International Journal of Industrial Engineering Computations 15 (2024) 667–684

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Ant colony algorithms for minimizing costs in multi-mode resource constrained project scheduling problems with spatial constraints

Miguel P. de la Pisa[a*], Jose C. Molina[a] and Ignacio Eguía[a]

[a]*Department of Industrial Management, University of Seville, Camino de los Descubrimientos, 41092, Sevilla, Spain*

C H R O N I C L E

Article history:
Received March 26 2023
Received in Revised Format April 28 2024
Accepted May 5 2024
Available online May 5 2024

Keywords:
Multi-mode resource constrained project scheduling
Ant colony system
Memetic algorithm
Spatial constraints
Aerospace

A B S T R A C T

This paper addresses the problem of activity scheduling and operator assignment in workstations of aerospace assembly lines. The problem is modelled as a new variant of the Multi-Mode Resource Constrained Project Scheduling Problem (MRCPSP), which incorporates practical features from aerospace workstations in assembly lines. These workstations have a substantial number of activities to be scheduled within a given assembly cycle time. It introduces particularities which are not usually addressed such as considering additional workers for performing activities, different workers' proficiency, and spatial limitations in work zones. The objective is to schedule the activities of an aerospace workstation, minimising the total labour cost, while satisfying the cycle time and the zone's limitations. The problem is initially formulated by employing mixed-integer linear programming methods with mathematical modelling and solved using two different algorithms: an Ant Colony System (ACS) and a memetic ACS. Given the novelty of the problem presented, new sets of benchmark cases of different sizes for this problem are also proposed and solved. To assess the performance of the algorithms, the solutions for the small-sized instances are compared in terms of deviation with the results obtained by an optimisation modelling software. Further experimentation with the algorithms is carried out with medium and large instances, showing good performance and providing reasonably good results in realistic problems.

1. Introduction

The aerospace industry has been at the vanguard of production methods for decades. Nevertheless, the need to remain competitive in terms of resource reduction, adapting to customers' requirements, and embracing the new digital paradigm of Industry 4.0 calls for further action to achieve even more efficient and flexible production systems. Scheduling plays a crucial role in optimising operations, but despite the high level of digitisation in the aerospace industry, scheduling processes have remained largely unaffected, with most activities being planned manually based on expert knowledge (Borreguero 2019). Aerospace Assembly Lines (AAL) scheduling problems are a variation of the well-known assembly line scheduling problem that considers the characteristics of the aerospace sector, such as low production rates, labour-intensive operations, and many tasks per product (Heike et al. 2001). These lines are mainly manual and paced. Since the failure to deliver on time may result in significant penalties for the manufacturer, it is crucial to meet schedules at each workstation of the AAL (Arkhipov et al., 2018). For the purposes of optimising the overall operations of an AAL, the first phase is to equitably distribute the tasks amongst the set of defined workstations, whilst also assigning the available workforce to each workstation. This problem is recognised as the assembly line worker assignment and balancing problem (ALWABP) originally defined by Miralles et al. (2007). The main objective of the ALWABP is the assignment of tasks to workstations such that the expected cycle time is minimised for the available workers (Ritt et al. 2016). This paper is focused on the following phase where tasks are assigned to workers at each workstation to find an optimal schedule of task processing that satisfies aerospace workstations constraints. The aerospace workstation scheduling problem contains features that are inherent to this particular manufacturing environment. First, workers may have different skills and proficiency levels which may result in different labour costs and execution times. Second, activities can be processed in several alternative modes, depending on the number of workers

* Corresponding author
E-mail migpasdel@alum.us.es (M. P. de la Pisa)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2024 Growing Science Ltd.
doi: 10.5267/j.ijiec.2024.5.002

assigned for the execution. As well as this, each mode for an activity is defined by a combination of operator skills and proficiency levels, number of operators and durations (Borreguero 2015). Finally, workstations are divided into smaller areas where a limited number of workers can operate simultaneously.

There has been little research on aerospace workstation scheduling, taking into consideration its distinctive characteristics. Borreguero et al. (2015) presented a mixed-integer linear programming (MILP) model to reduce labour costs. This model includes various execution modes for tasks, as well as constraints on zones and cycle time. Russell and Taghipour (2019) formulated a set of discrete-time multi-objective MILPs whose objectives include minimising makespan, incomplete activities, resource requirements, and deviations from cycle time and budget. The framework also considers activities with multiple resources and modes, with a maximum allowable capacity for work zones.

Based on the motivation behind this study, there are two main scientific contributions presented. The first is the formal definition of the aerospace workstation scheduling problem as a variant of the Multi-Mode Resource-Constrained Project Scheduling Problem (MRCPSP) which considers zone restrictions. The problem is referenced as MRCPSP-Z and is modelled using mixed-integer linear programming techniques. Realistic factors of AAL workstations, such as maximum cycle time, work zone limitations, and the number of available workers with varying proficiency levels are considered with the objective of minimizing the Total Labour Costs (TLC). The second contribution is the development of two different optimization approaches for addressing the MRCPSP-Z: (i) an Ant Colony System (ACS) algorithm, and (ii) a novel Memetic ACS (M-ACS) algorithm that combines ACS with a Variable Neighbourhood Descent (VND) algorithm. Both approaches incorporate a cost-peak-reduction mechanism aimed at enhancing the efficiency of generating improved solutions. To demonstrate the efficiency of ACS and M-ACS, tests are carried out on a newly created set of MRCPSP-Z instances.

