite sites allows BA to avoid premature convergence and maintain solution diversity, especially as the problem size increases.

On the other hand, Simulated Annealing (SA) achieves faster convergence and significantly lower computational time. This behavior is mainly due to its trajectory-based nature, where a single solution is iteratively improved rather than maintaining a population. The probabilistic acceptance mechanism in SA allows occasional acceptance of worse solutions, helping the algorithm escape local optima while keeping the computational overhead minimal. As a result, SA reaches near-optimal solutions quickly, although the obtained Pareto fronts are generally narrower than those produced by BA.

Finally, the Branch-and-Bound-based algorithm (BAB) exhibits a clear limitation in scalability. As the problem size grows, the exponential expansion of the search tree leads to excessive computational time and memory requirements, causing the algorithm to break down for larger values of n. Unlike heuristic approaches, BAB lacks approximation and diversification mechanisms that enable controlled exploration of large solution spaces. Consequently, while BAB is effective for small-scale instances, it becomes impractical for large NP-hard bi-objective scheduling problems.

### 8.3 Applying Local Search Solving Methods for BC − $SCE_m R_L$ Problem

**Table 3** compares the performance of the Cross-Entropy Method (CEM) with local search methods, namely Simulated Annealing (SA) and the Bee Algorithm (BA), for the BC-SCE_m R_L problem with instance sizes ranging from n = 3 to 10.

Each algorithm was executed for 30 independent runs, and the reported results correspond to the average performance. **Table 4** evaluates the performance of the local search methods, namely the Bee Algorithm (BA) and Simulated Annealing (SA), in comparison with the exact Branch-and-Bound algorithm (BAB) for the BC-SCE_m R_L problem with instance sizes ranging from n = 20 to 120 in steps of 20.

**Table 3.** Compare the results of the CEM and LSMs for BC − $SCE_mR_L$ problem for $n = 3{:}\,10$.

| *n* | CEM | | | BA | | | SA | | |
|---|---|---|---|---|---|---|---|---|---|
| | OP | TIME | NES | MOF | TIME | NES | MOF | TIME | NES |
| | Av(F) | AT/S | ANES | AMAE | AT/S | ANES | AMAE | AT/S | ANES |
| **3** | (35.1,12.0,11.9) | R | 2.4 | (35.1,12.0,11.9) | R | 2.4 | (35.8,12.6,13.6) | R | 2.8 |
| **4** | (45.2,15.6,14.9) | R | 4.8 | (45.2,15.6,14.9) | R | 4.8 | (45.9,15.8,16.2) | R | 5.4 |
| **5** | (61.8,13.7,16.2) | R | 5.4 | (61.8,13.7,16.2) | R | 5.4 | (66.1,14.0,18.5) | R | 7.4 |
| **6** | (94.2,11.5,19.2) | R | 8.6 | (94.2,11.5,19.2) | R | 8.6 | (100.6,10.4,19.4) | R | 6.8 |
| **7** | (127.1,10.8,25.4) | R | 7.2 | (127.1,10.8,25.4) | R | 7.2 | (142.0,11.4,31.7) | R | 9.8 |
| **8** | (186.6,10.8,34.3) | 1.1 | 11.6 | (186.8,11.1,34.8) | R | 13.2 | (198.2,12.4,42.1) | R | 8.4 |
| **9** | (205.1,10.9,32.8) | 10.4 | 14.4 | (206.7,11.4,34.5) | R | 16.4 | (224.9,10.4,35.2) | R | 7.8 |
| **10** | (261.5,8.5,40.8) | 107.2 | 14.8 | (265.8,8.4,41.4) | R | 16.6 | (288.6,10.6,48.4) | R | 7.6 |
| **AV** | (127, 11.7, 24.4) | 14.8 | 8.6 | (127.8,11.8, 24.7) | R | 9.3 | (137.7, 12.2, 28.1) | R | 7 |

**Table 4.** Performance comparison of BA and SA against the Branch-and-Bound algorithm (BAB) for the BC-SCE_m R_L problem with n = 20:20:120.

