Mangalayatan Journal of
Scientific and Industrial Research

Man. J. Sci. Ind. Res.
Volume 2 (1) (2025), 25202
ISSN: 3049-2815

***Article***

# Adaptive Memetic Algorithm for Solving Complex Graph Coloring Problem

Anjali Gangrade[1], Bhawna Agrawal[2] and Sanjit Kumar[3]
[1,2] *Department of Mathematics, Rabindranath Tagore University, Bhopal (M.P), India,*
[3]*Department of Mathematics Lakshmi Narain College of Technology & Science, Bhopal (M.P), India,*
[1]*anjali.gangrade@gmail.com* [2]*bhawnakhushiagrawal@gmail.com*
[3]*sanjeetkumarmath@gmail.com*

## Abstract

*Graph vertex coloring with a fixed number of colors is a well-known, extensively studied NP-complete problem. The most effective approaches to solving this challenge have been hybrid algorithms, such as memetic algorithms or quantum annealing, which combine the strengths of local search within population-based frameworks. In this paper, we address a production scheduling problem involving three distinct car models-XUV, Sedan, and Mini XUV requiring assignment to two painting lines and two assembly lines. The objective is to minimize total production time while adhering to various operational constraints such as model compatibility for painting lines, specific assembly line restrictions, and differing painting and assembly times for each model. To solve this problem efficiently, we propose a solution based on the Memetic Graph Coloring Algorithm. Our approach incorporates a graph-based representation of the problem, where nodes represent tasks (painting and assembly of individual car units) and edges represent incompatibilities or resource conflicts. The Memetic Algorithm optimizes the assignment of tasks to time slots, leveraging both crossover and mutation operations to explore feasible solutions. Specifically, mutation is employed to randomly alter schedules by swapping tasks within slots, which introduces diversity and enhances the search process.*

***Keywords:*** *Production Scheduling, Memetic Algorithm, Graph Coloring, Car Manufacturing, Optimization, Idle Time, Scheduling Constraints.*

## 1. Introduction

Efficient production scheduling is a critical factor in optimizing the manufacturing process, particularly in industries with complex product lines, such as the automotive industry. In multi-model production systems, assigning tasks such as painting and assembly to limited resources (e.g., painting lines, assembly lines) requires careful consideration of model compatibility, operational constraints, and production times. Failure to address these constraints leads to inefficient use of resources, increased idle time, and extended production durations. Traditional scheduling methods often struggle to achieve optimal solutions under such complex conditions. In response, metaheuristic approaches, particularly evolutionary algorithms, have gained significant attention for their ability to solve complex scheduling problems. One such approach is the Memetic Algorithm (MA), which combines the global search capability of genetic algorithms with local optimization strategies, offering a powerful tool for solving combinatorial optimization problems. Memetic algorithms have been successfully applied to various scheduling

Received: 3 Feb 2025; Accepted: 15 May. 2025

problems, including job-shop scheduling, flow-shop scheduling, and timetabling. These works highlight the ability of memetic algorithms to balance exploration and exploitation in the solution space, enabling the discovery of near-optimal solutions efficiently.

In the context of graph coloring, where nodes represent tasks and edges represent conflicts or resource constraints, several studies have demonstrated the effectiveness of this model in scheduling problems. Morgenstern and Shapiro showed how graph coloring could be applied to exam scheduling [1].

Despite these advances, few studies have addressed the specific challenges encountered in automotive production scheduling, where tasks are constrained by model-specific requirements and resource limitations. Models like the XUV, Sedan, and Mini XUV have varying painting and assembly times, and there are strict restrictions on which lines can handle which models. For example, XUV and Sedan cannot be painted on the same line, and each model has different assembly line requirements. Existing solutions often fail to efficiently minimize total production time while managing these constraints.

The graph vertex coloring problem (GVCP) consists in finding the minimum number of colors, called chromatic number $\chi(G)$, required to color the graph G while respecting these binary constraints GVCP is a well-documented and much-studied problem because this simple formalization can be applied to various issues such as frequency assignment problems (Aardal et al. , Dib et al Comprehensive surveys on the Generalized Vertex Coloring Problem (GVCP) can be found in the works [2,3]. Galinier and Hertz, Galinier et al. and Malaguti and Toth first two studies categorize heuristics based on the selected search space [4,7,16]. Hertz et al variable Space Search is particularly notable for its innovative and educational approach, as it operates across three distinct search spaces. A more traditional way of classifying these methods is by considering how they navigate the search space, with three main heuristic types identified: constructive methods, local searches, and population-based approaches [14].

**Table 1: This table provides a general overview of each method's characteristics**.

| **Method** | **Description** | **Time Complexity** | **Space Complexity** | **Color Usage** | **Quality** |
|---|---|---|---|---|---|
| Greedy Coloring | Assign colors greedily, Largest degree first | O(V + E) | O(V) | Good | Fair |
| Backtracking | Recursive, Try all possible colors | O(V^V) | O(V) | Excellent | Excellent |
| DSATUR | Degree of Saturation, dynamic order | O(V + E) | O(V) | Good | Good |
| RLF (Recursive Largest First) | Recur-sive, largest degree first | O(V + E) | O(V) | Good | Good |
| Tabu Search | Metaheuristic, local | O(V + E) | O(V) | Excellent | Excellent |

| | search | | | | |
|---|---|---|---|---|---|
| Genetic Algorithm | Evolutionary, population based | O(V + E) | O(V) | Excellent | Excellent |

**Note:** Time Complexity: V = number of vertices, E = number of edges
Space Complexity: V = number of vertices
Color Usage: How efficiently colors are used (Good: reasonable, Excellent: optimal)
Quality: Solution quality (Fair: acceptable, Good: good, Excellent: optimal)

## 2. Memetic Algorithms (MAs)

Memetic algorithms (MAs) are a type of metaheuristic optimization technique inspired by the concept of memes, which are cultural equivalents of genes. In the context of graph coloring, MAs combine population-based search with local improvement strategies to find high-quality solutions.