This paper is organised as follows: Section 2 presents a summary of related research, including variants of the problem and corresponding methodologies. Section 3 describes the problem's characteristics, including a mathematical model formulation for the MRCPSP-Z. A description of the main components of the approaches is provided in Section 4. The parameter settings and experimental results obtained from problem instances are presented in Section 5. Finally, conclusions are provided in the final section.

2. Literature review

The scheduling of operations in an aerospace workstation can be viewed as a specific application of the Resource-Constrained Project Scheduling Problem (RCPSP), which aims to determine the start/finish time of a set of activities subject to some precedence relations and a certain number of available resources. In AALs, the main resources are workers, as workstations are characterised by labour-intensive operations. However, the RCPSP may not account for all situations occurring in practice. Three extensions are considered to adapt the RCPSP for the scheduling of activities in a workstation of AAL: (i) setting minimum and maximum number of workers to perform each activity, (ii) a set of workers with different skills and proficiency levels, and (iii) a maximum allowable number of workers in each workstation area. Moreover, minimizing overall labour costs constitutes a crucial factor in achieving efficient task scheduling within an AAL.

The standard RCPSP has only one way of performing each activity, whereas in the aerospace environment some operations could be performed in different ways depending on the number of assigned workers. This feature is contemplated in the MRCPSP, which arises when each activity can be processed in several alternative modes. As a result, the duration of each activity mode relies on the number of workers assigned to it. Dolgui et al. (2018) presented a MILP model with the objective of minimising the cycle time for scheduling on a paced assembly line, where the operation time varies depending on the number of workers assigned to the activity. Baradaran et al. (2012) investigated the MRCPSP within a Programme Evaluation and Review Technique (PERT) network. Their work introduced a list of execution modes outlining the various resource combinations required to complete an activity. The multi-skill RCPSP can be considered as a particularisation of the MRCPSP. In their respective studies, Povéda et al. (2023) and Li et al. (2024) approached this extension by defining the activity modes as all the possible combinations of resources possessing the required skills.

Expertise level has a direct impact on performance, as there is a progression from a newcomer to an expert worker. The second extension of the standard RCPSP accounts for the varying expertise of operators, which affects the model in diverse ways. The most prevalent application of this extension involves considering that worker level affects task duration and labour cost. Therefore, this second extension can also be modelled as multiple modes of executing operations in an MRCPSP. Cheng and Chu (2012) examined the processing duration in relation to workers' proficiency levels, defined in the range of [0-1], and the increase in the unitary labour cost based on the worker's experience. It is also possible to incorporate the proficiency level into the goal. Shahnazari et al. (2017) aimed to minimise the utilisation of less skilled workers as part of the objective function. Finally, Ahmadpour and Ghezavati (2019) have argued that the critical activities can only be assigned to workers with a certain expertise level.

The use of platforms and gigs is prevalent in aerospace workstations to enable easier access to large components. This consequently restricts the working area to a maximum number of personnel, which is considered a third extension of the standard model. The restriction of the maximum number of workers in a specific area is a frequent application in research (see e.g. Kadrou & Najid 2006), so this third extension is modelled as a new constraint.

The RCPSP and its variants are widely recognised as strongly NP-hard (Blazewicz et al., 1993). Consequently, numerous authors have developed heuristics and metaheuristics to achieve good solutions within reasonable running times. In their review, Van Peteghem and Vanhoucke (2014) provided an overview of existing metaheuristic solution procedures for MRCPSP, including Simulated Annealing, Genetic Algorithms (GA), Particle Swarm Optimisation (PSO) and other approaches. Pellerin et al. (2020) conducted a survey on hybrid approaches for RCPSP, which explored the combination of local search strategies with population-based metaheuristics and the sequential or parallel execution of potentially different pure metaheuristics that exchange information about the search process.

The most similar research to the proposed MRCPSP-Z is introduced in Kadrou and Najid (2006). They developed a new version of the parallel schedule generation scheme (PSGS) heuristic through the inclusion of priority rules used in a multi-skill MRCPSP with limited zones for minimizing the makespan. The technique is applied to a set of generated instances and the results are compared to other heuristics.

In previous studies, Ant Colony algorithms have effectively been used to solve MRCPSP. Shan et al. (2007) introduced a single pheromone approach for selecting activity-modes, alongside a global and local pheromone update strategy. Chiang and Huang (2012) investigated PSGS as a solution presentation method, and a dynamic tournament strategy was designed to find a balance between local optima and infeasible solutions during the exploration phase. Li and Zhang (2013) examined the serial schedule generation scheme (SSGS) to construct the scheduling using two independent pheromones for activity sequence and mode selection. Wuliang et al. (2014) proposed the use of a pool of priority rules as heuristic information and a branch and bound mechanism to discard infeasible paths before selecting the next activity during solution construction.

