

SCHEDULING AUTONOMOUS MOBILE ROBOTS IN FLEXIBLE MANUFACTURING ENVIRONMENT USING VARIABLE NEIGHBORHOOD METAHEURISTICS

Roberto Carlos Vieira

Federal University of Santa Catarina

Rua Dona Francisca, 8300, Bloco U, Zona Industrial Norte, CEP: 89.219-600, Joinville, SC,
Brasil

rober7o.carlos.v@gmail.com

Silvia Lopes de Sena Taglialenha

Federal University of Santa Catarina

Rua Dona Francisca, 8300, Bloco U, Zona Industrial Norte, CEP: 89.219-600, Joinville, SC,
Brasil

s.taglialenha@ufsc.br

SUMMARY

The pursuit of increased efficiency in large-scale automated production systems, particularly in cases where manual process adaptability is possible, has prompted companies to embrace automation technologies and autonomous mobile robots (AMRs). Companies are adopting flexible manufacturing systems (FMS) in conjunction with AMRs to handle complex or repetitive tasks, such as material transportation and performing of manufacturing operations. In this context, the problem addressed in this paper is the scheduling of tasks to AMRs in a flexible manufacturing environment. The robots are responsible for supplying machine feeders to prevent interruptions in operations caused by a shortage of parts. A mixed-integer linear programming model is proposed to achieve optimal solutions for small instances of the problem, with the objective of minimizing the makespan. The problem is classified as NP-Hard, and two computational methods based on Variable Neighborhood Search and Variable Neighborhood Descent metaheuristics were presented. Computational experiments are also conducted, and the results indicate that the proposed approach can be successfully applied to more complex examples.

KEY WORDS. Autonomous mobile robots scheduling. Makespan. VNS/VND-AMR Metaheuristic.

1. Introduction

The inception of Automated Guided Vehicles (AGVs) can be traced back to 1953 when Barret Electronics of Northbrook, Illinois, USA introduced the first documented AGV. Since then, AGVs have gained widespread employment for streamlining intralogistics processes [Ullrich 2014]. In contemporary industrial production scenarios, the integration of robots with human-operated tasks is progressively expanding in shared workspaces. Over the recent decades, AGVs and, in certain contexts, automated mobile robots (AMRs) have provided increased flexibility within manufacturing environments.

The pursuit of increased efficiency in large-scale automated production systems, particularly in cases where manual process adaptability is possible, has prompted companies to embrace automation technologies and autonomous mobile robots [Thoben et al. 2017]. The central aim of production planning is to optimize manufacturing procedures, efficiently allocate resources, and account for operational constraints such as production processes and capacity limitations specific to the company [Silver et al. 1998]; [Shen et al. 2006]. Given the paramount importance of efficiency and resource optimization in the industrial field, improving service quality, streamlining processes, and reducing costs are essential factors for a company's consolidation. Consequently, the body of research concerning the utilization of AGV and AMRs has shown substantial growth [Dang et al. 2014]; [Riazi and Lennartson 2021]; [Fernandes and Tagliailenha 2021]; [Manafi et al. 2022]; [Vieira and Tagliailenha 2022]; [Pattnaik et al. 2022]; [Tomasson et al. 2022].

Scheduling is a critical phase in production planning problems. As proposed by Wight [Wight 1984, the objective is to minimize the aggregate processing time across machines while ensuring optimal resource allocation. In the general context, the scheduling problem is a fundamental decision-making process for companies. It involves the assignment of tasks and machines with the overarching objective of optimizing performance metrics such as delay time, lead time, and process time (Wight 1984). It is common to use the concept of scheduling with AMRs or AGVs to assign tasks, determine their sequence, and establish initiation and completion times [Dang et al. 2014]; [Fernandes and Tagliailenha 2021]; [Riazi and Lennartson 2021]; [Vieira and Tagliailenha 2022]. An efficient schedule aims to complete tasks in the shortest possible time or arrange them in a way that minimizes overall execution time, while still considering the constraints inherent in the problem [Dang et al. 2014].

Within this context, the present paper addresses the challenge of scheduling AMRs (Scheduling AMR problem – SAMR) for part-feeding tasks within a versatile manufacturing setting. The proposed approach is adaptable to any conveyance mechanism responsible for moving parts. Drawing from the framework outlined by [Dang et al. 2014], we introduce two novel objective functions that take into consideration work time and robot return time to the warehouse. These new metrics aim to minimize the overall task processing time (makespan). One objective caters to homogeneous robots, while the other caters to heterogeneous robots. The objective of the paper also includes the development of metaheuristic s based on Variable Neighborhood Descent (VND-AMR) and Variable Neighborhood Search (VNS-AMR) methodologies, aimed at efficiently solving the specified problem.