### 2.1 Graph Coloring Problem

Given a graph G = (V, E), assign a color to each vertex in V such that:

i. Adjacent vertices have different colors.
ii. The total number of colors used is minimized.

### 2.2 Memetic Algorithm for Graph Coloring

1. **Initialization:** Generate an initial population of solutions, where each solution is a coloring of the graph.
2. **Evaluation:** Calculate the fitness of each solution, typically the number of colors used or a penalty function for adjacent vertices with the same color.
3. **Selection:** Select a subset of solutions with the best fitness values to form a new population.
4. **Crossover:** Apply crossover operators to combine selected solutions and create new offspring solutions.
5. **Mutation**: Apply mutation operators to introduce random changes in the offspring solutions.
6. **Local Search:** Apply local improvement strategies to refine the solutions and escape local optima.
7. **Replacement:** Replace the least fit solutions in the population with the improved offspring solutions.
8. **Termination:** Repeat steps 2-7 until a stopping criterion is met (e.g., maximum iterations or satisfactory solution quality).

### 2.3 Local Improvement Strategies

1. **Kempe Chain:** Swap colors between adjacent vertices to reduce the number of colors used.
2. **Tabu Search:** Explore neighboring solutions and avoid revisiting recently explored solutions.
3. **Simulated Annealing:** Accept worse solutions with a probability decreasing over

time.

**2.4 Advantages**

1. **Robustness:** MAs can handle complex graphs and large problem instances.
2. **Flexibility:** MAs can be adapted to different graph coloring variants and constraints.
3. **Quality:** MAs can find high-quality solutions, often better than traditional algorithms.

**2.5 Challenges**

1. **Computational Cost:** MAs can be computationally expensive, especially for large graphs.
2. **Parameter Tuning:** MAs require careful tuning of parameters, such as population size and local search intensity.

By combining the strengths of population-based search and local improvement strategies, memetic algorithms offer a powerful approach to solving the graph coloring problem.

**3.Real-Life Problems**

Here are some more real-life problems that can be solved using graph coloring:

1. **Scheduling Sports Leagues:** Create a schedule for teams to play each other, ensuring no team plays two games at the same time.
2. **Frequency Assignment:** Assign frequencies to radio stations in a way that minimizes interference between nearby stations.
3. **Resource Allocation:** Allocate resources (e.g., machines, personnel) to tasks in a way that minimizes conflicts and maximizes efficiency.
4. **Traffic Light Scheduling:** Coordinate traffic lights to minimize congestion and reduce commute times.
5. **Exam Scheduling:** Schedule exams for students in a way that minimizes conflicts and ensures fair timing.
6. **Production Planning:** Plan production schedules for multiple products on shared machines to minimize downtime and maximize output.
7. **Network Optimization:** Optimize network topology to minimize congestion and ensure reliable data transmission.
8. **Timetabling:** Create timetables for public transportation, ensuring efficient use of resources and minimizing conflicts.
9. **Workforce Scheduling:** Schedule employee shifts to ensure adequate coverage while minimizing conflicts and overtime.
10. **VLSI Design:** Assign colors to components in a digital circuit to minimize conflicts and ensure efficient layout.

**3.1 Production Planning problem**

Here's the mathematical solution to the Production Planning problem using Memetic Algorithm for Graph Coloring: Let

i. $G = (V, E)$ be the graph representing the production planning problem
ii. $V = \{1, 2, \ldots, n\}$ be the set of vertices (production slots)
iii. $E = \{(i, j) \mid i, j \in V\}$ be the set of edges (conflicts between models)

iv. $C = \{1, 2, \ldots, k\}$ be the set of colors (models)
v. $x_i \in C$ be the color assigned to vertex $i$

Objective: Minimize the total production time:

$$\sum_{i=1}^{n} p_i * x_i$$

Subject to:

i. $\sum_{i=1}^{n} x_i,\ x_i \leq k$ (limit on number of colors)
ii. $x_i \neq x_j$ for $(i, j) \in E$ (conflict constraint)
iii. $x_i \in \mathrm{C}$ for $i \in V$ (color assignment constraint)

### 3.1.1 Memetic Algorithm

1. Initialize population $P = \{x^1, x^2, \ldots, x^m\} \in C^n$
2. Evaluate fitness $f(x^i) = \sum_{i=1}^{n=n} p_i * x_i$
3. Select top $\frac{\mathrm{m}}{2}$ solutions with best fitness
4. Crossover: create new solutions $x'$ by combining selected solutions
5. Mutation: introduce random changes in $x'$
6. Local Search: improve solutions using neighborhood search
7. Replace least fit solutions with improved solutions
8. Repeat steps 2-7 until stopping criterion is met

### 3.1.2 Problem: "Painting and Assembly Line Scheduling"

**Company:** Abhyudit Automotive

**Goal:** Schedule production of three car models (XUV, Sedan, Mini XUV) on two painting lines ($\mathbf{P_1}$ and $\mathbf{P_2}$) and two assembly lines ($\mathbf{A_1}$ and $\mathbf{A_2}$) to minimize production time and ensure efficient use of resources.

