

Task Sequencing and Scheduling of Autonomous Mobile Robots in Flexible Manufacturing Systems Using Metaheuristic Optimization

P.V.Pramila^{1}, S.Nagarani², Karthi Vinith K S³, Sivakumar Balu⁴, A.Vijaya Mahendra Varman⁵, and M.P. Natarajan⁶*

¹Department of Computer Science Engineering, SIMATS Engineering, Chennai, Tamil Nadu 602105, India

²Department of Mathematics, Sri Ramakrishna Institute of Technology, Coimbatore, Tamil Nadu 641010, India

³Department of Automobile Engineering, Kongu Engineering College, Erode, Tamil Nadu 638060, India

⁴Independent Researcher, IEEE Senior Member, San Jose, California, USA

⁵Department of Artificial Intelligence and Data Science, Panimalar Engineering College, Chennai, Tamil Nadu 600123, India

⁶Department of Mechanical Engineering, Bharath Institute of Higher Education and Research, Chennai, Tamil Nadu 600073, India

Abstract. The research mentioned in this document deals with the issue of scheduling and path planning for autonomous mobile robots (AMRs) in a manufacturing setup where multiple robots and feeders are present. In this context, two metaheuristic methods are put forward: VNS-AMR, which is based on the Variable Neighborhood Search (VNS), and VND-AMR, which utilizes the Variable Neighborhood Descent (VND) technique. With the help of the proposed algorithms, an initial solution is generated first by a controlled greedy heuristic (HSI) and then local search procedures are applied with respect to the previously defined neighborhood structures. Compliance with constraints on release times, robot capacities, and replenishment requirements is ensured. The methods are tested on case studies which are based on data from literature and take into account various numbers of feeders, robot capacities, and subtasks. The results show that both metaheuristics are capable of finding feasible solutions in a short time, for both small and large scale problems, which is the main advantage over conventional scheduling methods, alongside reduced idle times for robots and good allocation of tasks.

* Corresponding author: pramilapv.sse@saveetha.com

1 Introduction

Efficiency in scheduling tasks and planning routes for autonomous mobile robots (AMRs) is among the main challenges in modern manufacturing systems, where the coordination of several robots is required to supply feeders, process tasks, and keep the system running efficiently. Mathematical programming approaches are often not appropriate for multi-robot situations because of the combinatorial complexity and the NP-hardness of the issue. Heuristic and metaheuristic methods have shown to be effective in such cases [1-5]. The present research puts forward two methods based on VNS and VND metaheuristics, which are dubbed VNS-AMR and VND-AMR, respectively, to optimize the assignment of tasks, the routing of robots, and the usage of robots in a manufacturing cell with multiple feeders [6-8]. The controlled greedy heuristic (HSI) is utilized for generating an initial solution, while neighborhood local search procedures are employed to improve the quality of solutions. The consideration of multiple robots, the thorough treatment of robot capacities and replenishment constraints, and the employment of multi-objective functions to simultaneously minimize total processing time and idle periods are all noteworthy contributions of the study [9, 10]. The proposed methods are validated through computational experiments based on scenarios of different levels of complexity, thus giving an idea of the feasibility and scalability of metaheuristic approaches for AMR scheduling.

The novelty of this work is: (i) integration of task sequences, routing, replenishment, and warehouse return decisions within a unified AMR scheduling paradigm; (ii) development of VNS AMR and VND AMR algorithms tailored to address feeder replenishment operations of multiple robots; (iii) introduction of various objective functions to incorporate ‘processing time,’ ‘travel time,’ ‘waiting time,’ and ‘replenishment return time’; and (iv) application of ‘feasibility-driven local search procedure,’ resulting in a substantial ‘robot idle time’ saving compared to prior AMR schedules, making this work distinguishable from prior AMR scheduling and AGV scheduling works.

2 Literature Review

A bunch of recent research works target the improvement of task scheduling in flexible job shops and multi-robot systems that involve AGVs (Automated Guided Vehicles) reinforcement learning, and so on, through the use of different combinations of methodologies. Li et al. [11] came up with a multi-strategy-driven genetic algorithm that optimized the scheduling of flexible job shops with AGVs and provided task completion times and vehicle utilization as performance measures. Ho et al. [12] experimented with the concept of federated deep reinforcement learning for task scheduling in multi-agent autonomous robot systems emphasizing the idea that distributed learning between more than one agent can improve scheduling performance and at the same time lead to system flexibility through different task requirements. Tejer et al. [13] examined the area of robot applications where reinforcement learning was applied, thus pointing at the ability of learning-based systems to managing non-static and uncertain surroundings while securing high task completion rates. Xu et al. [14] unwrapped a multi-objective green scheduling model combining flexible job shop scheduling with AGVs, power efficiency and total operational sustainability being the main concerns along with performance. Eventually, Yi and Luo [15] introduced a heuristic mechanism for robotic job shops based on Petri nets and artificial potential fields, thus giving a practical and reliable technique for non-controversial task allocation and movement in complex manufacturing areas.