Ant Colony algorithms often use the optional local search phase extensively. The aim is to increase efficiency by improving the solutions constructed by ants using a local search algorithm. However, producing appropriate initial solutions for local search algorithms is a difficult task. Empirical evidence suggests that the probabilistic and adaptive solution generation process of ant colony algorithms is well-suited for this purpose (Dorigo et al., 2006). When a population-based global search approach is intensified with a local search procedure, the resultant method may be identified as a memetic algorithm (Moscato, 1989). Different approaches of memetic algorithms have been proposed to solve the MRCPSP in scientific research. Shen and Li (2013) suggested employing a PSO algorithm with a two-stage local search. The first step increases the level of resources to an activity to reduce the processing time. The second step is a swap strategy of adjacent activities. Khalilzadeh (2015) proposed a Honey Bee Swarm Optimization approach that incorporates a delay local search. This technique is performed by a mutation operator that delays scheduling each activity, regardless of its priority, to enable other activities to be scheduled earlier while preserving resources for other activities. Afshar et al. (2022) presented a GA with a local search component that applies mode permutation to activities on the critical path.

This paper presents the MRCPSP-Z. To the authors' best knowledge, no approach has been developed that specifically utilises ACS algorithms to address the MRCPSP, while considering zone limitations, minimum and maximum worker requirements for specific tasks, and variations in workers' skill levels that impact activity duration and costs.

3. Problem description and mathematical model

The Multi-Mode Resource Constrained Project Scheduling Problem with zone restrictions (MRCPSP-Z) is defined by a tuple $(\mathcal{V}; \mathcal{M}; \mathcal{p}; \mathcal{E}; \mathcal{R}; \mathcal{B}; \mathcal{b}; \mathcal{Z}; \mathcal{W}z, \mathcal{w}z)$ where:

- $\mathcal{V} = \{A_0, A_1, \ldots, A_n, A_{n+1}\}$ is a set of activities. Activities A_0 and A_{n+1} are dummy activities, representing, by convention, the start and the end of the schedule. The set of non-dummy activities is defined by $A = \{A_1, \ldots, A_n\}$.
- $\mathcal{M} \in \mathbb{N}^{n+2}$ is a vector of naturals, being M_j the number of modes that activity j can execute, with $M_0 = M_{n+1} = 1$, and $M_j \geq 1 \; \forall A_j \in A$.
- $\mathcal{p}$ is a vector of vectors of naturals, being P_{jm} the duration of activity j using mode m, with $1 \leq m \leq M_j$. For the dummy activities $P_{0,1} = P_{n+1,1} = 0$, and $P_{jm} > 0 \;\; \forall A_j \in A, \;\; 1 \leq m \leq M_j$.
- $\mathcal{E}$ is a set of pairs of activities representing precedence relations. Concretely, $(A_i, A_j) \in \mathcal{E}$ iff the execution of activity A_i must precede that of activity A_j, i.e., activity A_j must start after activity A_i has finished. Thus, all precedence relations between activities are considered finish-to-start with zero lag, A_0 is a predecessor of all other activities and A_{n+1} is a successor of all other activities.
- $\mathcal{R} = \{R_1, \ldots, R_{v-1}, R_v, R_{v+1}, \ldots, R_q\}$ is a set of resources. The first v resources are renewable, and the last q-v resources are non-renewable.
- $\mathcal{B} \in \mathbb{N}^q$ is a vector of naturals, being B_k the available amount of each resource R_k.
- $\mathcal{b}$ is a matrix of naturals corresponding to the resource demands of activities per mode. The value b_{jkm} represents the amount of resource R_k used during the execution of activity A_j in mode m. For the dummy activities $b_{0,k,1} = b_{n+1,k,1} = 0$ $\forall k \in \{1, \ldots, q\}$.
- $\mathcal{Z} = \{Z_1, \ldots, Z_m\}$ is a set of work zones.
- $\mathcal{WZ} \in \mathbb{N}^m$ is a vector of naturals, being WZ_z the capacity of each zone Z_z, i.e, maximum amount of renewable resources allowable at any time in zone Z_z.

- $\mathcal{WZ}$ is a vector of vectors of binaries, being wz_{jz} equal to 1 if activity j is processed in zone z. Only one zone per activity.

A schedule is a vector of naturals $S = (S_0, S_1, \ldots, S_n, S_{n+1})$ where S_j denotes the start time of activity A_j, and considering that $S_0 = 0$. A schedule of modes is a vector of naturals $\mathcal{SM} = (SM_0, SM_1, \ldots, SM_n, SM_{n+1})$ where SM_j, satisfying $1 \leq SM_j \leq M_j$, denotes the mode of each activity A_j. A solution to an MRCPSP instance is a schedule of modes $\mathcal{SM}$ and a schedule S, subject to the precedence relations, the resource constraints, and the zone limitations, and considering an objective function such as minimal makespan (S_{n+1}) or minimal number of resources used.