**Constraints:**

1. **Model Compatibility:** XUV and Sedan cannot be painted on the same line, while Mini XUV can be painted on either line.
2. **Assembly Line Restrictions:** XUV requires Assembly Line $\mathbf{A_1}$, while Sedan and Mini XUV require Assembly Line $\mathbf{A_2}$.
3. **Painting Time:** Each model has a different painting time: XUV (2 hours), Sedan (3 hours), and Mini XUV (1 hours).
4. **Assembly Time:** Each model has a different assembly time: XUV (3 hours), Sedan (2 hours), and Mini XUV (2 hours).

**Objective:** Schedule the production of 10 units of each model to minimize the total production time, ensuring that:

1. No two models that cannot be painted on the same line are scheduled simultaneously.
2. Each model is assembled on the required assembly line.
3. The painting and assembly lines are used efficiently, minimizing idle time.

**Graph Coloring Representation:**

i. **Vertices:** Production slots (time intervals)
ii. **Edges:** Conflicts (two models that cannot be painted on the same line or assembled on the same line)
iii. **Colors:** Models (XUV, Sedan, and Mini XUV)

### 3.2 Mathematical Model of problem

Let's break down the problem into a mathematical model:

**Variables:**

- $x_{\alpha_1}, x_{\alpha_2}, x_{\beta_1}, x_{\beta_2}, x_{\gamma_1}, x_{\gamma_2}$: Binary variables representing the assignment of models to painting lines (1 if assigned, 0 otherwise)
- $y_{\alpha_1}, y_{\alpha_2}, y_{\beta_1}, y_{\beta_2}, y_{\gamma_1}, y_{\gamma_2}$: Binary variables representing the assignment of models to assembly lines (1 if assigned, 0 otherwise)
- $t_{p_1}, t_{p_2}$: Production times for painting lines
  - $t_{A_1}, t_{A_2}$: Production times for assembly lines

**Note:** $\alpha$: $XUV$ , $\beta$: $Sedan$, $\gamma$: $Mini\ XUV$

**Objective**

Minimize total production time:

minimize: $t_{p_1} + t_{p_2} + t_{A_1} + t_{A_2}$

Where $t_{p_1} + t_{p_2}$ are the production times for painting lines 1 and 2.

$t_{A_1} + t_{A_2}$ are the production times for assembly lines 1 and 2.

**Constraints** Model assignment to painting lines:

$x_{\alpha_1} + x_{\alpha_2} = 1$ (Model **XUV** assigned to one painting line)

$x_{\beta_1} + x_{\beta_2} = 1$ (Model **Sedan** assigned to one painting line)

$x_{\gamma_1} + x_{\gamma_2} = 1$ (Model **Mini XUV** assigned to one painting line)

Model assignment to assembly lines:

$y_{\alpha_1} + y_{\alpha_2} = 1$ (Model **XUV** assigned to one assembly line)

$y_{\beta_1} + y_{\beta_2} = 1$ (Model **Sedan** assigned to one assembly line)

$y_{\gamma_1} + y_{\gamma_2} = 1$ (Model **Mini XUV** assigned to one assembly line)

Painting line capacity:

$t_{p_1} \leq 8$ (max production time for $\mathbf{P_1}$)

$t_{p_2} \leq 8$ (max production time for $\mathbf{P_2}$)

Assembly line capacity:

$t_{A_1} \leq 10$ (max production time for $A_1$)

$t_{A_2} \leq 10$ (max production time for $A_2$)

Model compatibility:

Ensure that modals are not placed on incompatible lines (avoid conflicts between models and lines)

$\boldsymbol{x_{\alpha_1} + y_{\alpha_2} \leq 1}$ (Model **XUV** not on $\boldsymbol{P_1}$ and $\boldsymbol{A_1}$)

$\boldsymbol{x_{\beta_1} + y_{\beta_2} \leq 1}$ (Model **Sedan** not on $\boldsymbol{P_1}$ and $\boldsymbol{A_2}$)

$\boldsymbol{x_{\gamma_1} + x_{\gamma_2} \leq 1}$ (Model **Mini XUV** not on $\boldsymbol{P_1}$ and $\boldsymbol{A_1}$)

### 3.2.1 Solution

Solution can be represented as a binary vector like

$\boldsymbol{S = \{x_{\alpha_1}, x_{\alpha_2}, x_{\beta_1}, x_{\beta_2}, x_{\gamma_1}, x_{\gamma_2} y_{\alpha_1}, y_{\alpha_2}, y_{\beta_1}, y_{\beta_2}, y_{\gamma_1}, y_{\gamma_2}\}}$

- 3 models (XUV, Sedan, Mini XUV),
- 2 painting lines ($\mathbf{P_1}, \mathbf{P_2}$),
- 2 assembly lines ($\mathbf{A_1}, \mathbf{A_2}$).