3 Proposed Solution Methods

The problem is tackled with two solution methods in this section. The first one, which is based on the VNS metaheuristic, is referred to as VNS-AMR. The second one, based on the VND, is referred to as VND-AMR.

3.1 Description of the Proposed Algorithms

It is worth mentioning that a mathematical programming model was not found in the scheduling problem considering two or more robots during the bibliographic research and theoretical foundation phase. As a result, the development of a heuristic algorithm was considered whose steps are shown in the flowchart illustrated in Fig. 1 and further detailed in the coming sections.

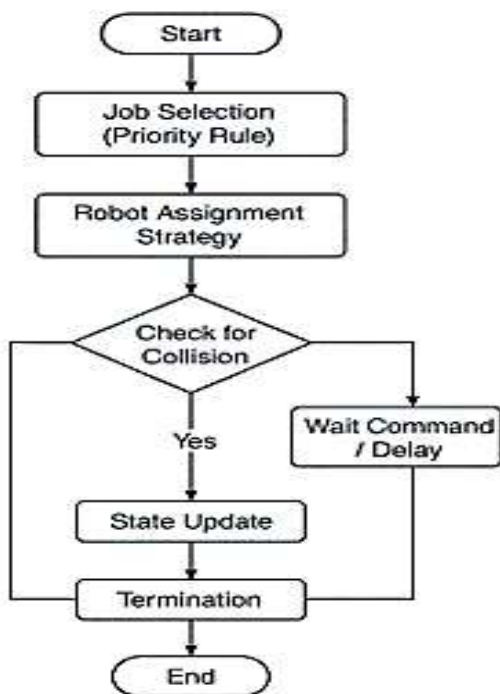


Fig. 1. Solution Method- Steps.

3.1.1 Solution Coding

In order to support the application of the suggested heuristic methods, the encoding of the solution is employed as shown in Table 1. An example of the solution is a matrix of the dimensions $(4 \times n)$, where the first line takes care of the order in which the machines that the jobs are represented by, the second line is the subtask of the respective job, the third line is the route to be executed, and last row indicates which AMR is selected. Here, n is the total number of jobs, (n) is the sum of all subtasks of tasks, r is the number of different paths, and amr is the picked robot.

Table 1. Solution Representation.

Category	Column 1	Column 2	Column 3	Column 4	Column 5	Column N
Task	0	1	4	2	3	n
Subtask	0	1	1	1	1	S(n)
Route	0	1	2	1	2	r
Robot	1,2	1	2	1	2	robot

The adoption of such representation was due to its interpretation simplicity, to the ease of solution neighborhood swapping during the search process, and to the evaluation of solution quality (FO) with minimum effort.

3.1.2 Heuristic Method for Initial Solution - HSI

A controlled greedy constructive heuristic was executed to obtain an initial solution called HSI. The HSI is responsible for conducting the initial steps of gathering the release time data for sub-task k belonging to task i (e_{ik}) and the deadline for sub-task k of task i (d_{ik}). The heuristic then, with the help of this data, takes the decision on which sub-task will be released to start the next execution [16-20]. In the case of several subtasks starting at the same time, the one with the shortest due date is chosen to be, i.e., the closest due date.

Table 2. Example of initial solution.

Task	0	1	2	3	4	1	4
Subtask	0	1	1	1	1	2	2
Route	0	1	2	1	2	3	4
Robot	1,2	1	2	1	2	1	2

An initial solution example has been presented in Table 2 where it can be seen that the two robots process the zero task in the warehouse, and afterwards the first robot does the subtask one of the task one, then it is the reverse of the robot one doing the subtask one of the task three on the same route, the robot one returns to the warehouse, takes the stock, and goes back to perform the operations on route three where it does subtask two of task one [19-22]. Robot two follows the same procedure.

3.1.3 Feasibility analysis of the initial solution

A controlled greedy constructive heuristic was executed to obtain an initial solution called HSI. The HSI is responsible for conducting the initial steps of gathering the release time data for sub-task k

Let: $i \in T$: tasks, $k \in S_i$: subtasks of task i , $r \in R$: routes, $a \in A$: autonomous mobile robots (AMRs), $x_{ik}^a = 1$ if AMR a performs subtask k of task i , st_{ik}^a : start time, ct_{ik}^a : completion time, p_{ik} : processing time, d_{ik} : deadline, e_{ik} : earliest release time, t_{uv} : travel time between feeders/warehouse, M : a large constant.

Task assignment: each subtask is performed exactly once (Equation 1),

$$\sum_{a \in A} x_{ik}^a = 1 \quad \forall i \in T, k \in S_i \quad (1)$$

Start time must respect release time (Equation 2),

$$st_{ik}^a \geq e_{ik} - M(1 - x_{ik}^a) \quad (2)$$

Completion time definition (Equation 3),

$$ct_{ik}^a = st_{ik}^a + p_{ik} \quad \forall i, k, a \quad (3)$$

Deadline constraint (Equation 4),

$$ct_{ik}^a \leq d_{ik} + M(1 - x_{ik}^a) \quad (4)$$

Precedence between subtasks of the same task (Equation 5),

$$st_{i(k+1)}^a \geq ct_{ik}^a + t_{loc(i,k),loc(i,k+1)} - M(2 - x_{i(k+1)}^a - x_{ik}^a) \quad (5)$$

Robots cannot execute two subtasks at the same time (Equation 6),

$$st_{ik}^a \geq ct_{jl}^a - M(1 - y_{(jl),(ik)}^a) \quad \text{and} \quad st_{jl}^a \geq ct_{ik}^a - My_{(jl),(ik)}^a \quad (6)$$

where y defines the sequence.