2. Problem Description

The problem studied in [Dang et al. 2014] involves autonomous mobile robots with mechanical arms that perform tasks related to transporting, collecting, and feeding machines with small, standardized boxes of parts (SLC - Small Loaded Carriers) in a production line structured with parallel machines. To perform the task of feeding parts, this study considers an AMR, which should be programmed to handle the feeding process while avoiding part shortages in the production line. The AMR should also adhere to the restrictions imposed by the machines' parts storage capacity, loading capacity, and battery autonomy.

The production line environment considered by [Dang et al. 2014] is composed of four machine feeders and a warehouse. It is worth noting that different machines should be fed with distinct parts. The production line environment is divided into manufacturing cells with two production lines: the first line is between feeders 1 and 2, and the second line is between feeders 3 and 4. The AMRs

are responsible for performing the tasks of positioning the boxes on the machines and supplying the machine feeders with their respective parts.

In addition, we assume that the warehouse operator places the parts into the SLCs, which are stored in the central warehouse. During the process, the AMR takes a specified quantity of SLCs from the warehouse and transports them to the feeder of the designated machine to unload the parts. The robot then moves to the warehouse to unload the empty SLCs, reload them, and start over again until all tasks are completed. When evaluating the execution of activities by a single robot, the concept of SAMR (Scheduling a Single AMR) arises, as mentioned by [Dang et al. 2014].

2.1 Mathematical Formulation for SAMR

In [Dang et al. 2014] authors present a mathematical approach for scheduling a single robot. The authors assume that all information about the tasks, such as the supply of parts to the feeders, are known in advance, as well as the load capacity of the AMR.

To ensure that the machines comply with the storage capacity and to prevent parts shortages during the replenishment period, time limits were established for the feeding activities. The limitations were based on the inventory control system (s, Q), taking into consideration the processing time of the machines.

Thus, in this procedure, when the machine part stock reaches a certain level v_i , a replenishment request Q is made. In addition, the load on the robot, its speed, and the rate at which parts are fed to the machines were also considered. From this information, the calculation of the periodicity is performed to determine when the replacement of parts should be made based on stock levels. In other words, the closer the stock level is to the periodicity value, the lower the stock level is.

The goal is to minimize the total travel time of an AMR during a specific planning period, considering the following assumptions:

- the AMR starts in the warehouse;
- load capacity of the AMR is known;
- tasks are periodic, independent, and assigned to the same AMR;
- supply rate of parts from the feeder to the machine is known and constant;
- working and travel time of the AMR between each pair of locations is known;
- feeders must be loaded to the maximum supported parts limit.

The SAMR problem is considered NP-hard. Therefore, as the size of the problem increases, so does the complexity of solving it. Consequently, in practice, the application of the MIP (Mixed Integer Linear Programming) model is restricted to a smaller scale, using only a few feeders on the production line, a limited number of subtasks, and a reduced planning horizon [Dang et al. 2014].

The input parameters of the model are based on the following notations:

- N : task group, ($N = 0, 1, 2, \dots, n$) where 0 represents the task in the warehouse;
- n_i : number of times the task i needs to be executed;
- R : set of possible routes ($R = 1, 2, \dots, Rmax$; $Rmax = \sum n_i, \forall i \in N \setminus \{0\}$);
- e_{ik} : time when the subtask k task i task is released, in seconds;
- d_{ik} : subtask's deadline for completion k of the task i in seconds;
- p_i : task periodicity time i in seconds;
- w_i : AMR processing time at location i in seconds;
- t_{ij} : AMR travel time from location i location to location j in seconds;
- c_i : part feed rate from feeder to machine in seconds i to machine, in seconds;
- v_i : minimum volume of pieces units in the feeder i ;
- u_i : maximum volume of pieces units in the feeder i ;
- Q_i : maximum number of SLC that the AMR can load;
- T : planning horizon, in seconds;

The decision variables used in the model are:

- $X_{ik}^{jlr} = 1$, if the AMR travels from task location i subtask k to the location of task j , subtask l , on route r ; 0 otherwise;
- Y_{ik} : index of the route in which subtask k of task i is inserted;
- S_{ik} : Starting moment of subtask k of the task.

Since the tasks on the feeders have cyclical characteristics, it is necessary to determine their frequency, which can be obtained using Equation (1). This results to a certain number of subtasks, which are defined by the equation $n_i = \lceil T/i \rceil$.