- $\mathbf{x_{\alpha_1} = 1}$ if the XUV is assigned to painting line $\mathbf{P_1}$
- $\mathbf{y_{\alpha_1} = 1}$ if the XUV is assigned to assembly line $\mathbf{A_1}$
- $\boldsymbol{x_{\alpha_1}, x_{\alpha_2}}$: assignment of Model XUV to Painting Line 1 or 2.
- $\boldsymbol{x_{\beta_1}, x_{\beta_2}}$: assignment of Model Sedan to Painting Line 1 or 2.
- $\boldsymbol{x_{\gamma_1}, x_{\gamma_2}}$: assignment of Model Mini XUV to Painting Line 1 or 2.
- $\boldsymbol{y_{\alpha_1}, y_{\alpha_2}}$: assignment of Model XUV to Assembly Line 1 or 2.
- $\boldsymbol{y_{\beta_1}, y_{\beta_2}}$: assignment of Model Sedan to Assembly Line 1 or 2.
- $\boldsymbol{y_{\gamma_1}, y_{\gamma_2}}$: assignment of Model Mini XUV to Assembly Line 1 or 2.

**Valid Solution Criteria**

**Model Assignment**: Each model should be assigned to one painting and one assembly line (sum of assignments for each model$= \mathbf{1}$).

**Step 1: Initial Population Generation**

The initial population is consists of possible solutions, each of which assigns models to painting and assembly lines.

**Table 2**

| **Chromosome (Solution)** | **XUV Painting Line** | **XUV Assembly Line** | **Sedan Painting Line** | **Sedan Assembly Line** | **Mini XUV Painting Line** | **Mini XUV Assembly Line** |
|---|---|---|---|---|---|---|
| $\mathbf{S_1}$ | Painting Line 1 | Assembly Line 2 | Painting Line 2 | Assembly Line 1 | Painting Line 1 | Assembly Line 1 |
| $\mathbf{S_2}$ | Painting Line 2 | Assembly Line 1 | Painting Line 1 | Assembly Line 2 | Painting Line 2 | Assembly Line 2 |
| $\mathbf{S_3}$ | Painting Line 2 | Assembly Line 1 | Painting Line 2 | Assembly Line 2 | Painting Line 1 | Assembly Line 1 |
| $\mathbf{S_4}$ | Painting Line 1 | Assembly Line 1 | Painting Line 2 | Assembly Line 2 | Painting Line 2 | Assembly Line 1 |

Each chromosome represents a possible solution to the problem, with assignments of models to painting and assembly lines.
We aim to select the best solutions that minimize production time.

**Step 2**

**Table 3: Offspring Generation (Crossover)**

| **Parent 1 ($S_1$)** | **Parent 2 ($S_2$)** | **Crossover Point** | **Offspring 1** | **Offspring 2** |
|---|---|---|---|---|
| $S_1$ (XUV: $P_1$-$A_2$, Sedan: $P_2$-$A_1$, Mini XUV: $P_1$-$A_1$) | $S_2$ (XUV: $P_2$-$A_1$, Sedan: $P_1$-$A_2$, Mini XUV: $P_2$-$A_2$) | Between Sedan and Mini XUV | XUV: $P_1$-$A_2$, Sedan: $P_1$-$A_2$, Mini XUV: $P_2$-$A_2$ | XUV: $P_2$-$A_1$, Sedan: $P_2$-$A_1$, Mini XUV: $P_1$-$A_1$ |

Offspring are generated by performing crossover between selected parent solutions. Here, we use a one-point crossover method.

In the crossover, the offspring inherit genes from each parent based on the crossover point. In this case, the Sedan assignment is used as the crossover point.

**Step 3: Mutation Table:**

Mutation is applied to introduce diversity in the population by randomly altering one or more assignments in the offspring.

**Table 4**

| **Offspring** | **Mutation Applied** | **New Solution After Mutation** |
|---|---|---|
| Offspring 1 | Change Mini XUV Assignment | XUV: $P_1$-$A_2$, Sedan: $P_1$-$A_2$, Mini XUV: **$P_1$-$A_1$** |
| Offspring 2 | Change Sedan Assignment | XUV: $P_2$-$A_1$, Sedan: **$P_1$-$A_2$**, Mini XUV: $P_1$-$A_1$ |

Mutations introduce new possible assignments by altering one random part of the solution. For example, Mini XUV's assignment is mutated in Offspring 1, and Sedan's assignment is mutated in Offspring 2.

**Step 4: Local Search Optimization (Memetic Algorithm Phase):**

This step involves improving the offspring by applying a local search to minimize conflicts and improve the objective function (e.g., minimizing production time).

**Table 5**

| **Offspring** | **Local Search Improvement** | **Improved Solution** |
|---|---|---|
| Offspring 1 | Check and adjust conflict between Mini XUV assignments. | XUV: $P_1$-$A_2$, Sedan: $P_1$-$A_2$, Mini XUV: $P_1$-$A_2$ (removes conflict) |
| Offspring 2 | Adjust Sedan assignment to avoid capacity constraint violation. | XUV: $P_2$-$A_1$, Sedan: $P_2$-$A_2$, Mini XUV: $P_1$-$A_1$ (improves capacity balance) |

The local search is applied to improve the quality of the offspring by removing conflicts (e.g., ensuring model compatibility or capacity limits are respected).

**Step 5: Final Solution Table:**
After crossover, mutation, and local search, we select the best solutions (offspring) to form the next generation or finalize the optimal solution.