One route must be chosen per subtask (Equation 7),

$$\sum_{r \in R} z_{ik}^r = 1 \quad \forall i, k \quad (7)$$

Link between robot assignment and route (Equation 8),

$$z_{ik}^r \leq \sum_a x_{ik}^a \quad \forall i, k, r \quad (8)$$

Travel time before starting next subtask (Equation 9),

$$st_{ik}^a \geq ct_{jl}^a + t_{route(jl),route(ik)} - M(1 - x_{ik}^a) \quad (9)$$

Collision avoidance on same route segment (Equation 10),

$$st_{ik}^a \geq ct_{jl}^a - M(1 - q_{(jl),(ik)}^{ab}) \quad \text{and} \quad st_{jl}^a \geq ct_{ik}^a - Mq_{(jl),(ik)}^{ab} \quad (10)$$

Warehouse loading/unloading time (Equation 11),

$$st_{ik}^a \geq ct_{wh}^a + t_{wh,route(ik)} - M(1 - x_{ik}^a) \quad (11)$$

Robot availability (Equation 12),

$$st_{ik}^a \geq av_a \quad (12)$$

Sequence continuity: robot must finish previous task before starting next (Equation 13),

$$ct_{ik}^a + t_{route(ik),route(i(k+1))} \leq st_{i(k+1)}^a + M(1 - x_{ik}^a) \quad (13)$$

Non-negativity constraints (Equation 14),

$$st_{ik}^a \geq 0, ct_{ik}^a \geq 0 \quad \forall i, k, a \quad (14)$$

Sequence continuity (robot must finish previous task and travel before starting the next (Equation 15):

$$ct_{ik}^a + t_{route(ik),route(i(k+1))} \leq st_{i(k+1)}^a + M(1 - x_{ik}^a) \quad (15)$$

Non-negativity of time variables (Equation 16):

$$st_{ik}^a \geq 0, ct_{ik}^a \geq 0 \quad (16)$$

A practical objective for the AMR scheduling problem — consistent with the constraints and the elements described (warehouse travel, travel between feeders, processing times, waiting times, and deadline adherence) — is to minimize the total operational time (flow time) plus a weighted penalty for tardiness. This balances throughput (short total flow times) with meeting deadlines.

An auxiliary (non-negative) tardiness variable: $T_{ik} \geq 0, T_{ik} \geq ct_{ik}^a - d_{ik} \quad \forall i, k, a$ with $x_{ik}^a = 1$, then the objective (Equation (17)) is:

$$\min FO = \sum_{i \in T} \sum_{k \in S} \sum_{a \in A} x_{ik}^a (ct_{ik}^a - e_{ik}) + \lambda \sum_{i \in T} \sum_{k \in S_i} T_{ik} \quad (17)$$

Objective Function for Heterogeneous Robots (Equation 18),

$$\min FO_{het} = \sum_{i \in T} \sum_{k \in S} \sum_{a \in A} x_{ik}^a w_a (ct_{ik}^a - e_{ik}) + \lambda \sum_{i \in T} \sum_{k \in S_i} T_{ik} + \mu \sum_{i \in T} \sum_{k \in S} \sum_{a \in A} x_{ik}^a (Travel_{ik}^a + r_a^{ret}) \quad (18)$$

Where $Travel_{ik}^a = \sum_{segments(u \rightarrow v) \in path_{ik}^a} \tau_{uv}^a$ represents the total travel time robot a incurs to reach and complete subtask (i, k) along its chosen route (including warehouse \leftrightarrow feeder and feeder \leftrightarrow feeder segments), and $\mu \geq 0$ is an optional weight to explicitly penalize travel/return time (use $\mu=0$ if travel is already fully captured by start/completion times).

Prior to determining the solution's quality (FO from Equation (17)), it is critical to confirm the solution's feasibility, that is, whether it satisfies all model constraints as per Equations (1)-(16). If it does not, the solution is immediately rejected and a new attempt is made to get a solution.

If the solution is found to be feasible, then the quality of the solution will be evaluated considering the processing of all tasks and subtasks of the indicated solution task sequence. In other words, the time taken for the warehouse to and from the feeders, the time taken between feeders, the task processing time, the time spent in the warehouse and the time the robot is stopped waiting for the task release time are all taken into account.

3.1.4 Defining the neighborhoods for performing local searches

In order to carry out thereby descent searches around the working solution, three kinds of neighborhood structures (i.e., perturbation in the parameters of the solution) have been taken into consideration as follows:

Internal Insertion: The application of this structure involves taking out a task from its current position in the solution vector and placing it at a new position already determined randomly. (See Fig. 2).