The robot must start executing a subtask k , belonging to task i within a time interval $[e_{ik}, d_{ik}]$, specified for that subtask. The robot cannot perform task i after the upper bound of the time interval. If the robot arrives at the location of task i before the lower time of the time interval e_{ik} , it must wait until it can start the service.

The lower limit of the time window, or the release time, is defined as the time when the number of parts inside the feeder reaches the level v_i and is provided by Equation (2). The upper limit of the time window of subtask k , from task i , on the other hand, is defined as the time when there are no parts in the feeder and is represented by Equation (3).

$$p_i = (u_i - v_i) * c_i, \forall i \in N \setminus \{0\} \quad (1)$$

$$e_{ik} = e_{(ik-1)} + p_i, \forall i \in N \setminus \{0\}, k \in 1, 2, \dots, n_i, e_{i0} = 0 \quad (2)$$

$$d_{ik} = e_{ik} + (v_i * c_i), \forall i \in N \setminus \{0\}, k = 1, 2, \dots, n_i \quad (3)$$

The MIP model is given by Equations (4)-(17), where objective function is given in Equation (4).

$$\text{Minimize } Z = \sum_{i \in N} \sum_{k=1}^{n_i} \sum_{j \in N} \sum_{l=1}^{n_j} \sum_{r \in R} t_{ij} * X_{ik}^{jlr} \quad (4)$$

Subject to:

$$e_{ik} < S_{ik} < d_{ik}, \forall i \in N \setminus \{0\}, k \in 1, 2, \dots, n_i \quad (5)$$

$$\sum_{j \in N \setminus \{0\}} \sum_{l=1}^{n_j} X_{01}^{jlr} = 1 \quad (6)$$

$$\sum_{j \in N \setminus \{0\}} \sum_{l=1}^{n_j} \sum_{r \in R} X_{01}^{jlr} \leq 1 \quad (7)$$

$$\sum_{(i,k),(j,l) \in Z} X_{ik}^{jlr} \leq |Z| - 1, \forall r \in R, \forall Z \in Z_T, Z_T = \{(i,k) | i \in N \setminus \{0\}, k \in 1, 2, \dots, n_i\} \quad (8)$$

$$\sum_{j \in N} \sum_{l=1}^{n_j} \sum_{r \in R} X_{ik}^{jlr} = 1, \forall i \in N \setminus \{0\}, k \in 1, 2, \dots, n_i \quad (9)$$

$$\sum_{i \in N} \sum_{k=1}^{n_i} \sum_{r \in R} X_{ik}^{jlr} = 1, \forall j \in N \setminus \{0\}, k \in 1, 2, \dots, n_j \quad (10)$$

$$\sum_{i \in N} \sum_{k=1}^{n_i} \sum_{j \in N \setminus \{0\}} \sum_{l=1}^{n_j} X_{ik}^{jlr} \leq Q_m, \forall r \in R \quad (11)$$

$$S_{ik} + \left(w_i + t_{ij} \sum_{r \in R} X_{ik}^{jlr} \right) - L \left(1 - \sum_{r \in R} X_{ik}^{jlr} \right) + (Y_{jl} - Y_{ik})(t_{i0} + w_0 t_{0j} - t_{ij}) \quad (12)$$

$$\leq S_{jl} \quad \forall i, j \in N, k \in 1, 2, \dots, n_i, l \in 1, 2, \dots, n_j$$

$$Y_{jl} = \sum_{i \in N} \sum_{k=1}^{n_i} \sum_{r \in R} r * X_{ik}^{jlr}, \forall j \in N \setminus \{0\}, l \in 1, 2, \dots, n_j \quad (13)$$

$$Y_{jl} \geq Y_{ik} \sum_{r \in R} X_{ik}^{jlr}, \forall i, j \in N, k \in 1, 2, \dots, n_i, l \in 1, 2, \dots, n_j \quad (14)$$

$$X_{ik}^{jlr} \in \{0, 1\}, \forall r \in R, \forall i, j \in N, k \in 1, 2, \dots, n_i, l \in 1, 2, \dots, n_j \quad (15)$$

$$Y_{ik} : \text{positive integer } \forall i \in N \setminus \{0\}, k \in 1, 2, \dots, n_i \quad (16)$$

$$S_{ik} : \text{non-negative } \forall i \in N \setminus \{0\}, k \in 1, 2, \dots, n_i \quad (17)$$

The objective function defined in Equation (4) minimizes the total time of the robot's displacement between the machines. It is noteworthy that, in this model, the robot's working time at each machine and the number of operations at the feeders in each machine are priorly known. The set of Constraints (5) guarantees that the start of the processing time of subtask k , from task i , respects the lower and upper bounds of the time window. Constraints (6) will cause the robot to start the problem in the warehouse, while the set of Constraints (7) will cause the initial stage of Constraint (6) to occur only once.