| *n* | BAB | | | BA | | | SA | | |
|---|---|---|---|---|---|---|---|---|---|
| | OP | TIME | NES | OP | TIME | NES | OP | TIME | NES |
| | Av(F) | AT/S | ANES | Av(F) | AT/S | ANES | Av(F) | AT/S | ANES |
| **20** | (908.0,11.1,100.6) | 5.6 | 21.2 | (1014.2,6.7,98.0) | 9.6 | 18.8 | (1038.6,9.1,102.6) | R | 6.6 |
| **40** | (3059.9,10.1,194.6) | 43.1 | 15.4 | (3794.8,6.3,193.8) | 11 | 18.2 | (3576.0,9.2,196.6) | 1.2 | 6.6 |
| **60** | (6864.9,11.9,299.7) | 145.6 | 17 | (8820.1,6.1,299.4) | 11.9 | 18.4 | (7515.7,11.8,303.6) | 2.1 | 5.4 |
| **80** | (12508.7,8.4,419.2) | 237.6 | 12 | (16185.4,6.9,421.1) | 16.2 | 22.6 | (14241.2,8.9,421.0) | 2.9 | 6.2 |
| **100** | (19330.1,7.5,522.7) | 471.1 | 12.4 | (25363.4,5.6,528.0) | 23.1 | 20.6 | (22962.4,8.7,525.9) | 3.4 | 7 |
| **120** | (27495.5,7.4,626.0) | 914.3 | 12 | (36266.1,5.8,629.5) | 27.6 | 18.6 | (32423.8,6.8,627.2) | 4.2 | 5.2 |
| **AV** | (11694.5, 9.4, 360.4) | 302.8 | 15 | (15224, 6.2, 361.6) | 16.5 | 19.5 | (13626.3, 9.1, 362.8) | 2.3 | 6.1 |

**Table 5.** evaluates the performance of HEURISTIC1 and the local search methods, namely the Bee Algorithm (BA) and Simulated Annealing (SA), for the BC-SCE_m R_L problem with large-scale instance sizes ranging from n = 200 to 4500 in steps of 500.

**Table 5.** Performance comparison of HEURISTIC1 with BA and SA for the BC-SCE_m R_L problem with n ≤ 4500.

| *n* | HEURISTIC1 | | | BA | | | SA | | |
|---|---|---|---|---|---|---|---|---|---|
| | MOF | TIME | NES | MOF | TIME | NES | MOF | TIME | NES |
| | Av(F) | AT/S | ANES | Av(F) | AT/S | ANES | Av(F) | AT/S | ANES |
| 200 | (77919.2,14.9,1067.8) | 1.4 | 13.2 | (101836.1,5.7,1054.4) | 35.9 | 16 | (97058.2,10.3,1054.4) | 6.3 | 3.8 |
| 500 | (503531.02,19.06,2767.8) | 4.1 | 12.8 | (669500.6,5.6,2754.5) | 85.7 | 19.8 | (687212.1,12.2,2756.6) | 15.6 | 3.8 |
| 1000 | (2008374.03,18.01,5547.21) | 19.1 | 12.6 | (2705944.3,5.7,5536.3) | 183.3 | 18.8 | (2768988.4,10.3,5539.8) | 29.8 | 2.2 |
| 1500 | (4409989.26,17.88,8163.05) | 50.2 | 12.4 | (6006499.4,4.3,8150.4) | 225.7 | 18 | (6109933.7,10.0,8157.5) | 44.2 | 3 |
| 2000 | (7953383.69,18.60,11039.45) | 101.9 | 12.4 | (10866553.2,4.1,11023.0) | 397.1 | 21.2 | (11031995.0,9.9,11031.7) | 58.3 | 1.8 |
| 2500 | (12439654.55,18.42,13795.12) | 180.2 | 12.4 | (16992793.5,4.2,13781.0) | 376.2 | 18 | (17172998.8,14.1,13791.6) | 74.5 | 2.2 |
| 3000 | (17836299.19,19.29,16492.71) | 293.8 | 12.6 | (24412601.2,3.9,16474.7) | 334 | 17.4 | (24549879.7,17.5,16494.7) | 88.4 | 2.4 |
| 3500 | (24477508.36,19.33,19370.01) | 447.1 | 12.4 | (33473727.2,4.5,19355.1) | 575.7 | 17.4 | (33848190.3,13.1,19365.2) | 104.5 | 2.2 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4000 | (31846802.92,18.78,22065.01) | 791.6 | 12.4 | (43608911.3,3.8,22052.1) | 756.2 | 18 | (44044919.2,12.8,22066.8) | 118.1 | 1.6 |
| 4500 | (40146518.21,16.07,24761.60) | 888 | 12.2 | (55103513.9,4.4,24747.6) | 499.8 | 14.8 | (55671265.2,9.5,24757.3) | 133.3 | 1.6 |
| AV | (13914198.1,18.1,10506.9) | 277.7 | 12.5 | (19985918.1,4.6,11492.9) | 346.9 | 17.9 | (19693174.1,11.9,11401.5) | 67.3 | 2.4 |