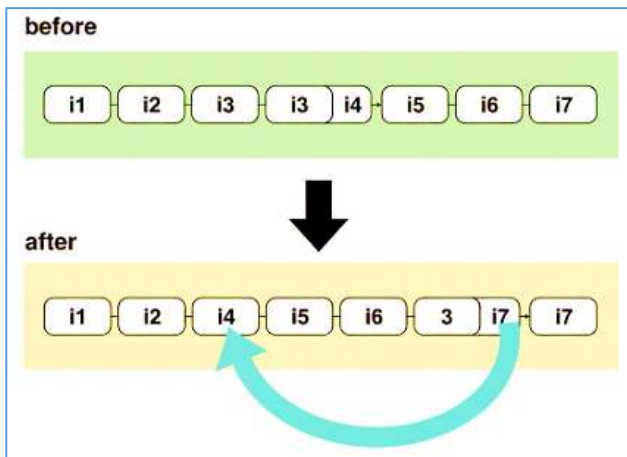


Fig. 2. Internal Insertion.

Internal Swap: The procedure of this structure includes two different tasks whose positions are swapped randomly. (See Fig. 3).

External Insertion: This method uses two different solution vectors to extract one position from each of them and insert it into a new vector, which results in a possible new solution. (See Fig. 4).

If the attempted swap satisfies the constraints imposed for the problem, the swap is carried out; otherwise, the swap is discarded, and new task positions are chosen for swapping.

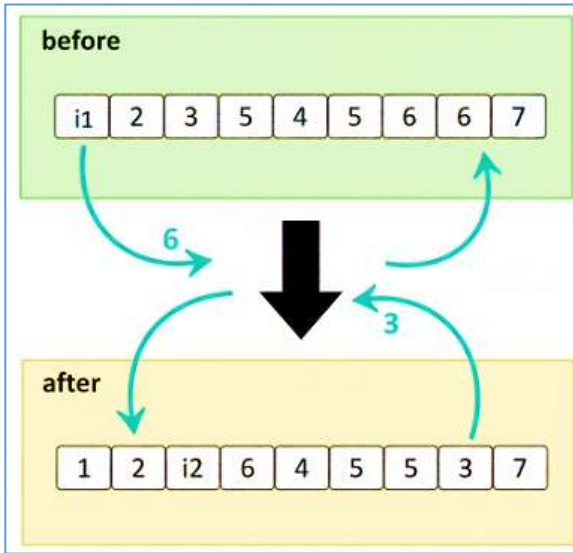


Fig. 3. Internal Swap.

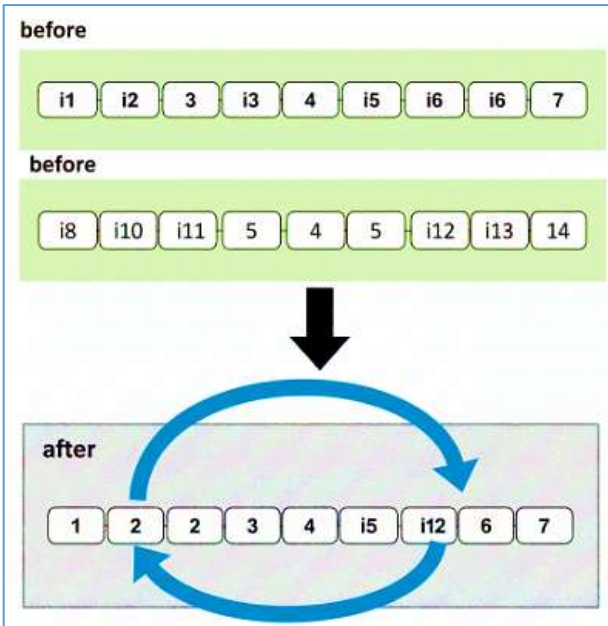


Fig. 4. External Insertion.

3.1.5 Local Search Methods

The initial solution that was produced with the HSI was the major determinant around which local searches were applied to the various neighborhood structures defined in the former section, following the VNS and VND algorithms.

In the case of the search was considered as per the VNS metaheuristic, the VNS-AMR Algorithm is obtained. And in the case of the search was considered as per the VND metaheuristic, the VND-AMR Algorithm is obtained.

If the local search is downward, i.e., all solutions in the vicinity of the current solution are examined, and a random movement is made to find a better solution, then we have VNS-AMR (Algorithm 1, Fig. 5).

```
1. function VNS-AMR(S0)
2. s ← S0;
3. k ← 1;
4. r ← 3;
5. while (k ≤ r) do
6. Generate a random solution s' in Nk(s) and perform a local search in Nk(s') and obtain s'';
7. if f(s'') < f(s) then
8. s ← s''
9. k ← 1
10. else k ← k + 1;
11. end-if;
12. end-while;
13. return s;
14. end VNS-AMR.
```

Fig. 5. Algorithm 1 - VNS-AMR Algorithm.