Constraints (8) guarantee the elimination of sub routes among the subtasks, where Z is a subset of Z_T , and Z_T is the set of all the subtasks. Constraints (9) and (10) guarantee that only one subtask of a task will be performed immediately after and right before subtask k of task i , respectively. Constraints (11) ensure that the robot does not load more SLC s than the permitted capacity.

Constraints (12) guarantee the travel times between locations, the same constraint ensures that the start time of subtask k from task i is not greater than that of subtask i from task j , if they are carried out in sequence. L is a positive very large constant. Constraints (13) assign a subtask of a task to a route r . Constraints (14) warrants the indices of the routes used to increase sequentially. And finally, Constraints (15), (16), and (17) indicate the domain of the variables.

2.2 Mathematical modeling to consider multiple AMRs

To enable the use of more than one robot in the execution of the tasks, apart from the premises mentioned in the previous section, it was necessary to add a few more parameters:

- N_p : Number of robots;
- N : Group of all tasks;
- w_i : Process time of robot v at location i .

Moreover, with the overarching goal of minimizing the aggregate time required for task execution by robots, this work considers the MIP model given by Equations (1)-(17). However, a significant modification is introduced to the objective function outlined in Equation (4), which is subsequently replaced by Equation (18). This revised objective function not only incorporates the travel time of the robots between the machines, but also considers the work time and the return time to the warehouse of each robot.

$$\text{Min } Z = \sum_{i \in N} \sum_{k=1}^{n_i} \sum_{j \in N} \sum_{r \in R} (w_i + t_{ij}) X_{ik}^{jlr} \quad (18)$$

The first portion of Equation (18) represents the makespan when added to the second portion, resulting to the total time to process all the tasks of the study in question.

Considering practical situations where the robot processing time is different, we define a new FO given by Equation (19), considering the new parameter w_i^v and the decision variable defined as follows:

- w_i^v : Process time of robot v at location i ;
- $X_{ikv}^{jlr} = 1$ if robot v travels from task location i , subtask k , to task location j , subtask l , on route r ; and $X_{ikv}^{jlr} = 0$, otherwise.

$$\text{Min } Z = \sum_{i \in N} \sum_{k=1}^{n_i} \sum_{j \in N} \sum_{r \in R} \sum_{v \in N_v} (w_i^v + t_{ij}) X_{ikv}^{jlr} \quad (19)$$

2.3 Other Formulations

In [Riazi and Lennartson 2021] the authors developed a mathematical model to solve the problem of conflict-free routing of AGVs. The proposed model is divided into a main problem, which involves task assignment and sequencing, and a sub-problem, which aims to assess the feasibility of solving the main problem. Because it is a complex problem, the authors considered small instances ranging from 8 to 12 tasks and 4 to 12 VFAs. An algorithm was implemented in Python using the Z3 and IBM ILOG CP solvers' bindings. According to the results presented by the authors, both solvers are effective in solving the proposed models, and the heuristic achieves high-quality solutions quickly.

In [Manafi et al. 2022], the AGV scheduling problem in a manufacturing environment is considered. The objective of the work is to minimize the makespan. The authors present an MIP model for solving the problem and propose a coral reef optimization algorithm that combines centroid opposition with a heuristic to address large-scale problems. The developed heuristic utilizes Dijkstra's algorithm to extract collision-free paths. To evaluate the results of the proposed algorithm, computational experiments are designed using various random instance problems.

3. Proposed Solution Method

In this section, a two-step approach is presented. Both are based on Variable Neighborhood Search (VNS) metaheuristics and the Variable Neighborhood Descent (VND) [Mladenović and Hansen 1997]. The first method, called VNS-AMR, and the second one, called VND-AMR, are summarized in Figure 1.

Figure 1 - VNS-AMR and VND-AMR algorithm

a)	b)
1 Function VNS-AMR(S_0)	1 Function VND-AMR(S_0)
2 $S \leftarrow S_0$;	2 $S \leftarrow S_0$;
3 $k \leftarrow 1$;	3 $k \leftarrow 1$;
4 $r \leftarrow 3$;	4 $r \leftarrow 3$;
5 while ($k \leq r$) do	5 while ($k \leq r$) do
6 Take a random solution S' from $N(S)$ and	6 Find the best neighbor S'' from $N(S)$ and do
do a local search on $N(S')$ and get S''	a local search on $N(S')$ and get S''
7 If $f(S'') < f(S)$ then	7 If $f(S'') < f(S)$ then
8 $S \leftarrow S''$	8 $S \leftarrow S''$
9 $k \leftarrow 1$	9 $k \leftarrow 1$
10 else $k \leftarrow k + 1$;	10 else $k \leftarrow k + 1$;
11 end-if;	11 end-if;
12 end-while;	12 end-while;
13 Return S ;	13 Return S ;