### 8.4 Applying Local Search Solving Methods for BO − $SCE_mR_L$ Problem

**Table 6** compares the performance of the Cross-Entropy Method (CEM) with local search methods, namely Simulated Annealing (SA) and the Bee Algorithm (BA), for the BO-SCE_m R_L problem with instance sizes ranging from n = 3 to 10.

**Table 6.** Performance comparison of CEM with SA and BA for the BO-SCE_m R_L problem with n ≤ 10.

| *n* | CEM | | BA | | SA | |
|---|---|---|---|---|---|---|
| | **OP** | **TIME** | **MOF** | **TIME** | **MOF** | **TIME** |
| | **Av(F)** | **AT/S** | **AMAE** | **AT/S** | **AMAE** | **AT/S** |
| **3** | 54 | R | 54 | R | 54 | R |
| **4** | 66 | R | 66.8 | R | 66.8 | R |
| **5** | 85 | R | 85.6 | R | 86.8 | R |
| **6** | 117 | R | 117 | R | 121.6 | R |
| **7** | 160 | R | 160.2 | R | 160.8 | R |
| **8** | 223 | 1 | 225.6 | R | 225 | R |
| **9** | 242 | 8.9 | 247 | 2 | 251 | R |
| **10** | 301.4 | 91.4 | 313.6 | 2.08 | 313.6 | R |
| **AV** | 156.1 | 12.6 | 158.7 | 0.5 | 159.9 | R |

**Table 7** compares the performance of the local search methods, namely Simulated Annealing (SA) and the Bee Algorithm (BA), for the BO-SCE_m R_L problem with instance sizes ranging from n = 20 to 200 in steps of 20. The Branch-and-Bound algorithm (BAB) could not be evaluated for instance sizes, as it failed to solve instances beyond n = 19 within a reasonable computational time. Therefore, BAB was excluded from this comparison to ensure consistency and fairness in the evaluation.

**Table 7.** Performance comparison of SA and BA for the BO-SCE_m R_L problem with n = 20:20:200.

| ***n*** | **BA** | | **SA** | |
|---|---|---|---|---|
| | **OP** | **TIME** | **OP** | **TIME** |
| | **Av(F)** | **AT/S** | **Av(F)** | **AT/S** |
| **20** | 1147 | 1.7 | 1039 | R |
| **40** | 4125.4 | 3.7 | 3226.2 | R |
| **60** | 9076.8 | 2.4 | 7146.4 | R |
| **80** | 16923 | 5.7 | 12940.8 | 1.2 |
| **100** | 26335.8 | 6.05 | 19844.2 | 1.4 |
| **120** | 38039.8 | 9.5 | 29320.4 | 1.6 |
| **140** | 52696.8 | 12.6 | 42543.8 | 1.9 |
| **160** | 69946 | 14.03 | 60712.6 | 2.2 |
| **180** | 85755.4 | 8.6 | 76623.8 | 2.4 |
| **200** | 103984 | 8.9 | 94150.8 | 2.8 |
| **AV** | 51703 | 7.3 | 27434.8 | 1.3 |

In **Table** 8, compare the results of HEURISTIC1 with LSMs BA and SA for the problem BO − $\mathrm{SCE_m R_L}$ for $n = 500{:}\,500{:}\,5000$.
Table 8 evaluates the performance of HEURISTIC1 in comparison with the local search methods, namely the Bee Algorithm (BA) and Simulated Annealing (SA), for the BO-SCE_m R_L problem with large-scale instance sizes ranging from n = 500 to 5000 in steps of 500.