In case the local search is going down, meaning that all the solutions in the current solution's neighborhood are considered and a move is made to the best solution, we obtain VND-AMR (Algorithm 2, Fig. 6).

```
1. function VND-AMR(S0)
2. s ← S0;
3. k ← 1;
4. r ← 3;
5. while (k ≤ r) do
6. Find the best neighbor s' in Nk(s);
7. if f(s'') < f(s) then
8. s ← s''
9. k ← 1
10. else k ← k + 1;
11. end-if;
12. end-while;
13. return s;
14. end VND-AMR.
```

Fig. 6. Algorithm 2 - VND-AMR Algorithm.

3.1.6 Feasibility Analysis of the Solution in Local Search

In order to check the feasibility of a solution during local search in the proposed VNS-AMR and VND-AMR algorithms, aside from validating compliance with Constraints (2)-(17), a constraint was introduced to optimize the FO with the use of two robots, which assesses whether the sum of the robot's current time and the time to the warehouse for replenishment and to the feeder for the next task is less than the release time of the next task; the robot comes back to the warehouse before conducting the operation even if it still has the capacity to process the task.

The Fig. 7 gives a visual representation of the solution corresponding to the capacitor robot 3 SLC in Table 2. Robot 1, in red on the diagram, takes the task of subtask 1 in task 1; afterwards, it verifies if it has enough capacity for the next operation and if the return time to the warehouse—its replenishment time plus the return time—is less than the release time of the operation. The answer is no, so the operation is executed immediately afterward. Robot 1, however, during the next operation, has a long release time, hence he goes back to the warehouse, restocks the supply, and is on his way to do the processing of route 3, and is thereby ready to do a batch of operations if their release times are near.

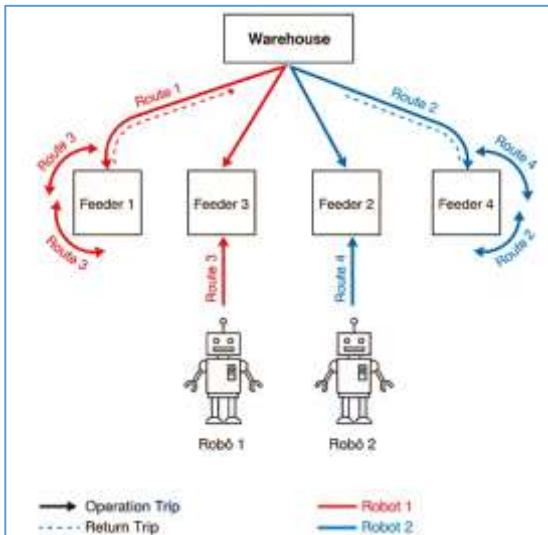


Fig. 7. Representation of the solution.

Robots' idle time, as seen in the example depicted in Fig. 7, will be significantly reduced, which leads to limiting the solution space, making it so that the robot is always supplied and can perform operations in the subtasks of the tasks that are either very close to or equal to its release time.

3.2 Highlight of the main contributions of the proposed model

The present paper proposed two objective functions, one for homogeneous robots expressed in Equation 17, and the other for heterogeneous robots expressed in Equation 18, which take into account not only the robot's travel time between machines but also working time and the return time to the warehouse. Furthermore, the number of robots participating in the case study is another contribution; while Dang et al. (2013) consider only one robot, this study includes two robots. In this scenario, the decision of which robot to use for the tasks is based on the robot's release time, that is, the first initial task in the warehouse represents the supply

of the robots, so that they are able to execute the operations, and it is done by both robots at the same instant in time, the next task is performed by robot 1, and from then on the robot with the shortest operating time is selected to carry out the next task.

The following section reveals the data employed in this research, together with the outcomes of the application of the proposed solution techniques.

4 Results

The sources of the data that are examined in this research are the ones that were published in Dang et al. (2013). The data show the maximum and minimum part levels on all feeders and their corresponding feed rates, as seen in Table 3.

Table 3. Part levels, feed rate and working time.

Location	Ware house	1	2	3	4
Maximum part level	–	250	2000	2000	250
Minimum part level	–	125	900	900	125
Feeding rate (s/part)	–	4.5	1.5	1.5	4.5
Robot working time (s)	90	42	42	42	42

The information from Table 3 allows the calculation of the periodicity, the release time of subtask k of task i , and the deadline of subtask k of task i via Equations (19), (20), and (21), respectively.

$$P_i = \begin{cases} \frac{1}{\Delta_i-1} \sum_{h=1}^{\Delta_i-1} \theta(c_i^{h+1}) - \theta(c_i^h), & \Delta_i \geq 2, \\ P_i^{(default)}, & \Delta_i = 1, \end{cases} \quad (19)$$

where $P_i^{(default)}$ is a user-specified period when the task appears only once in the horizon (for example, the planning horizon length or an externally given nominal period).

$$e_{ik} = \theta(c_{ik}) + \rho_{ik} \quad (20)$$

$$d_{ik} = e_{ik} + \sigma_{ik} \quad (21)$$

Table 4. Travel time (seconds) of the robot between locations.