Source: Authors 2023

The fundamental idea of VNS/VND is based on a basic principle: to explore the space of solutions by systematically changing neighborhood structures during the search process. Thus, the transition through the search space of the problem is always accomplished with an improvement of the objective function. Therefore, the transition does not allow for a solution of worse quality, as is

the case with most metaheuristics that accept degradation of the current solution as a strategy to escape a local-optimal solution [Mladenović and Hansen 1997]; [Glover and Kochenberger 2003].

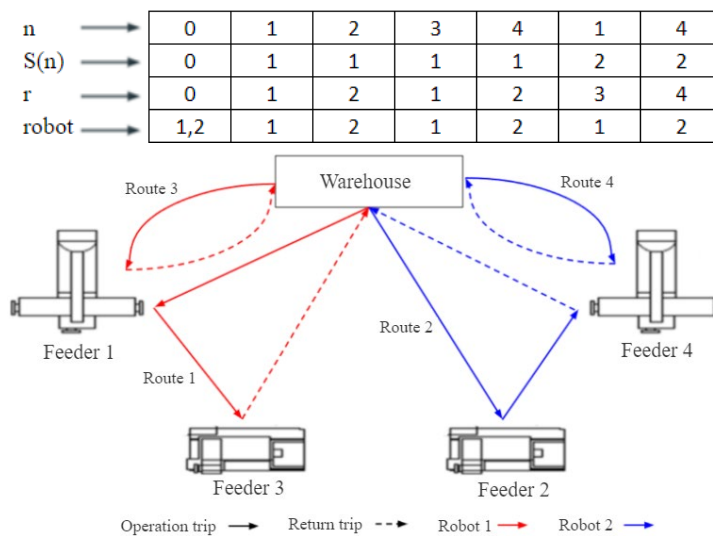
In [Hansen et al. 2019] authors present the basic schemes of VNS, some extensions, and describe recent developments concerning formulation space search and variable formulation search.

The VNS algorithm has been successfully used in several operational research [Mladenović and Hansen 1997]; [Hansen et al. 2019]; [Tomasson 2022], but at the course of this research, no applications were found to solve the specific SAMR problem being considered here.

3.1 Solution Coding

To facilitate the application of the proposed approach, a solution coding is used, as illustrated in Figure 2. In this coding, a matrix of order $(4 \times n)$ is considered. The first row represents the sequence of machines that the tasks are assigned to. The second row indicates the subtask of each respective task. The third row represents the route that needs to be performed. Finally, the last row indicates the chosen AMR. Here n is the total number of tasks, $S(n)$ is the total number of subtasks, r is the number of routes and lastly, the chosen robot.

Figure 2 - Data structure for representing a solution



Source: Authors 2023.

3.2 Heuristic Method for Initial Solution - HIS

A constructive heuristic, referred HIS, was developed to obtain an initial solution $S = S_0$ for the VNS-AMR algorithm, as depicted in line 1 of Figure 1(a). HIS performs the initial computations to obtain the release time information of each subtask k of task i (e_{ik}), as well as the deadline of subtask k of task i (d_{ik}), and, iteratively, selects the first subtask to be released for completion. If there are multiple subtasks that start at the same instant, the one with the shortest due date is chosen. The same is done for VND-AMR.

An example of an initial solution is illustrated in Figure 2. In this scenario, both robots are executing tasks in the warehouse. Robot 1 performs subtask 1 of task 1, on the same route, and then it proceeds to perform subtask 1 of task 3. After completing these tasks, it returns to the warehouse to replenish and then continues operations on route 3, where it performs subtask 2 of task 1. Likewise, Robot 2 does.

3.2.1 Solution Feasibility Analysis

Before evaluating the quality of the solution (Z at Equation (18)), it is necessary to verify whether the solution is feasible or not. This means checking if it adheres to all the constraints

(Equations (5)-(17)) of the model. If not, the solution is automatically discarded, and a new attempt is made to obtain a solution.