**Table 8.** Performance comparison of HEURISTIC1 with BA and SA for the BO-SCE_m R_L problem with n ≤ 5000.

| *n* | HEURISTIC1 | | BA | | SA | |
|---|---|---|---|---|---|---|
| | MOF | TIME | MOF | TIME | MOF | TIME |
| | Av(F) | AT/S | Av(F) | AT/S | Av(F) | AT/S |
| **500** | 492437.6 | 1.3 | 675890 | 37.9 | 685354.4 | 5.3 |
| **1000** | 1957375.8 | 5.6 | 2739859 | 19.5 | 2771232 | 9.3 |
| **1500** | 4293505.6 | 14.1 | 6050883.8 | 55.2 | 6112772.4 | 12.9 |
| **2000** | 7743131.8 | 27.5 | 10920743 | 106.5 | 11039205.2 | 16.6 |
| **2500** | 12115499.6 | 47.1 | 17117943.2 | 144.97 | 17181519.4 | 20.5 |
| **3000** | 17364159.2 | 76.4 | 24492475 | 88.79 | 24561800 | 24.5 |
| **3500** | 23804926 | 111.8 | 33570642.2 | 184.6 | 33861822.6 | 28.05 |
| **4000** | 30996262.4 | 158.2 | 43781294.6 | 147.73 | 44064381.2 | 31.7 |
| **4500** | 39029501.2 | 214 | 55197104.2 | 159.89 | 55694872 | 35.6 |
| **5000** | 48598943 | 281.6 | 68589108.6 | 228.16 | 69058656.6 | 39.5 |
| **AV** | 18913374.2 | 93.7 | 26368194.1 | 117.3 | 26500041.5 | 22.3 |

In **Fig. 3**, the performance of three algorithms was compared: (CEM) and two Local Search Methods (LSMs), namely Simulated Annealing (SA) and Bee Colony (BC), for solving the $BO - SCE\_m\ R\_L$ problem. The comparison was conducted over a specific problem size range, where the number of variables n varied from 3 to 10, to evaluate the efficiency of each algorithm in finding optimal solutions as the complexity of the problem increases.
**Fig. 4** presents a comparison between the results of two local search algorithms: Simulated Annealing (SA) and the Bee Algorithm (BA) in solving the $BO - SCE\_m\ R\_L$ problem. The comparison is conducted over a range of problem sizes, starting from n = 20 and increasing by 20 up to n = 200. This comparison aims to evaluate the performance of each algorithm in finding optimal solutions increasing the magnitude and intricacy of the problem increase, and the figure illustrates the efficiency of each method under different conditions.
In **Fig. 5**, we compare the graphical results of HEURISTIC1 (a heuristic algorithm used to obtain reasonable solutions within a short time) with the results of two other methods, BA and SA, within the framework of Local Search Models, based on the data presented in **Table 6**.

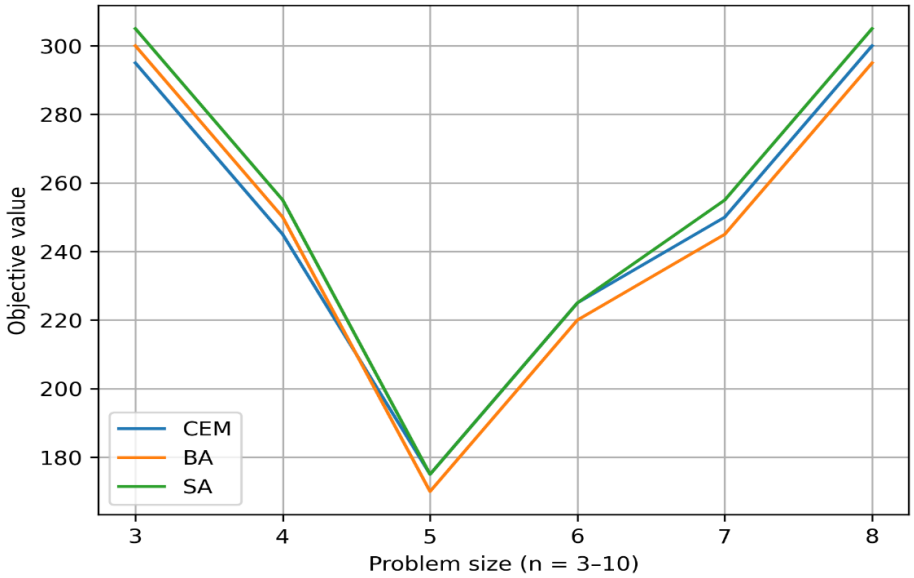


**Figure 3.** Performance comparison of the Cross-Entropy Method (CEM) with local search methods, namely Simulated Annealing (SA) and the Bee Algorithm (BA), for the BO-SCE_m R_L problem with instance sizes ranging from n = 3 to 10.