As mentioned before, the aerospace workstation scheduling problem is a particular case of the Multi-Mode Resource Constrained Project Scheduling Problem with zone restrictions (MRCPSP-Z). The following assumptions are considered in this problem that differs from the standard MRCPSP-Z:

- All resources are renewable from period to period and are associated with a set $\mathcal{K} = \{1, \ldots, k, \ldots, K\}$ of individual workers.
- A set of worker profiles $\mathcal{F} = \{1, \ldots, f, \ldots, F\}$ are defined. A worker profile is associated to one worker skill and one proficiency level in that skill.
- $\mathcal{D}$ is a vector of vectors of binaries, being D_{kf} equal to 1 if profile f is mastered by worker k, so that, each individual worker $k \in \mathcal{K}$ corresponds to only one profile $f \in \mathcal{F}$.
- Each activity requires a variable number of workers of certain profiles. The variable number of workers required for each activity $A_j \in$ A and the different profiles that could perform the activity A_j, generate several execution modes M_j for each activity A_j. Then, $\mathcal{WB}$ is a matrix of naturals corresponding to the resource demands of activities per mode. The value wb_{jmf} represents the number of workers with profile f used during the execution of activity A_j in mode m. For the dummy activities $wb_{0,1,f} = wb_{n+1,1,f} = 0 \; \forall f \in \mathcal{F}$.
- The cycle time (C) of the aerospace assembly line is a parameter of the aerospace workstation scheduling problem and is considered as a limitation in which the ending activity assigned to the workstation must be completed: $S_{n+1} \leq C$.
- The objective of the proposed problem is to determine the best assignment of the available workers to the activities in order to minimise the total labour costs per unit of time (TLC) in the workstation. Then, $\mathcal{WC} \in \mathbb{R}^K$ is a vector of reals, being WC_k the unitary cost of the individual worker $k \in$ K.
  While the main objective of RCPSP is to reduce the total project duration or makespan, AAL stations follow a fixed cycle time for completing all tasks assigned to each workstation in a synchronous line. It is worth noting that finishing a task in a single workstation prior to the stipulated cycle time is unproductive since the assigned workers will remain idle until the next cycle, despite still counting for costing calculations.

A new mixed-integer linear programming formulation is proposed for the MRCPSP-Z applied to the aerospace workstation based on the continuous-time formulation proposed by Correia et al. (2012) for the resource-constrained project scheduling problem with multi-skill resources (MSRCPSP). This model is extended to include the cycle completion time and the spatial limitations of the workstation in AALs. The spatial constraints make it necessary to include variables with a discrete-time formulation.

The new proposed model makes use of the following notation:

Sets:

$\mathcal{V} = \{A_0, A_1, \ldots, A_n, A_{n+1}\}$	Set of activities. Activities A_0 and A_{n+1} are the dummy activities
$\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$	Pairs of activities (i, j) such that A_i directly precedes A_j
$\mathcal{N} \subseteq \mathcal{V} \times \mathcal{V}$	Pairs of activities which have no precedence relations
$\mathcal{K} = \{1, \ldots, k, \ldots, K\}$	Set of individual workers
$\mathcal{F} = \{1, \ldots, f, \ldots, F\}$	Set of worker profiles
$\mathcal{Z} = \{1, \ldots, z, \ldots, Z\}$	Set of work zones
$\mathcal{T} = \{1, \ldots, h, \ldots, t, \ldots, C\}$	Set of time periods during the cycle time

Parameters:

M_j	amount of execution modes of activity A_j ($A_j \in \mathcal{V}$; $M_j \geq 1$); $M_0 = M_{n+1} = 1$
p_{jm}	processing time of activity A_j using mode m
wb_{jmf}	number of workers of profile f required by the mode m of activity A_j. The dummy activities have no profile requirements: $wb_{0,1,f} = wb_{n+1,1,f} = 0 \; \forall f \in \mathcal{F}$
C	cycle time of the assembly line

WZ_z capacity of zone z, i.e., maximum number of workers allowable at any time in zone z

D_{kf}=1 if profile f is mastered by worker k. Only one profile per worker

wz_{jz}=1 if activity A_j is processed in zone z. Only one zone per activity

WC_k unitary cost of worker k

UBS upper bound of the start time of any activity

UBW upper bound of the number of activities a worker can contribute

Variables:

S_j start time of activity A_j ($A_j \in \mathcal{V}$), that is, the schedule (S)

SM_{jm}=1 if activity A_j is executed in mode m ($A_j \in \mathcal{V}$, $1 \leq m \leq M_j$), that is, the schedule of modes ($\mathcal{SM}$)

U_{ij}=1 if activity A_i is completed before activity A_j starts ($A_i, A_j \in \mathcal{V} \wedge (i,j) \in \mathcal{N}$)

W_{jk}=1 if worker k contributes to activity A_j ($A_j \in \mathcal{V}$, $k \in \mathcal{K}$)

X_{jmt}=1 if activity A_j using mode m is completed exactly at time t ($A_j \in \mathcal{V}$, $1 \leq m \leq M_j$, $t \in \mathcal{T}$)

O_k=1 if worker k contributes at least to one activity ($k \in \mathcal{K}$)

TLC total labour cost of workers used in the AAL station per unit of time.