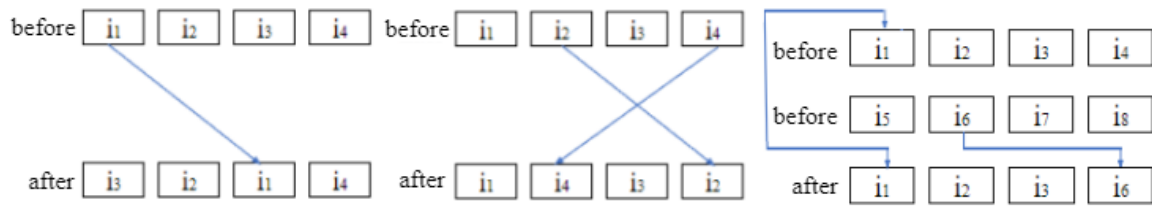
If the obtained solution is feasible, the quality of the solution is then assessed by considering the processing of all tasks and subtasks in the indicated task sequence. In other words, the time traveled between the warehouse and the feeders, the time traveled between feeders, the processing time of the tasks, the processing time in the warehouse, and the time in which the robot stands still waiting for the release time of the tasks are calculated.

3.2 Definition of Neighborhoods for Local Searches

For the application of descent searches in the neighborhoods of the working solution (line 6 Figure 2), three different types of neighborhood structures are considered, as listed below, based on [Zobolas et al. 2009]:

- **Internal Insertion:** in this structure, the change extracts a task at any position in the solution vector and reinserts it at a new position randomly (Figure 3a);
- **Internal Swap:** the change swaps the position of two different tasks randomly (Figure 3b);
- **External Insertion:** Using two solutions, the change extracts a position from each solution vector and inserts it into a new one, generating a new solution (Figure 3c).

Figure 3 - Internal Insertion, Internal Swap and External Insertion



Source: Authors 2023.

If the attempted swap/insertion (new neighbor solution) satisfies the constraints imposed by the problem, the swap/insertion is performed to obtain a new feasible solution. Otherwise, the neighbor is discarded, and a new neighbor is selected for evaluation.

3.5 Local Search

Denoting $N_1(s)$, $N_2(s)$ and $N_3(s)$ as the pre-defined neighborhood structures in Section 3.2.1, where s is the initial solution obtained by HIS, local search procedures are systematically applied. Starting from $N_1(s)$, apply some local search method and denote with s'' the obtained local optimum; if the solution thus obtained is better than s , move there ($s \leftarrow s''$), and continue the search with $N_1(s'')$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$. This summarizes VND-AMR (Figure 2b). If a random solution was generated s' before doing de local search method, we have the VNS-AMR (Figure 2a).

While in VNS we select a random neighbor of the current solution (line 6 Figures 1(a)) before conducting de local search. In VND, on the other hand, we directly analyze all neighbors of the current solution and perform the best movement (line 6 Figure 1(b)).

The feasibility of a solution during the local search also is determined by verifying if the constraints (2)-(17) are met.

4. Computational Experiments

Since our problem is a generalization of the SAMR from [Dang et al. 2014], the computational testing considers the same data used, which are shown in Table 1. Applying Equations (1), (2) and (3), the data allow us to calculate the periodicity, the release time of subtask k of task i , the deadline of subtask k of the task i and time windows are defined.

Table 2 illustrates the travel times between the source and destination pairs of the robots in the factory.

Table 1 - Part levels, Feed rate, and Work time data

Location	Warehouse	Feeder 1	Feeder 2	Feeder 3	Feeder 4
Maximum Part Level	-	250	2000	2000	250
Minimum Part Level	-	125	900	900	125
Feed Rate (seconds/piece)	-	4,5	1,5	1,5	4,5
Robot working time (s)	-	42	42	42	42

Source: Based on [Dang et al. 2014].

Table 2 - Robot travel time (seconds) between locations

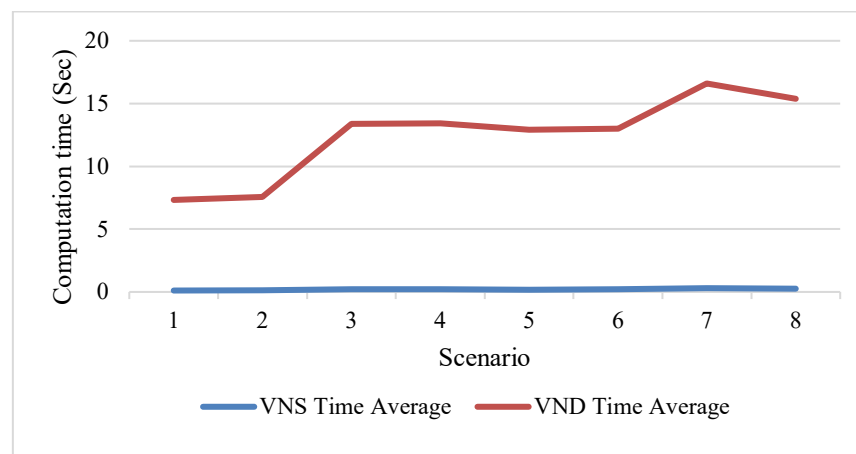
Location	Warehouse	Feeder 1	Feeder 2	Feeder 3	Feeder 4
Warehouse	0	34	37	34	40
Feeder 1	39	0	17	34	50
Feeder 2	35	17	0	35	49
Feeder 3	34	33	35	0	47
Feeder 4	36	47	48	46	0

Source: Based on [Dang et al. 2014].