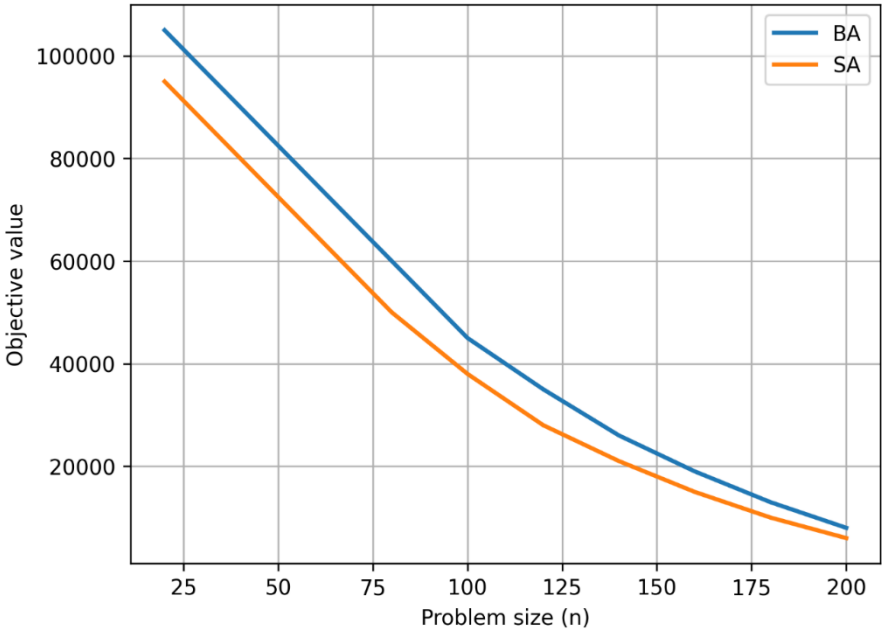


**Figure 4.** Performance comparison of the local search methods, namely Simulated Annealing (SA) and the Bee Algorithm (BA), for the BO-SCE_m R_L problem with instance sizes ranging from n = 20 to 200 in steps of 20.