Model:

$$min \;\; TLC \tag{1}$$

$$s.t.,$$

$$S_j \geq S_i + \sum_{1 \leq m \leq \mathcal{M}_i} p_{im} \cdot SM_{im} \qquad \forall (i,j) \in \mathcal{E} \tag{2}$$

$$S_j \geq S_i + \sum_{1 \leq m \leq \mathcal{M}_i} p_{im} \cdot SM_{im} - UBS \cdot (1 - U_{ij}) \qquad \forall (i,j) \in \mathcal{N} \tag{3}$$

$$U_{ij} + U_{ji} \leq 1 \qquad \forall (i,j) \in \mathcal{N} \tag{4}$$

$$\sum_{k \in \mathcal{K}} W_{jk} \cdot D_{kf} = \sum_{1 \leq m \leq \mathcal{M}_j} wb_{jmf} \cdot SM_{jm} \qquad \forall A_j \in \mathcal{V} \neq \{A_0, A_{n+1}\};\ \forall f \in \mathcal{F} \tag{5}$$

$$W_{ik} + W_{jk} \leq U_{ij} + U_{ji} + 1 \qquad \forall k \in \mathcal{K};\ \forall (i,j) \in \mathcal{N} \tag{6}$$

$$\sum_{1 \leq m \leq \mathcal{M}_j} SM_{jm} = 1 \qquad \forall A_j \in \mathcal{V} \neq \{A_0, A_{n+1}\} \tag{7}$$

$$S_{n+1} \leq C \tag{8}$$

$$O_k \leq \sum_{j=1}^{N} W_{jk} \leq UBW \cdot O_k \qquad \forall k \in \mathcal{K} \tag{9}$$

$$TLC = \sum_{k \in \mathcal{K}} WC_k \cdot O_k \tag{10}$$

$$\sum_{t \in \mathcal{T}} X_{jmt} = SM_{jm} \qquad \forall A_j \in \mathcal{V} \tag{11}$$

$$(S_j + p_{jm}) - (1 - X_{jmt}) \cdot UBS \leq t \leq (S_j + p_{jm}) + (1 - X_{jmt}) \cdot UBS \quad \forall A_j \in \mathcal{V};\ 1 \leq m \leq \mathcal{M}_j; \forall t \in \mathcal{T} \tag{12}$$

$$\sum_{j\in\mathcal{N}}\sum_{1\le m\le\mathcal{M}_j}\sum_{f\in\mathcal{F}}\sum_{h=t}^{t+p_{jm}-1} wz_{jz}\cdot wb_{jmf}\cdot X_{jmh}\le WZ_z \qquad \forall z\in\mathcal{Z};\ \forall t\in\mathcal{T} \tag{13}$$

$$S_j \ge 0 \quad \forall A_j\in\mathcal{V}\neq\{A_0\};\ \ S_0=0 \tag{14}$$

$$X_{jmt}\in\{0,1\} \quad \forall A_j\in\mathcal{V};\ 1\le m\le\mathcal{M}_j;\ \forall t\in\mathcal{T} \tag{15}$$

$$U_{ij}\in\{0,1\} \quad \forall(i,j)\in\mathcal{N} \tag{16}$$

$$W_{jk}\in\{0,1\} \quad \forall A_j\in\mathcal{V}\neq\{A_0,A_{n+1}\};\forall k\in\mathcal{K} \tag{17}$$

$$SM_{jm}\in\{0,1\} \quad \forall A_j\in\mathcal{V};\ 1\le m\le\mathcal{M}_j \tag{18}$$

$$O_k\in\{0,1\} \quad \forall k\in\mathcal{K} \tag{19}$$

The objective function Eq. (1) aims to minimise the total labour cost of workers used in the AAL station per unit of time. Constraints Eq. (2) assure that the precedence relations hold for all pairs of activities $(i, j) \in \mathcal{E}$. Constraints Eq. (3) determine the values of variables U_{ij} for each pair of activities $(i, j) \in \mathcal{N}$ which have no precedence relations, where *UBS* denotes an upper bound of S_j (could be equal to the cycle time of the assembly line C). Constraints Eq. (4) complement constraints Eq. (3) to provide consistency for variables U_{ij}, that is, for each pair $(i, j) \in \mathcal{N}$ or A_i starts after A_j is completed (if U_{ij}=1 and U_{ji}=0) or A_j starts after A_i is completed (if U_{ij}=0 and U_{ji}=1) or both are processed simultaneously (if U_{ij}=U_{ji}=0). Constraints Eq. (5) ensure that the profile requirement of the activities in the mode to be performed are fulfilled through the assignment of the necessary workers. Constraints Eq. (6) limit the assignment of each worker to at most one activity at a time. Constraints Eq. (7) ensure that each activity will be performed in one of its modes. Constraint Eq. (8) assures that the last activity is scheduled before the cycle time assigned to the station. In constraints Eq. (9), variables O_k are computed from the values of variables W_{jk}, where *UBW* denotes an upper bound of the number of activities a worker can contribute (could be equal to the number of activities n to be performed in the workstation). In constraint Eq. (10), the total labour cost of workers per unit of time is computed. Constraints Eq. (11) to Eq. (13) are mandatory to include the spatial limitations in the workstation zones. Constraints Eq. (11) assure that if an activity A_j is performed using a mode m then the corresponding variable X_{jmt} is activated to complete the activity at time period t. Otherwise, X_{jmt} is not activated. Constraints Eq. (12) provide consistency for variables X and S, that is, the completion time t of activity A_j in mode m (if X_{jmt}=1) must be equal to the starting time of activity A_j (S_j) plus the processing time of j using mode m (p_{jm}). Otherwise (if X_{jmt}=0), then S_j is limited by an upper bound *UBS* (could be equal to C). Constraints Eq. (13) ensure that the maximum number of workers per zone is respected. Constraints Eq. (14) and Eq. (15) define the continuous-time and discrete-time variables respectively. Finally, constraints Eq. (16) to Eq. (19) define the binary auxiliary variables.