For the computational experiments, different scenarios were considered, first separated by the number of feeders. Next, they were separated into two groups based on the maximum load capacity of the AMR. The first group had a maximum capacity of two SLCs, while the second group had a capacity of three SLCs. Finally, each scenario was divided into cases with varying numbers of tasks, as detailed in Table 3, which provides a summary of the results. The objective values and computation times for VNS are the averages of ten runs.

The algorithm was implemented in Python. All the experiments were run on PC having an Intel® Core(TM) i5-7400 3.00 GHz processor and 16 GB RAM. The planning horizon for study T is approximately 2,700 seconds due to the robot's battery capacity.

Figure 4 – Results by method



Source: Authors 2023.

There can be multiple sets of operations for the same number of total task subtasks. For example, scenario 8 can be represented by 3 subtasks to complete task 1, 2 subtasks to complete task 2, 2 subtasks to complete task 3, and 3 subtasks to complete task 4, resulting in a total of 10 subtasks. Alternatively, it can also be represented by 4 subtasks to complete task 1, 1 subtask to complete task 2, 1 subtask to complete task 3, and 4 subtasks to complete task 4, also resulting in a total of 10 subtasks. This last example was discussed here.

Table 3 – Summary of computational testing results obtained

Scenarios	N. of feeders	Q	Subtasks of tasks	MIP		VNS		VND	
				Obj.value (s)	Computation time (s)	Obj.value (s)	Computation time (s)	Obj.value (s)	Computation time (s)
1	2	2	4	604.5	0,1221	604.5	0,1125	604.50	7.3270
2	2	3	4	604.5	0,1425	604.5	0.1188	604.50	7.5756
3	3	2	6	-	-	619.7	0.1970	604.50	13.3845
4	3	3	6	-	-	610.4	0.2014	604.50	13.4342
5	4	2	8	-	-	1,178.28	0.1924	1.167.000	12.9351
6	4	3	8	-	-	1,174.37	0.2059	1,038.75	12.9930
7	4	2	10	-	-	1,729.50	0.3022	1,729.50	16.6002
8	4	3	10	-	-	1,729.50	0.2692	1,729.50	15.3653

Source: Authors 2023.

4.1. Analysis of the Results

Comparisons of the objective value and computation time for the MIP and proposed heuristics are presented in Table 3. It is possible to notice that the VNS-AMR and VND-AMR algorithms metaheuristics find feasible solutions for large problems, whereas when using MIP, it is not possible to find a solution to the problem due to its NP-Hard characteristic.

When the problem size increases, especially the number of tasks or subtasks, the solution time of the proposed metaheuristics becomes significantly longer. This is particularly true for VND-AMR, as shown in Figure 5.

VND's processing time is higher, which supports the fact that it utilizes a top-down search approach to find the best neighboring solution in each iteration, whereas VNS selects the first neighboring solution that improves the current solution.

However, it is worth noting that VND-AMR yields better objective function results.

It is imperative to acknowledge that the objective function presented in this study differs from the one examined in the study by [Dang et al. 2014]. Notably, the contrast extends to the consideration of a single robot in the former study, whereas the present investigation involves the contemplation of two robots. Considering these significant differences, any attempt to compare the outcomes obtained in this study and those documented in the previous work would lack meaningful validity.

5. Highlighting the Main Contributions of the Proposed Approach

In this work, two new objective functions are presented. The first objective function, described in Equation (18), is for homogeneous robots. The second objective function, described in Equation (19), is for heterogeneous robots. We also consider the travel time of the robots between machines, the working time, and the time it takes for them to return to the warehouse. In [Dang et al.

2014] authors only consider Equation (4), which calculates the total time the robot spends traveling between machines.

Another contribution is regarding the number of robots considered in our study. In [Dang et al. 2014] only one robot is considered, while in this study, two robots are considered. The determination of the optimal choice of robots assigned to task execution depends on the release time of the robots. Precisely, the initial task within the warehouse serves as a provisioning step for the robots, equipping them for subsequent operations. This initialization is executed simultaneously by both robots. Subsequent tasks are allocated to robot 1 initially, followed by an iterative selection process where the robot with the shortest operation time is designated to undertake the next task.