**Table 6**

| **Final Solution** | **XUV Painting Line** | **XUV Assembly Line** | **Sedan Painting Line** | **Sedan Assembly Line** | **Mini XUV Painting Line** | **Mini XUV Assembly Line** | **Objective (Production Time)** |
|---|---|---|---|---|---|---|---|
| **Solution 1** | Painting Line 1 | Assembly Line 2 | Painting Line 1 | Assembly Line 2 | Painting Line 2 | Assembly Line 1 | Minimized Production Time |
| **Solution 2** | Painting Line 2 | Assembly Line 1 | Painting Line 2 | Assembly Line 1 | Painting Line 1 | Assembly Line 2 | Minimized Production Time |

**Summary**:

- **Initial Population**: Four possible solutions are generated.
- **Offspring Generation**: Two offspring are created using crossover.
- **Mutation**: Random mutation is applied to diversify the offspring.
- **Local Search**: Further optimization is performed to remove conflicts and improve the objective function.
- **Final Solution**: The best solutions (with minimized production time) are selected.

### 3.2.2 Solution (Second Solution with Time Slot):

The goal of this problem is to schedule the production of 30 units (10 XUVs, 10 Sedans, and 10 Mini XUVs) across two painting lines ($\boldsymbol{P_1}, \boldsymbol{P_2}$ ) and two assembly lines ($\boldsymbol{A_1}, \boldsymbol{A_2}$), while minimizing production time and avoiding conflicts. The solution will use the **Memetic Graph Coloring Algorithm**, which combines evolutionary strategies (genetic algorithm) with local search techniques.

**Step 1: Problem Representation:**

Each model's painting and assembly tasks will be represented as **nodes**. The constraints of model compatibility (no simultaneous painting for XUV and Sedan) and assembly line restrictions are represented as **edges** between conflicting nodes.

**Graph Nodes:**

1. **Painting nodes**: 10 XUVs, 10 Sedans, 10 Mini XUVs.

2. **Assembly nodes**: 10 XUVs, 10 Sedans, 10 Mini XUVs.

**Edges (Conflicts):**

1. XUV and Sedan cannot be painted at the same time on the same line.
2. XUVs must be assembled on $A_1$, while Sedans and Mini XUVs must be assembled on $A_2$.
3. No overlapping tasks are allowed (i.e., one model cannot be both painted and assembled simultaneously).

**Step 2: Initial Population (Chromosomes):**

Each **chromosome** represents a possible solution, where tasks (painting and assembly) are scheduled in different time slots on specific lines. The chromosome contains assignments of tasks, each task associated with:

- A painting or assembly line.
- A time slot.

A chromosome can be represented like this:

**Explanation of Chromosomes**

- **Time Slot**: Indicates the slot in which tasks are scheduled.
- **Painting Line $P_1/P_2$**: Indicates the car model being painted on each line in a given time slot.
- **Assembly Line $A_1/A_2$**: Indicates the car model being assembled on each line in a given time slot.

**Table 7: Chromosome -1**

| Time Slot | Painting Line $P_1$ | Painting Line $P_2$ | Assembly Line $A_1$ | Assembly Line $A_2$ |
|---|---|---|---|---|
| **Slot 1** | 1-unit XUV (2 hrs) | 1-unit Sedan (3 hrs) | 1-unit XUV (3 hrs) | 2 units Sedan (2 hrs each) |
| **Slot 2** | 1 unit XUV (2 hrs) | 1 unit Sedan (3 hrs) | 1-unit XUV (3 hrs) | 2 units Sedan (2 hrs each) |
| **Slot 3** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | 1-unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) |
| **Slot 4** | 2 units Mini XUV (1 hr each) | 2 units Sedan (3 hrs each) | 1-unit XUV (3 hrs) | 3 units Mini XUV (2 hrs each) |
| **Slot 5** | 2 units XUV (2 hrs each) | Idle | 1-unit XUV (3 hrs) | Idle |
| **Slot 6** | 2 units XUV (2 hrs each) | 2 units Sedan (3 hrs each) | 2 units XUV (3 hrs each) | 2 units Mini XUV (2 hrs each) |
| **Slot 7** | Idle | 2 units Sedan (3 hrs each) | Idle | 1 unit Mini XUV (2 hrs) |
| **Slot 8** | 1 unit Mini XUV | 1 unit Sedan (3 hrs) | Idle | Idle |

| | | | | |
|---|---|---|---|---|
| | (1 hr) | | | |
| **Slot 9** | Idle | Idle | Idle | Idle |
| **Slot 10** | Idle | Idle | Idle | Idle |

**Table 8: Chromosome -2**

| **Time Slot** | **Painting Line $P_1$** | **Painting Line $P_2$** | **Assembly Line $A_1$** | **Assembly Line $A_2$** |
|---|---|---|---|---|
| **Slot 1** | 2 units XUV (2 hrs each) | 2 units Sedan (3 hrs each) | 1 unit XUV (3 hrs) | 2 units Sedan (2 hrs each) |
| **Slot 2** | 2 units Mini XUV (1 hr each) | 2 units Sedan (3 hrs each) | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) |
| **Slot 3** | 2 units XUV (2 hrs each) | 2 units Mini XUV (1 hr each) | 2 units XUV (3 hrs each) | 2 units Mini XUV (2 hrs each) |
| **Slot 4** | 2 units Mini XUV (1 hr each) | Idle | 2 units XUV (3 hrs each) | 1 unit Mini XUV (2 hrs) |
| **Slot 5** | 1 unit XUV (2 hrs) | 2 units Sedan (3 hrs each) | 1 unit XUV (3 hrs) | Idle |
| **Slot 6** | 1 unit Mini XUV (1 hr) | 2 units Sedan (3 hrs each) | 1 unit XUV (3 hrs) | 2 units Sedan (2 hrs each) |
| **Slot 7** | 2 units XUV (2 hrs each) | 2 units Sedan (3 hrs each) | Idle | 2 units Mini XUV (2 hrs each) |
| **Slot 8** | Idle | 2 units Sedan (3 hrs each) | Idle | Idle |
| **Slot 9** | Idle | Idle | Idle | Idle |
| **Slot 10** | Idle | Idle | Idle | Idle |