Location	Ware house	1	2	3	4
Warehouse	0	34	37	34	40
Feeder 1	39	0	17	34	50
Feeder 2	35	17	0	35	49
Feeder 3	34	33	35	0	47
Feeder 4	36	47	48	46	0

Now, these time parameters put the time windows into place. The last row of Table 4 points out the times when the robot is working in the feeder and the central warehouse.

The second Table 5 gives a view of the robot's travel times in seconds between the various origin-destination pairs inside the factory.

Table 5. Results obtained by the proposed method.

Scenario	Number of Feeders	Robot Capacity	Total Subtasks	VND (s)	VNS Worst (s)	VNS Best (s)	VNS Average (s)
1	2	2	4	604.50	604.50	604.50	604.50
2	2	3	4	604.50	604.50	604.50	604.50
3	3	2	6	1,167.00	1,167.00	1,167.00	1,167.00
4	3	3	6	1,167.00	1,167.00	1,167.00	1,167.00
5	4	2	8	1,167.00	1,167.00	1,167.00	1,167.00
6	4	3	8	1,167.00	1,167.00	1,167.00	1,167.00
7	4	2	10	1,729.50	1,729.50	1,729.50	1,729.50
8	4	3	10	1,729.50	1,729.50	1,729.50	

In Table 5, the different experimental setups are listed along with the results in seconds for each method and each case. A solution is deemed feasible only if the feeder is fully supplied; if that is not the case, the solution is not considered at all. The identical objective values for Table 5 are due to the high quality of the solutions produced by the controlled greedy heuristic, HSI, which, in many cases, converges to nearoptimal schedules for deterministic instances. In order to assess the true improvement capability of the proposed metaheuristics, further experiments were conducted using randomized feasible initial solutions.

Operations for the same total number of subtasks could be different, for example, in Scenario 8, with 3 subtasks to finish task 1, 2 subtasks for task 2, 2 subtasks for task 3, and 3 subtasks for task 4, which is a total of 10 subtasks. Might also be thought of as 4 subtasks to finish task 1, 1 subtask for task 2, 1 subtask for task 3, and 4 subtasks for task 4, making the total 10 subtasks altogether. The last case was taken into account during the preparation of the paper.

Interestingly, the method put forward in this research uses 2 robots and 2 metaheuristics for enhancement. It can be seen that the metaheuristic values are the same; this is because of the quality of the initial solution, which selects the subtask with the least release time.

For the purpose of checking the quality of the metaheuristics, it was decided to construct a controlled random heuristic that would produce a feasible initial solution. The results of the objective function values of the metaheuristics are shown in Table 6. To construct Table 6 iterations were used which led to an average difference of 1.25% in task completion times compared to the studied scenarios. As depicted in Table 6, in all cases, VNS-AMR and VND-AMR improve upon random initial solutions, where, on average, we obtain a reduction of about 1.25% in total completion time. More, VND-AMR shows faster convergence, while VNS-AMR shows better robustness via diversification of solutions.

Table 6. Results obtained by the proposed method with random heuristic.

Scenario	Number of Feeders	Robot Capacity	Total Subtasks	VND (s)	VNS Worst (s)	VNS Best (s)	VNS Average (s)
1	2	2	4	604.50	692.50	604.50	628.50
2	2	3	4	604.50	680.50	604.50	612.10
3	3	2	6	1,167.00	1,257.00	1,167.00	1,190.90
4	3	3	6	1,167.00	1,257.00	1,167.00	1,181.90
5	4	2	8	1,167.00	1,243.00	1,167.00	1,186.40
6	4	3	8	1,167.00	1,243.00	1,167.00	1,180.50
7	4	2	10	1,729.50	1,729.50	1,729.50	1,729.50
8	4	3	10	1,729.50	1,729.50	1,729.50	

The different scenarios include alterations in the number of feeders, the robots' capacity and the total number of subtasks to be completed. The orange line in Fig. 8 indicates the VND-AMR results while the blue line shows the average of VNS-AMR results.

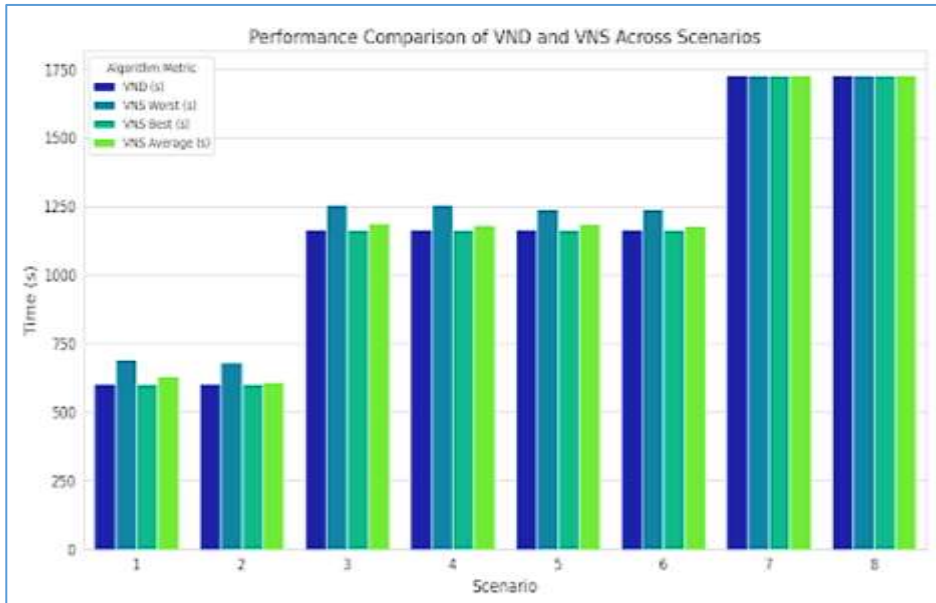


Fig. 8. Results obtained by the proposed method.