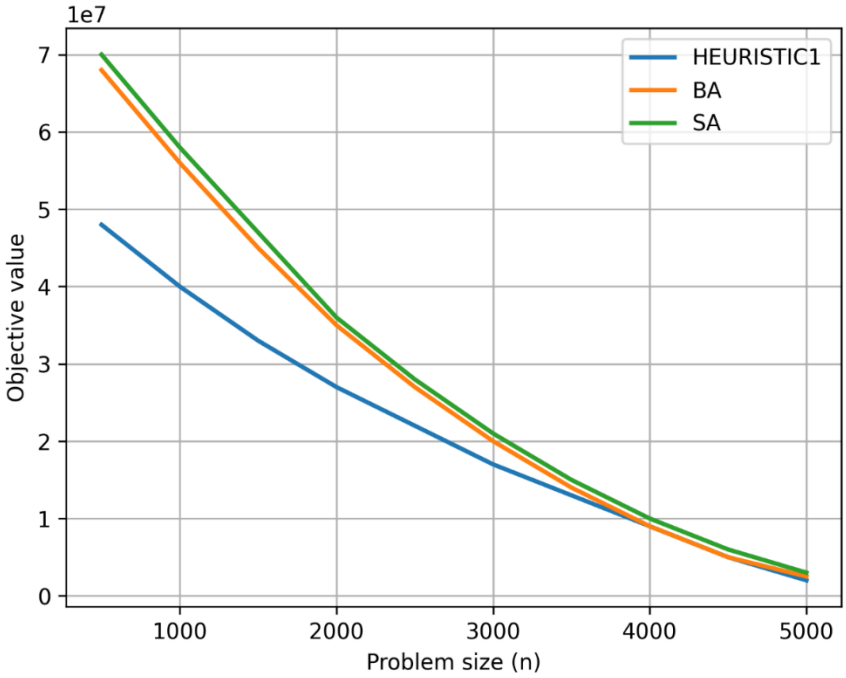


**Figure 5.** Performance comparison of HEURISTIC1 with the local search methods, namely the Bee Algorithm (BA) and Simulated Annealing (SA), for the BO-SCE_m R_L problem with instance sizes ranging from n = 500 to 5000 in steps of 500.

## 9. CONCLUTIONS

This study addressed a critical research gap identified in the literature, namely the lack of efficient and scalable metaheuristic approaches capable of simultaneously optimizing multiple conflicting objectives in NP-hard machine scheduling problems. While exact methods suffer from rapid performance degradation as problem size increases, many existing heuristics fail to balance solution quality, diversity, and computational efficiency. The experimental results demonstrate that both Simulated Annealing (SA) and the Bee Algorithm (BA) effectively bridge this gap. SA provides fast convergence and low computational cost, making it suitable for time-sensitive applications, whereas BA consistently achieves superior solution quality and Pareto front diversity. The comparative analysis confirms that the proposed approaches offer a practical and scalable alternative to exact methods, particularly for large-scale instances where traditional algorithms become infeasible. These findings directly respond to the initial research gap by showing that nature-inspired metaheuristics can deliver balanced, high-quality solutions within reasonable computational time for bi-criteria and bi-objective scheduling problems.

## Nomenclature

| Symbols | Description | symbols | Description |
|---|---|---|---|
| $n$ | Number of jobs | $Ti$ | Tardiness of job i |
| $m$ | Number of machines | $di$ | Due date of job i |
| $pi$ | Processing time of job i | $f1$ | First objective function |
| $Ci$ | Completion time of job i | $f2$ | Second objective function |
| $Cmax$ | Makespan | $S$ | Solution schedule |

**Acknowledgements**

The authors would like to thank the University of Baghdad, College of Science, Department of Mathematics, for supporting this research.

**Credit Authorship Contribution Statement**

Iraq Tareq Abbas**:** Conceptualization, Methodology, Supervision, Writing review & editing. Yasameen M. Mohammed: Software, Data curation, Validation, Writing – original draft.

**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**

Al-Kayiem, H.H., 2010. Study on the thermal accumulation and distribution inside a parked car cabin. *American Journal of Applied Sciences*, 7(6), pp. 784–789.

Al-Nuaimi, A., 2015. Local search algorithms for multiobjective scheduling problem. *Journal of Al-Rafidain University College for Sciences*, pp. 201–217. https://doi.org/10.55562/jrucs.v36i2.255

Ali, F.H., 2020. Optimal and near optimal solutions for multi objective function on a single machine. In: *2020 International Conference on Computer Science and Software Engineering (CSASE).* https://doi.org/10.1109/CSASE48920.2020.9142053

Bakar, M.R., Abbas, I.T., Kalal, M.A., AlSattar, H.A., Bakhayt, A.G. and Kalaf, B.A., 2017. Solution for multi-objective optimisation master production scheduling problems based on swarm intelligence algorithms. *Journal of Computational and Theoretical Nanoscience*, 14(11), pp. 5184–5194.

Błażewicz, J., Ecker, K.H., Pesch, E., Schmidt, G. and Węglarz, J., 2007. *Handbook on scheduling: From theory to applications*. Berlin: Springer.

Burmeister, S.C., Guericke, D. and Schryen, G., 2024. A memetic NSGA-II for the multi-objective flexible job shop scheduling problem with real-time energy tariffs. *Flexible Services and Manufacturing Journal*, 36, pp. 1530–1570.

Celik, E. and Topaloglu, S., 2021. A novel simulated annealing-based optimization approach for cluster-based task scheduling. *Cluster Computing*, 24(4), pp. 2927–2956.

Chen, B., Zeng, Z. and Zhang, Y., 2014. A new local search-based multiobjective optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 19(1), pp. 50–73.