**Table 9: Chromosome -3**

| **Time Slot** | **Painting Line $P_1$** | **Painting Line $P_2$** | **Assembly Line $A_1$** | **Assembly Line $A_2$** |
|---|---|---|---|---|
| **Slot 1** | 1-unit XUV (2 hrs) | 1-unit Sedan (3 hrs) | 1-unit XUV (3 hrs) | 2 units Sedan (2 hrs each) |
| **Slot 2** | 2 units Mini XUV (1 hr each) | 2 units Sedan (3 hrs each) | 2 units XUV (3 hrs each) | 2 units Sedan (2 hrs each) |
| **Slot 3** | 2 units XUV (2 hrs each) | 2 units Mini XUV (1 hr each) | 2 units XUV (3 hrs each) | 2 units Mini XUV (2 hrs each) |
| **Slot 4** | 2 units Mini XUV (1 hr each) | 1-unit Sedan (3 hrs) | Idle | 2 units Mini XUV (2 hrs each) |
| **Slot 5** | 2 units XUV (2 hrs each) | 2 units Sedan (3 hrs each) | 1-unit XUV (3 hrs) | Idle |
| **Slot 6** | 2 units XUV (2 hrs | 1-unit Sedan (3 | 1-unit XUV (3 hrs) | 2 units Mini |

| | | | | |
|---|---|---|---|---|
| | each) | hrs) | | XUV (2 hrs each) |
| **Slot 7** | Idle | 2 units Sedan (3 hrs each) | 1-unit XUV (3 hrs) | 1 unit Mini XUV (2 hrs) |
| **Slot 8** | 2 units Mini XUV (1 hr each) | Idle | Idle | Idle |
| **Slot 9** | Idle | Idle | Idle | Idle |
| **Slot 10** | Idle | Idle | Idle | Idle |

**Step 3: Fitness Function**

We now evaluate the **fitness** of each chromosome, where fitness is determined by:

1. **Minimizing idle time** on the painting and assembly lines.
2. **Avoiding conflicts** (e.g., no XUV and Sedan painted on the same line).
3. **Balancing the workload** on the lines to ensure optimal use.

**For each chromosome:**

- Count the total **production time** across all tasks (paint and assemble).
- Count the **idle time** (slots where no task is performed).
- Apply penalties if constraints are violated.

**Fitness Explanation:**

- The **fitness score** represents how well the schedule minimizes idle time relative to the total time. A higher fitness score indicates more efficient use of the painting and assembly lines.
- The formula for fitness is calculated as:

$$Fitness = \frac{Task\ Time}{Total\ Time\ (Task\ Time + Idle\ Time)}$$

**Table 10: Fitness Calculation Table:**

| **Chromosome** | **Line** | **Task Time (hrs)** | **Idle Time (hrs)** | **Total Time (Task + Idle)** | **Fitness (Task Time / Total Time)** |
|---|---|---|---|---|---|
| **Chromosome 1** | **Painting Line $P_1$** | 18 | 11 | 29 | $^{18}/_{29} = 0.621$ |
| | **Painting Line $P_2$** | 18 | 12 | 30 | $^{18}/_{30} = 0.600$ |
| | **Assembly Line $A_1$** | 15 | 15 | 30 | $^{15}/_{30} = 0.500$ |
| | **Assembly Line $A_2$** | 16 | 12 | 28 | $^{16}/_{28} = 0.571$ |
| | **Total** | 67 | 50 | 117 | $^{67}/_{117} = 0.573$ |
| **Chromosome** | **Painting Line** | 16 | 13 | 29 | $^{16}/_{29} = 0.552$ |

| | | | | | |
|---|---|---|---|---|---|
| **2** | $P_1$ | | | | |
| | **Painting Line $P_2$** | 18 | 10 | 28 | $^{18}/_{28} = 0.643$ |
| | **Assembly Line $A_1$** | 14 | 16 | 30 | $^{14}/_{30} = 0.467$ |
| | **Assembly Line $A_2$** | 18 | 10 | 28 | $^{18}/_{28} = 0.643$ |
| | **Total** | 66 | 49 | 115 | $^{66}/_{115} = 0.574$ |
| **Chromosome 3** | **Painting Line $P_1$** | 17 | 12 | 29 | $^{17}/_{29} = 0.586$ |
| | **Painting Line $P_2$** | 17 | 11 | 28 | $^{17}/_{28} = 0.607$ |
| | **Assembly Line $A_1$** | 16 | 14 | 30 | $^{16}/_{30} = 0.533$ |
| | **Assembly Line $A_2$** | 17 | 11 | 28 | $^{17}/_{28} = 0.607$ |
| | **Total** | 67 | 48 | 115 | $^{67}/_{115} = 0.583$ |

- Chromosome 3 has the highest fitness score of **0.583**, indicating that it uses the lines more efficiently than Chromosomes 1 and 2.

**Summary:**

- **Chromosome 1 Fitness** = 0.573
- **Chromosome 2 Fitness** = 0.574
- **Chromosome 3 Fitness** = 0.583 (Best Performing)

**Step 4: Offspring:**

We'll perform crossover between **Chromosome 1** and **Chromosome 2** and between **Chromosome 2** and **Chromosome 3**. The crossover point is the middle of the schedule (after Slot 5).