One can clearly see an exponential relationship in Fig. 8, which states that with a rise in the number of subtasks, there is an increase in the time taken for processing them.

They are classified into two categories according to the maximum capacity utilized. The first category regarded the maximum capacity with two SLCs and the second one referred to the maximum capacity considering three SLCs, as explained in Tables 5 and 6. After that, each scenario was broken down into cases with varying numbers of tasks and subtasks. The

planning time of study T was set to around 2,700 seconds (i.e., 45 minutes), limited by the robot's battery power. Imagine a scenario where the case study is prolonged, four feeders remain, and the results of the viable solutions in Table 7 are obtained.

Table 7. Case study extension.

Number of Feeders	Robot Capacity	Total Subtasks	VND (s)	VNS Worst (s)	VNS Best (s)	VNS Average (s)
4	2	16	2854.50	2944.50	2854.50	2897.65
4	3	16	2854.50	2942.50	2854.50	2884.40
4	2	32	6229.50	6319.50	6229.50	6278.60
4	3	32	6229.50	6319.50	6229.50	6260.80

The results shown in Table 5 indicate clearly that the proposed metaheuristic combining VND and VNS is capable of locating a feasible solution even for a large-scale problem, while using PLIM does not even allow the problem to be solved due to its NP-Hard nature. Moreover, the growing number of subtasks that comes with increasing problem size leads to extended solution time for the proposed metaheuristic. The conclusion of this research study, together with the chosen models, the used data, and the future work suggestions on the topic presented are unfolded in the next section.

Exact optimization approaches, such as mixed-integer linear problems, may no longer be computationally affordable for large-scale problems in the optimization of the AMR schedule due to the problems' NP-hardness. In such scenarios, the proposed approaches of VNS-AMR and VND-AMR can yield good and feasible solutions within a reasonable CPU time.

While the approach under discussion claims no superiority over exact solution methods, it rather presents itself as a heuristic approach that can be implemented on a larger scale within actual manufacturing environments, in which optimality is not as critical as viability.

As a basis for comparing performance against a benchmark method, the randomized greedy heuristic was used for generating initial schedules. The performance using this method is shown in Table 6.

The proposed VNS-AMR and VND-AMR approaches have been found to be better than the random heuristic, as they have resulted in not only lesser CPU execution time but also stability of the generated schedule.

Both have low computing overhead. In fact, for small and medium-scale problems, convergence for these two meta-heuristics takes place in a few seconds, whereas for larger-scale problems, complete convergence takes place well within the planning horizon for these meta-heuristics, which is 2700 seconds.

VND-AMR has faster convergence speed owing to the best improvement strategy employed, whereas VNS-AMR covers greater solution space, giving quicker robustness results but slightly increasing the overall running time. The convergence speed can be shown through Fig. 8:

4.1 Limitations

This model is considered to hold the assumptions for deterministic time taken for the travel and processing. This is because there are no stochastic disturbances from failures or additional task arrival considered in the proposed model. Also, the experimental validation

is considered for the benchmark problem from literature, but the real-time implementation of the proposed solution has not yet considered for an industrial experiment.

5 Conclusions

In this paper, a new method for multi-robot task scheduling and path planning was presented through the use of the metaheuristics VNS-AMR and VND-AMR. The suggested system can manage a number of robots very well, taking into consideration the limitations on their capacity and the time when the tasks become available. It does all this while still having task execution that is both feasible and optimized. Computational results show that the two metaheuristics, VNS-AMR and VND-AMR, are able to come up with solutions of high quality even in case of large-scale problems where exact optimization methods fail due to NP-hardness. The use of the controlled greedy heuristic for solution initialization has a significant positive effect on performance because it reduces initial idle times and makes task allocation more effective. The paper points out that for multi-robot scheduling models it is crucial to take replenishment times and travel distances into account in order to get realistic and implementable solutions. Further research could investigate the combination of the proposed methods with real-time adaptive scheduling, heterogeneous robot fleets, and energy-efficient operations as ways of further increasing the applicability of the methods in industrial settings.