6. Conclusion

This paper presents the results of a study on scheduling mobile robots to perform part-feeding tasks on production lines in a flexible manufacturing environment. The study considers time window constraints for the feeding tasks and considers machine attributes, such as the feeding rate (parts per second), to minimize the total processing time using autonomous mobile robots.

It is worth noting that the SAMR problem is an NP-Hard problem. Therefore, the MIP solution approach considered here can only be applicable to small-scale problems with few feeders and a short planning horizon.

The survey conducted along the course of this study has revealed a significant gap in the current literature, particularly the scarcity of models that can accommodate multiple robots. This paper proposes two new objective functions that consider applications involving two or more robots. The first one considers robots that are equal, while the other one involves robots with distinctive characteristics. There are proposed two metaheuristic methods, VND-AMR and VNS-AMR, based on systematic exchanges of neighborhood structures.

For larger cases, solutions of the problem only can be obtained using approximate methods, such as the metaheuristic methods based on systematic exchange of neighborhood structures considered in this study. These methods can escape from local optima and can find high-quality feasible solutions for medium and large problems involving two robots.

The adaptability of the method to real-world scenarios is evident. It seamlessly aligns with practical use cases by allowing adjustments to the robot's capacities, travel durations to feeders and depots, feeder and depot processing times, and the count of feeders in the manufacturing domain.

To facilitate future extensions, we recommend expanding the approach to include stochastic or dynamic order arrivals and exploring methods for implementing the approach in different real-world settings.

References

- Dang, Q. V.; Steger-Jensen, I. N. K.; Madsen, O. (2014). Scheduling a single mobile robot for part-feeding tasks of production lines. *Journal Of Intelligent Manufacturing*, v.25, p. 1271-1287.
- Fernandes, H. A.; Taglialenha, S. L. S. (2021). Task sequencing and scheduling for autonomous mobile robots. In: *Anais do XLI ENEGEP*, v. 1, Foz do Iguaçu. ENEGEP.
- Glover, F.; Kochenberger, G.A. (2003) *Handbook of metaheuristics*. Kluwer Academic Publishers, New York.
- Hansen, P., Mladenović, N., Brimberg, J., Pérez, J. A. M. (2019). Variable Neighborhood Search. In: Gendreau, M., Potvin, JY. (eds) *Handbook of Metaheuristics*. International Series in *Operations Research & Management Science*, vol 272. Springer, Cham.
- Manafi, E.; Tavakkoli-Moghaddam, R.; Mahmoodjanloo, M. (2022). A centroid opposition-based coral reefs algorithm for solving an automated guided vehicle routing problem with a recharging constraint. *Applied Soft Computing*, v.128.

Mladenovic, N.; Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, Oxford, v. 24, n. 11, p. 1097-1100.

Pattnaik, S. K.; Mishra, D.; Panda, S. A. (2022). Comparative study of metaheuristics for local path planning of a mobile robot. *Engineering Optimization*, v. 54, n. 1, p. 134-152.

Riazi, S.; Lennartson, B. (2021). Using CP/SMT solvers for Scheduling and Routing of AGVs. *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 218-229.

Shen, W.; Wang, L.; Hao, Q. (2006). Agent-based distributed manufacturing process planning and scheduling: a state-of-the-art survey. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 36, no. 4, pp. 563-577.

Silver, E. A.; Pyke, D. F.; Peterson, R. (1998) *Inventory management and production planning and scheduling*. New York: Wiley.

Thoben, K. D.; Wiesner, S.; Wuest, T. (2017) Industry 4.0 and smart manufacturing- a review of research issues and application examples. *Int. J. Autom. Technol.* 11(1), 4–16.

Thomasson, O.; Battarra, M.; Erdoğan, G.; Laporte, G. (2022). Pallet location and job Scheduling in a Twin-Robot system. *Computers & Operations Research*, v. 147, ed. 105956, 12 p.

Ullrich, G. (2014). *Automated guided vehicle systems - a primal with practical applications*. Voerde, Germany: Springer, v. 1, p. 4-14.

Vieira, R. C.; Taglialenha, S.L.S. (2022). Heurística de vizinhança variável para programação de robôs móveis autônomos em ambiente de manufatura flexível. In: *Anais do XLII ENEGEP*, v. 1, Foz do Iguaçu. ENEGEP.

Wight, O. W. (1984). *Production and inventory management in the computer age*. New Jersey: John Wiley & Sons, Inc.

Zobolas, G. I.; Tarantilis, C. D.; Ioannou, G. (2009). Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. *Computers & Operations Research*, v. 36, p. 1249-1267.