4. Solution Approach

Due to the NP-hard nature of MRCPSP-Z, the difficulty of finding an optimal solution increases exponentially with the problem size. Therefore, this section proposes an ACS and its hybridisation with local search (M-ACS) to optimise the proposed problem. This decision is based on the remarkable success of swarm intelligent algorithms, such as ACO and ACS, in MRCPSP problems (Li and Zhang 2013; Wuliang et al. 2014). In addition, these algorithms can easily incorporate local search procedures due to the inherent use of local heuristics (Dorigo & Stützle 2019). The ACS differs from ACO and other previous ant systems in three main aspects: first, the state transition rule provides a strategy for balancing the exploration of new paths with the knowledge gathered from previous solutions. Second, the global update rule is only applied to the best solution found in the iteration. Third, a local update rule is applied while the ants are building the solution. The aim of this new rule is to diversify the ants' solutions within the iteration by reducing the pheromone concentration in existing scheduling solutions. As a result, previous ants encourage current ants to choose other unexplored paths, making it less likely that several ants will produce identical solutions during an iteration (Dorigo et al., 2006). As ACS is an improvement on ACO, the expected results will therefore be more promising.

Since ACS has the advantages of robustness, parallel search characteristics and high solution efficiency (Xu et al. 2023), the search process can easily converge to a local minimum. Therefore, to escape from local optima, the ACS algorithm is combined with a local search phase performed by a Variable Neighbourhood Descent (VND) algorithm, which aims to improve the solution by intensifying the search. This hybridisation is referred to as M-ACS. In addition, to obtain a balance between intensification and diversification of the search space, the local search procedure is only applied to the ant's best solution during each iteration.

The remainder of this section introduces the main concepts of the proposed algorithm. Section 4.1 describes the ACS approach in detail and Section 4.2 describes the VND algorithm used in the local search procedure of the M-ACS.

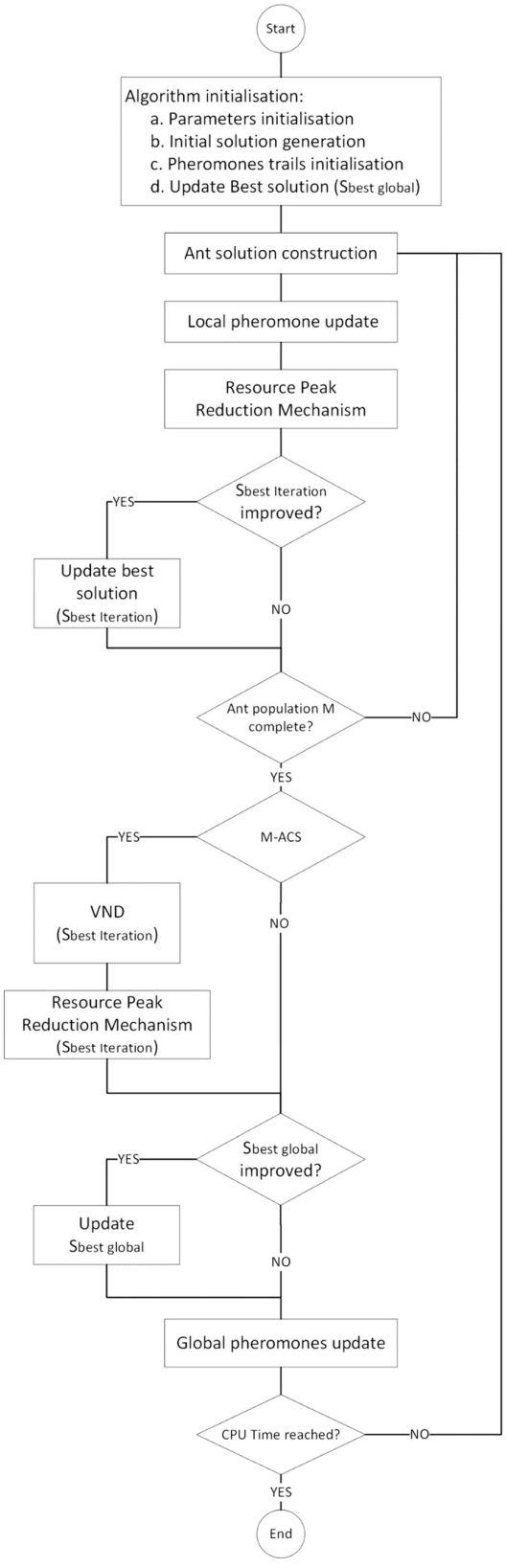

Fig. 1. Title: ACS algorithms workflow
Fig. 1. Description: Schematic depiction of ACS and its memetic variant

4.1. ACS mechanism for solving MRCPSP-Z

The Ant Colony System (ACS) algorithm is a probabilistic metaheuristic inspired by the behaviour of real ants, which find the shortest path when travelling from their nest to a food source. In the ACS algorithm, solutions are constructed in a probabilistic way, considering the attractiveness of the movement and the pheromone trails, which change during the execution of the algorithm. The two main phases of the algorithm are the ant route construction and the pheromone update, which are executed iteratively depending on two global parameters: the ant population (M) and the maximum number of iterations, which is limited by the maximum execution time (time_max). In addition, a minimum slack insertion heuristic is first introduced to start the pheromone trails. The algorithm flowchart of the proposed ACS is shown in Fig. 1.