References

1. Z. Lin, P. Ding, J. Li, Task scheduling and path planning of multiple AGVs via cloud and edge computing. *Proc. IEEE Int. Conf. Netw. Sens. Control* 1, 1–6 (2021).
2. R. Liaqait, S. Hamid, S. Warsi, A. Khalid, A critical analysis of job shop scheduling in context of Industry 4.0. *Sustainability* 13, 7684 (2021).
3. H. Al-Momani, K. M. Al-Aubidv, Fuzzy-based task scheduling of mobile robots in flexible manufacturing systems. *Proc. Int. Multi-Conf. Syst. Signals Devices*, 565–571 (2020).
4. H. Xiong, S. Shi, D. Ren, J. Hu, A survey of job shop scheduling problem: The types and models. *Comput. Oper. Res.* 142, 105731 (2022).
5. N. Senthil Kannan, R. Parameshwaran, P. T. Saravanakumar, P. M. Kumar, M. L. Rinawa, Performance and quality improvement in a foundry industry using fuzzy MCDM and lean methods. *Arab. J. Sci. Eng.* 47, 15379–15390 (2022).
6. B. Fu, W. Smith, D. Rizzo, M. Castanier, M. Ghaffari, K. Barton, Robust task scheduling for heterogeneous robot teams under capability uncertainty. *IEEE Trans. Robot.* 39, 1087–1105 (2023).
7. T. Ravichandra, P. Murugeswari, M. Revathi, S. Senthilkumar, D. S. Rathore, M. Sudhakar, Future of machine learning and robotics in digital technology for hospitality. In: *Cutting-Edge Technologies for Business Sectors*; IGI Global, pp. 401–428 (2023).
8. J. Liu, B. Sun, G. Li, Y. Chen, An integrated scheduling approach considering dispatching strategy and conflict-free route of AMRs in flexible job shop. *Int. J. Adv. Manuf. Technol.* 127, 1979–2002 (2023).
9. R. Chitharaj, H. Perumal, M. Almeshaal, P. Manoj Kumar, Optimizing performance of a solar flat plate collector using Box–Behnken design. *Sustainability* 17, 461 (2025).
10. Y. J. Yao, Q. H. Liu, X. Y. Li, L. Gao, A novel MILP model for job shop scheduling problem with mobile robots. *Robot. Comput. Integr. Manuf.* 81, 102506 (2023).

11. W. Li, H. Li, Y. Wang, Y. Han, Optimizing flexible job shop scheduling with automated guided vehicles using a multi-strategy-driven genetic algorithm. *Egypt. Inform. J.* 25, 100437 (2024).
12. T. M. Ho, K.-K. Nguyen, M. Cheriet, Federated deep reinforcement learning for task scheduling in heterogeneous autonomous robotic system. *IEEE Trans. Autom. Sci. Eng.* 21, 528–540 (2024).
13. M. Tejer, R. Szczepanski, T. Tarczewski, Robust and efficient task scheduling for robotics applications with reinforcement learning. *Eng. Appl. Artif. Intell.* 127, 107300 (2024).
14. G. Xu, Q. Bao, H. Zhang, Multi-objective green scheduling of integrated flexible job shop and automated guided vehicles. *Eng. Appl. Artif. Intell.* 126, 106864 (2023).
15. S. Yi, J. Luo, Heuristic scheduling for robotic job shops using Petri nets and artificial potential fields. *IEEE Trans. Autom. Sci. Eng.* 22, 7556–7568 (2025).
16. J.S. Prasath, V.I. Shyja, P. Chandrakanth, B.K. Kumar, A. Raja Basha, An optimal secure defense mechanism for DDoS attack in IoT network using feature optimization and intrusion detection system. *J. Intell. Fuzzy Syst.* 46, 6517–6534 (2024).
17. K. B. Prakash, A. Amarkarthik, M. Ravikumar, P. Manoj Kumar, S. Jegadheeswaran, Optimizing performance characteristics of blower for combustion process using Taguchi based grey relational analysis. In: *International Conference on Advances in Materials Research*, pp. 155–163 (2019).
18. N. Subramanyam, C.V. Kumar, A.R. Reddy, P. Chandrakanth, Image diagnosis using CNN deep learning model. In: *AIP Conference Proceedings*, vol. 2802, p. 120005. AIP Publishing LLC (2024).
19. C. Lakshmanpriya, A. Kumaravel, M. Saravanan, P. Manoj Kumar, Selecting the optimal green supplier and order allocation under linear discount. *Math. Probl. Eng.* 2453703 (2022).
20. K. R. Kunduru, Y. D. Dwivedi, R. Aruna, G. R. Thippeswamy, S. Selvakumar, M. Sudhakar, Elevating performance for enhancing AI-powered humanoid robots through innovation. In: *Applied AI and Humanoid Robotics for the Ultra-Smart Cyberspace*; IGI Global, pp. 85–119 (2023).
21. M. Balaji, S. N. Dinesh, S. V. Vetrivel, P. M. Kumar, R. Subbiah, Augmenting agility in production flow through ANP. *Mater. Today Proc.* 47, 5308–5312 (2021).
22. T. R. Saravanan, S. Suvitha, D. Banavath, S. Gogula, J. Upadhyay, M. Sudhakar, AI and machine learning integration in medical assistive robotics. In: *Fostering Cross-Industry Sustainability with Intelligent Technologies*; IGI Global, pp. 130–151 (2023).