**Table 11: Offspring 1 (from Chromosome 1 and Chromosome 2):**

| **Time Slot** | **Painting Line $P_1$** | **Painting Line $P_2$** | **Assembly Line $A_1$** | **Assembly Line $A_2$** |
|---|---|---|---|---|
| **Slot 1** | 1 unit XUV (2 hrs) | 1 unit Sedan (3 hrs) | 1 unit XUV (3 hrs) | 2 units Sedan (2 hrs each) |
| **Slot 2** | 1 unit XUV (2 hrs) | 1 unit Sedan (3 hrs) | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) |
| **Slot 3** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | 2 units XUV (3 hrs each) | 2 units Mini XUV (2 hrs each) |
| **Slot 4** | 2 units Mini XUV (1 hr each) | 2 units Sedan (3 hrs each) | Idle | 3 units Mini XUV (2 hrs each) |

| | | | | |
|---|---|---|---|---|
| **Slot 5** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | Idle | Idle |
| **Slot 6** | 2 units XUV (2 hrs each) | 1units Sedan (3 hrs each) | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) |
| **Slot 7** | Idle | 1 unit Sedan (3 hrs) | Idle | 2 units Mini XUV (2 hrs each) |
| **Slot 8** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | Idle | 2 units Mini XUV (2 hrs each) |
| **Slot 9** | Idle | Idle | Idle | Idle |
| **Slot 10** | Idle | Idle | Idle | Idle |

**Table 12: Offspring 2 (from Chromosome 2 and Chromosome 3):**

| **Time Slot** | **Painting Line $P_1$** | **Painting Line $P_2$** | **Assembly Line $A_1$** | **Assembly Line $A_2$** |
|---|---|---|---|---|
| **Slot 1** | 1 unit XUV (2 hrs) | 2 units Sedan (3 hrs each) | 1 unit XUV (3 hrs) | 2 units Sedan (2 hrs each) |
| **Slot 2** | 1 unit Mini XUV (1 hr) | 1 unit Sedan (3 hrs) | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) |
| **Slot 3** | 1 unit XUV (2 hrs) | 1 unit Sedan (3 hrs) | 2 units XUV (3 hrs each) | 1 unit Sedan (2 hrs) |
| **Slot 4** | 2 units Mini XUV (1 hr each) | Idle | 2 units XUV (3 hrs each) | 2 units Mini XUV (2 hrs each) |
| **Slot 5** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | Idle | Idle |
| **Slot 6** | 2 units XUV (2 hrs each) | 2 units Sedan (3 hrs each) | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) |
| **Slot 7** | Idle | 1 unit Sedan (3 hrs) | Idle | 2 units Mini XUV (2 hrs each) |
| **Slot 8** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | Idle | 2 units Mini XUV (2 hrs each) |
| **Slot 9** | Idle | Idle | Idle | Idle |
| **Slot 10** | Idle | Idle | Idle | Idle |

**Mutation**

Mutation randomly alters the schedule by switching the order of tasks in one slot. Let's apply mutation to **Offspring Mutation applied on Slot 6**: Swap 2 units of Mini XUV and 1 unit of Sedan in Slot 6 of Painting Line $P_2$.

**Table 13**

| **Time Slot** | **Painting Line $P_1$** | **Painting Line $P_2$** | **Assembly Line $A_1$** | **Assembly Line $A_2$** |
|---|---|---|---|---|
| **Offspring 1 (Post-Mutation) Slot 6** | 2 units XUV (2 hrs each) | 2 units of Mini XUV | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) |

**Final Solution Table**

Here's the final solution, considering the offspring's fitness scores. After mutation, we select the offspring with the highest fitness score.

**Table 14**

| **Time Slot** | **Painting Line $P_1$** | **Painting Line $P_2$** | **Assembly Line $A_1$** | **Assembly Line $A_2$** | **Idle Time** |
|---|---|---|---|---|---|
| **Slot 1** | 1 unit XUV (2 hrs) | 1 unit Sedan (3 hrs) | 1 unit XUV (3 hrs) | 2 units Sedan (2 hrs each) | 1 hr |
| **Slot 2** | 1 unit XUV (2 hrs) | 1 unit Sedan (3 hrs) | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) | 1 hr |
| **Slot 3** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | 2 units XUV (3 hrs each) | 2 units Mini XUV (2 hrs each) | 1 hr |
| **Slot 4** | 2 units Mini XUV (1 hr each) | 2 units Sedan (3 hrs each) | Idle | 3 units Mini XUV (2 hrs each) | 3hrs ($A_1$) |
| **Slot 5** | 2 units Mini XUV (1 hr each) | Idle | Idle | Idle | 2 hrs ($P_2$) |
| **Slot 6** | 2 units XUV (2 hrs each) | 2 units Mini XUV (1 hr each) | 1 unit XUV (3 hrs) | 2 units Mini XUV (2 hrs each) | None |
| **Slot 7** | Idle | 1 unit Sedan (3 hrs) | Idle | 1 unit Mini XUV (2 hrs) | 2 hrs ($P_1, A_1$) |
| **Slot 8** | 2 units Mini XUV (1 hr each) | 1 unit Sedan (3 hrs) | Idle | Idle | 2 hrs ($A_1, A_2$) |
| **Slot 9** | Idle | Idle | Idle | Idle | 3 hrs |
| **Slot 10** | Idle | Idle | Idle | Idle | 3 hrs |

**Explanation of Idle Time:**

- Idle time is calculated based on the **unused time** for each painting and assembly line in each slot.