4.1.1. Individual representation

For the MRCPSP-Z, a solution comprises not only the activity sequence but also the activity modes. Thus, to represent an ant solution, a pair of lists $I = (L^S, L^M)$ is introduced. L^S represents a precedence activity list, while L^M assigns the activity modes. In order to evaluate solutions, schedules are generated from the information obtained in L^S and L^M. Two distinct schedule generation schemes can be used in an RCPSP: serial and parallel. SSGS utilises activity incrementation in the stepwise procedure while PSGS utilizes time incrementation. This study adopts SSGS to evaluate solutions. The representation and scheme mentioned above have also been utilized by other authors in MRCPSP, including Zhang (2012), Li and Zhang (2013) and Wuliang et al. (2014). The SSGS therefore selects an activity from the ACS solution according to the sequence in the L^S, and resource and zone requirements are determined based on the mode selection L^M. Next, the start time of the activity is determined based on the earliest time at which the solution satisfies all MRCPSP-Z constraints (precedence relationship, zone restrictions and maximum number of workers per profile). The SSGS finishes its iteration once *J* activities are completed and then provides a makespan value, which is defined as the total time required to complete a group of activities. The makespan value is then compared to the cycle time to ultimately determine the solution feasibility.

4.1.2. Fitness function

The solution resulting from an evolutionary search must preferably be feasible, meaning that it satisfies all constraints. Nevertheless, the ACS algorithm may commence from infeasible solutions if the makespan value exceeds the total cycle time (*C*). To explore the infeasible solution space for searching better solutions, it is reasonable that infeasible solutions with smaller makespan values have better fitness values. Taking this consideration into account, a new fitness function, presented in Eq. (20), is proposed in this paper; where *PC* is a penalty cost coefficient defined as the total cost of using the entire available workforce, and finally *mak(S)* represents the makespan of the scheduled solution.

$$f(S) = TLC(S) + PC \cdot max\{0, mak(S) - C\} \tag{20}$$

4.1.3. Initial solution and pheromone trails initialisation

To initialise the pheromone trails, an initial solution is created using a simple insertion heuristic at the start of the ACS algorithm. A point of primary importance for the MRCPSP-Z is to produce solutions that satisfy the specified cycle time of the workstation. Consequently, the suggested heuristic solely considers the mode with a shorter activity time. In addition, the heuristic prioritizes the insertion of activities with minimum slack during each cycle. The slack of an activity is the difference between its latest start time (*LS*) and its earliest start time (*ES*). Note that the calculations are based on the group of unscheduled activities that satisfy the precedence relations, with durations associated to the mode with shorter time. Therefore, the algorithm selects the unscheduled activity with minimum slack and schedules it using a SSGS approach to satisfy all constraints. Most research on ACS algorithms, such as the studies by Reed et al. (2014) and Molina et al. (2020), employed Eq. (21) to determine the initial pheromone value. In this equation, *N* represents the total number of customers, and L represents the objective function of the solution obtained through the heuristic approach. Similarly, the initial pheromone value is computed by identifying *N* as the total number of activities.

$$\tau_0 = \frac{1}{N \cdot L} \tag{21}$$

```
1  A_s ← ActivityList() "Unassigned activities";
2  While (A_s ≠ 0) do:
3    For each activities a of A_s do:
4      mode← ShorterTimeDuration (a)
5      Slack_a,mode← Calculate_slack (a, mode, S),
6      (best_act, best_mode) ← StoreBest(Slack_a,mode)
7    EndFor
8    S← InsertActivity(best_act, best_mode), A_s ← RemoveActivity(best_act)
9  EndWhile
10 Return (S)
```

Fig. 2. Title: Algorithm 2: Minimum slack insertion heuristic

Fig. 2. Description: Illustration presenting the programming code of the Minimum slack insertion heuristic algorithm

4.1.4. Solution construction

Following the scheme considered by Li and Zhang (2013), two different decisions are made by an ant in solution searching; firstly, the selection of an activity j at the position i in the precedence activity list (L^S); secondly, the selection of the execution mode m for the activity j at the position i. To facilitate these decisions, two types of pheromones (τ_{ij} and τ_{ijm}), two types of attractiveness (η_{ij} and η_{ijm}) and two types of probabilities (p_{ij} and p_{ijm}) are respectively considered.

```
1  Initialize schedule (L^S, L^M);
2  Initialize non-yet-scheduled-activity list A_j, j=1,2,...,N;
3  For all positions i of L^S do:
4    J(i)←Find_Feasible_Insertion_Activities(A_j);
5    q←Select_Random_Number();
6    For all activities j of J(i) do:
7      mode←Min_Duration(j);
8      η_{i,j} ←Min_slack(L^S, L^M,mode);
9      p_{i,j} ←Calculate_Probabilities(η_{i,j}, τ_{(N,N)});
10   EndFor
11   If (q<= q_0) then:
12     t ←Maximum_Value(p_{i,j});
13   Else:
14     t ←Roulette_Wheel_Selection (p_{i,j});
15   EndIf
16   L^S← Insert(i, t); A_j ← Remove_Activity(t);
17   q_2←SelectRandomNumber();
18   For all modes m of t do:
19     η_{i,t,m} ←Min_Difference_Cost(L^S, L^M);
20     p_{i,t,m} ←Calculate_Probabilities( η_{i,t,m}, τ_{(N,N,MODES)});
21   EndFor
22   If (q_2<= q_0) then:
23     m←Maximum_Value(p_{i,t,m});
24   Else:
25     m ←Roulette_Wheel_Selection (p_{i,t,m});
26   EndIf
27   L^M ← Insert(i, mode);
28 EndFor
29 Return (L^S, L^M);
```