Chong, C.S., Low, M.Y.H., Sivakumar, A.I. and Gay, K.L., 2006. A bee colony optimization algorithm to job shop scheduling. In: *Proceedings of the Winter Simulation Conference*, pp. 1954–1961.

Colombo, F., 2014. *Mathematical programming algorithms for network optimization problems*. PhD thesis. Università degli Studi di Milano.

Davidović, T., Ramljak, D. and Šelmić, M., 2015. Bee colony optimization—Part I: The algorithm overview. *YUJOR*, 25(1), pp. 33–56.

Doctor, S., Venayagamoorthy, G.K. and Gudise, V.G., 2004. Optimal PSO for collective robotic search applications. In: *Proceedings of the Congress on Evolutionary Computation*, pp. 1390–1395.

Festa, P., 2014. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In: *International Conference on Transparent Optical Networks*, pp. 1–20.

Gulati, R. and Singh, N., 2014. A literature review of bee colony optimization algorithms. In: *Innovative Applications of Computational Intelligence on Power, Energy and Controls (CIPECH)*, pp. 499–504.

Ibrahim, M.H., 2022. Solving multi-objectives function problem using branch and bound and local search methods. *International Journal of Nonlinear Analysis and Applications*, pp. 1649–1658.

Khan, K.A. and Khan, A.S., 2012. A comparison of BA, GA, PSO, BP and LM for training feed forward neural networks in e-learning context. *International Journal of Intelligent Systems and Applications*.

Khare, A. and Rangnekar, S., 2013. A review of particle swarm optimization and its applications in solar photovoltaic system. *Applied Soft Computing*, 13(5), pp. 2997–3006.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., 1983. Optimization by simulated annealing. *Science*, 220(4598), pp. 671–680.

Lalwani, S., Sharma, H. and Dashora, Y., 2013. A comprehensive survey: Applications of multi-objective particle swarm optimization algorithm. *Transactions on Combinatorics*, 2(1), pp. 39–101.

Maraveas, C., 2022. Applications of IoT for optimized greenhouse environment and resources management. *Computers and Electronics in Agriculture*, 198, P. 106993.

Marini, F. and Walczak, B., 2015. Particle swarm optimization (PSO): A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 149, pp. 153–165.

Meissner, M., Schmuker, M. and Schneider, G., 2006. Optimized particle swarm optimization and its application to artificial neural network training. *BMC Bioinformatics*, 7(1), P. 125.

Meng, L., Zhang, C., Zhang, B., Gao, K., Ren, Y. and Sang, H., 2023. MILP modelling and optimization of multi-objective flexible job shop scheduling problem with controllable processing times. *Swarm and Evolutionary Computation*, P. 101374.

Möhring, R.H., Schulz, A.S. and Uetz, M., 2003. Solving project scheduling problems by minimum cut computations. *Management Science*, 49(3), pp. 330–350.

Mou, J., Song, W. and Chen, Q., 2017. Multi-objective inverse scheduling optimization of single-machine shop system with uncertain due-dates. *Cluster Computing*, 20(1), pp. 371–390.

Mousa, A.A., El-Shorbagy, M.A. and Abd-El-Wahab, A., 2012. Local search-based hybrid particle swarm optimization algorithm for multiobjective optimization. *Swarm and Evolutionary Computation*, 3, pp. 1–14.

Onwunalu, J.E. and Durlofsky, L.J., 2010. Application of a particle swarm optimization algorithm for determining optimum well location and type. *Computational Geosciences*, 14(1), pp. 183–198.

Pham, D.T., Castellani, M., Ghanbarzadeh, A. and Koç, E., 2007. The bees algorithm: A novel tool for complex optimization problems. *International Journal of Manufacturing Research*, 2(4), pp. 454–472.

Pinedo, M.L., 2016. *Scheduling: Theory, algorithms, and systems*. 5th ed. Cham: Springer.

Poli, R., 2007. An analysis of publications on particle swarm optimization applications. *Journal of Artificial Evolution and Applications*.

Teodorovic, D., 2006. Bee colony optimization: Principles and applications. In: *Neural Network Applications Conference*, pp. 151–156.

Thomas, H., Cormen, T., Leiserson, C. and Rivest, R., 2009. *Introduction to algorithms*. 3rd ed.