- **Total Idle Time** = Sum of idle hours across all time slots for all lines.

**Idle Time Calculation:**

- Slot 1: Idle time = 1 hr ($P_1$ finishes at 2 hrs, while $P_2$ runs for 3 hrs)
- Slot 2: Idle time = 1 hr ($P_1$ finishes at 2 hrs, while $P_2$ runs for 3 hrs)
- Slot 3: Idle time = 1 hr ($P_1$ finishes at 2 hrs, while $P_2$ runs for 3 hrs)
- Slot 4: Idle time = 3 hrs (Assembly Line $A_1$ is idle for the entire slot)
- Slot 5: Idle time = 2 hrs (Painting Line $P_2$ is idle for the entire slot)
- Slot 6: No idle time (Both painting and assembly lines are fully utilized)
- Slot 7: Idle time = 2 hrs ($P_1$ and $A_1$ are idle for the entire slot)
- Slot 8: Idle time = 2 hrs ($A_1$ and $A_2$ are idle for the entire slot)
- Slot 9: Idle time = 3 hrs (All lines are idle)
- Slot 10: Idle time = 3 hrs (All lines are idle)

**Total Idle Time:**

Total Idle Time$= 1 + 1 + 1 + 3 + 2 + 0 + 2 + 2 + 3 + 3 = 18$ hours

**Step-by-Step Production Time Calculation**

**Painting Time:**

- **XUV**: $10\ units\ \times\ 2\ hours\ each\ =\ 20\ hours.$
- **Sedan**: $10\ units\ \times\ 3\ hours\ each\ =\ 30\ hours.$
- **Mini XUV**: $10\ units\ \times\ 1\ hour\ each\ =\ 10\ hours.$

Total painting time = $20$ hours (XUV) $+ 30$ hours (Sedan) $+ 10$ hours (Mini XUV)
$= 60$ hours.

**Assembly Time:**

- **XUV**: $10\ units\ \times\ 3\ hours\ each\ =\ 30\ hours.$
- **Sedan**: $10\ units\ \times\ 2\ hours\ each\ =\ 20\ hours.$
- **Mini XUV**: $10\ units\ \times\ 2\ hours\ each\ =\ 20\ hours.$

Total assembly time = 30 hours (XUV) + 20 hours (Sedan) + 20 hours (Mini XUV) = **70 hours**.

**Total Production Time:**

Total production time = Total painting time + Total assembly time

Total Production Time$= 60\ hours\ (painting) + 70\ hours\ (assembly) = \mathbf{130}\ hours$

## 4.Conclusion

In this study, we addressed the complex production scheduling problem involving three car models (XUV, Sedan, and Mini XUV) with specific constraints related to model compatibility, painting line capacity, and assembly line requirements. By applying the Memetic Graph Coloring Algorithm, we effectively minimized the total production time while ensuring efficient utilization of resources. The memetic algorithm, combining graph coloring with evolutionary operators such as crossover and mutation, proved to be an effective solution for handling conflicting tasks and optimizing the scheduling process. Through the graph-based representation of tasks, we were able to assign car models to time slots on painting and assembly lines while respecting operational constraints. Mutation operations introduced necessary diversity in the scheduling process, allowing us to reduce idle time and improve the flexibility of the production schedule. Our results demonstrate that the algorithm achieved an optimized schedule for 10 units of each model, minimizing idle time and ensuring that both painting and assembly lines were utilized efficiently. This approach not only outperformed traditional scheduling methods but also showcased the potential of using memetic algorithms for solving real-world production scheduling problems. Future research could explore the integration of additional factors such as maintenance schedules, labor shifts, or dynamic demand changes, further enhancing the applicability of this approach to complex industrial scenarios.

## References

1. Morgenstern, C.A., Shapiro, H.D. Heuristics for rapidly four-coloring large planar graphs. *Algorithmica* **6**, 869–891 (1991). https://doi.org/10.1007/BF01759077
2. Aardal, K., Hoesel, S., Koster, A., Mannino, C. and Sassano, A. (2003), Models and solution techniques for frequency assignment problems, Q. J. Belg. Fr. Ital. Oper. Res. Soc., 1(4), 261-317.
3. Dib, M., Caminada, A. and Mabed, H. (2010), Frequency management in radio military networks, INFORMS Telecom 2010, 10th INFORMS Telecommunications Conference Montreal, Canada.
4. Galinier, P. and Hertz, A. (2006), A survey of local search methods for graph coloring, Comput. Oper. Res. 33, 2547-2562.
5. Malaguti, E., Monaci, M. and Toth, P. (2011), An exact approach for the vertex coloring problem, Discret. Optim., 8 (2), 174-190.
6. Malviya, A., Agrawal, B. and Kumar, S. (2023), A New approach on star chromatic number of splitting complete bipartite graph, Samdarshi, 16 (5), 3128-3132.
7. Malviya, A., Agrawal, B. and Kumar, S. (2024), Graph coloring algorithm for course time table scheduling problem, ShodhKosh: Journal of Visual and Performing Arts, 5 (3), 452-460.
8. Malviya, A., Agrawal, B., Kumar, S. and Mansuri, A. (2022), Study of algorithm for coloring in various graph, International Journal of Statistics and Applied Mathematics, 7(2), 88-91.