Umar, M.F., 2025. Predicting the durability properties of concrete with recycled plastic aggregate. *Journal of Engineering*, 31(10), pp. 1–22. https://doi.org/10.31026/j.eng.2025.10.01

Wang, Y., Li, J. and Zhang, H., 2022. A hybrid particle swarm optimization and simulated annealing algorithm for job shop scheduling. *European Journal of Operational Research*, 299(3), pp. 1021–1035.

Wu, C.C., 2007. Heuristic algorithms for solving maximum lateness scheduling problem with learning considerations. *Computers & Industrial Engineering*, 52(1), pp. 124–132.

Wu, R., Luo, E., Li, X., Tang, H. and Li, Y., 2025. Hybrid artificial bee colony algorithm with Q-learning for distributed scheduling. *Swarm and Evolutionary Computation*.

Yousif, S.F., Ali, F.H. and Alshaikhli, K.F., 2023. Using local search methods for solving two multi-criteria machine scheduling problems, *Al-Mustansiriyah Journal of Science*, 34(4), pp. 96–103. https://doi.org/10.23851/mjs.v34i4.1430

Yousif, S.F., 2024. Solving maximum early jobs time and range of lateness jobs times problem. *Iraqi Journal of Science*, pp. 923–937. https://doi.org/10.24996/ijs.2024.65.2.28

Zhang, W., 2024. Enhancing multi-objective evolutionary algorithms with machine learning for scheduling problems. *Frontiers in Industrial Engineering*, 2, P. 1337174.

Zhang, Y., 2015. A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, P. 931256.

# حل مشاكل جدولة الماكنة ثنائية المعايير وثنائية الأهداف باستخدام خوارزميتين تطويريتين

**ياسمين معين محمود[1,2] ، عراق طارق عباس[1,*]**

[1] قسم الرياضيات وتطبيقات الحاسوب, كلية العلوم, جامعة النهرين, بغداد, العراق
[2] قسم الرياضيات، كلية العلوم، جامعة بغداد، بغداد، العراق

**الخلاصة**

تضمن العديد من الأنظمة الصناعية معايير وأهداف متعددة، مما يجعلها مشكلات معقدة للغاية في علم الحوسبة، مثل مشكلات جدولة المهام وغيرها. في هذا البحث، تم اقترح دراسة مشكلات جدولة الماكنة ثنائية المعايير وثنائية الأهداف، والتي تم حلها باستخدام خوارزميتين تطوريتين مستوحاتين من الطبيعة، وهما خوارزمية التلدين (Simulated Annealing – SA) وخوارزمية وكيل النحل (Bee Agent based Algorithm – BA). يمكن تمثيل المشكلة اعلاه بجدولة مجموعة من المهام على عدة آلات، حيث تُعد هذه المسألة من المسائل الاساسية لجدولة الماكنة، لأن الحل يجب أن يركّز على تحقيق تحسين متزامن لهدفين متضاربين: تقليل وقت الإكمال الكلي (Makespan) وتقليل إجمالي التأخير (Total Tardiness). وتُصنف هذه المشكلة ضمن فئة المشكلات المعروفة بصعوبتها الحسابية (NP–Hard)، ولهذا السبب تم استخدام طريقتين تطوريتين للبحث عن حلول ذكية ضمن مساحة حلول واسعة ومعقدة للغاية. ايضا، تم تطوير نموذج رياضي لمشكلة الجدولة متعددة الاهداف والمعاييراعتمادًا على الأهداف المذكورة أعلاه. وقد قمنا باقتراح تعديل ثنائي القاعدة (dual–based tune–up) لكل من خوارزميتي SA وBA، حيث تم برمجة كل منهما وتنفيذها خصيصًا لحل النموذج المقترح الخاص بالجدولة بما يتناسب مع الطبيعة الثنائية لوظائف المشكلة قيد الدراسة. وأظهرت النتائج أن كلتا الخوارزميتين قادرتان على الحصول على حلول متوازنة وفعّالة من حيث الزمن.

**الكلمات المفتاحية:** الجدولة، التلدين المُحاكى، خوارزمية النحل متعدد الأهداف، جبهة باريتو، التحسين التطوري.