Fig. 3. Title: Algorithm 3: Ant Solution Construction Procedure
Fig. 3. Description: Illustration presenting the programming code of the Ant Solution Construction Procedure

In the ACS algorithm, ants travel from one activity to another activity to construct a solution. During each construction step, ants firstly select an unscheduled activity to be inserted into the next position of the L^S. This is done using a pseudo-random proportional rule. If a random number, q, which is uniformly distributed over [0.100], is less than q_0, the best activity is selected based on its pheromone and attractiveness. For this purpose, Eq. (22) is utilised where p_{ij} represents the probability of selecting an activity j in position i. Otherwise, the activity is selected by a fitness proportionate selection, also known as roulette wheel selection, according to the probability distribution provided in Eq. (22). Probability p_{ij} incorporates two separate components; the pheromone level during selection (i,j) denoted as τ_{ij}, and attractiveness (η_{ij}). $J(i)$ denotes the groups of activities with feasible insertions in position i.

$$p_{ij} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha} \cdot (\eta_{ij})^{\beta}}{\sum_{w \in J(i)} (\tau_{iw})^{\alpha} \cdot (\eta_{iw})^{\beta}}, & if\ j \in J(i) \\ 0, & otherwise \end{cases} \tag{22}$$

The attractiveness of the move in this decision is based on the priority given to the activity with the smaller total slack, following the same criterion used in the construction heuristic proposed in Section 4.1.3. The slack is calculated for every feasible insertion associating activity durations to the mode with a shorter time and considering the selected modes for the scheduled activities i. This measure was also utilised by Zhang (2012) and Li and Zhang (2013) in their corresponding studies and can be computed with Eq. (23).

$$\eta_{ij} = \max_{h \in J(i)} (LS_h - ES_h) - (LS_j - ES_j) + 1 \tag{23}$$

Once an activity is inserted into the L^S, the second decision is to select its execution mode m. The process is like the one explained for the first step but considering a different type of pheromone (τ_{ijm}) and a new definition of attractiveness (η_{ijm}). Consequently, Eq. (24) is utilized to compute the probability to select a mode m.

$$p_{ijm} = \begin{cases} \dfrac{(\tau_{ijm})^{\alpha} \cdot (\eta_{ijm})^{\beta}}{\sum_{w \in M(j)} (\tau_{ijw})^{\alpha} \cdot (\eta_{ijw})^{\beta}}, & \text{if m} \in \text{M(j)} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

Attractiveness η_{ijm} is defined with the aim of decreasing the overall solution cost and makespan. Therefore, a mode is considered less attractive if it requires additional workers to be introduced into a partial solution, as it penalises the objective function. Among modes that do not introduce further workers in the solution, the one with the shortest processing time is preferred. In this case, we define attractiveness η_{ijm} in Eq. (25), where Δf_{ijm} is the difference in the objective function value of the partial solution after introducing activity j with mode m at position i, and P_{jm} is the processing time of activity j in mode m.

$$\eta_{ijm} = \frac{1}{(1 + \Delta f_{ijm}) \cdot P_{jm}} \tag{25}$$

4.1.5. Pheromone update trail

The ACS method uses two types of pheromone updates: local and global. Each time a solution is created, the local update is performed by modifying the pheromone level of the selection (i, j) for τ_{ij} and (i, j, m) for τ_{ijm} of the obtained solution (S) as shown respectively in Eq. (26a) and (26b), where the parameter ρ is introduced to regulate the reduction of pheromone on the arcs. Otherwise, at the end of each iteration, when all ants have constructed their solution, the global update is only performed by the ant that produced the best solution found so far (S^*). The pheromone trail of the selection (i, j) and (i, j, m) are respectively updated as shown in Eq. (27a) and (27b), where $\Delta\tau^{bs} = (N \cdot L_{best})^{-1}$.

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \qquad \text{if } (i,j) \in S \tag{26a}$$

$$\tau_{ijm} = (1 - \rho) \cdot \tau_{ijm} + \rho \cdot \tau_0 \qquad \text{if } (i,j,m) \in S \tag{26b}$$

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{bs} \qquad \forall (i,j) \in S^* \tag{27a}$$

$$\tau_{ijm} = (1 - \rho) \cdot \tau_{ijm} + \rho \cdot \Delta\tau_{ijm}^{bs} \qquad \forall (i,j,m) \in S^* \tag{27b}$$

4.1.6. Resource Peak Reduction Mechanism (RPRM)

The SSGS schedules activities once the constraints of workers and zones allow it. This strategy is incompatible with minimising the total labour cost as it could cause a peak in the number of workers during a certain period. The MRCPSP-Z does not impose a penalty on the objective function if an activity is scheduled for a later time if the total cycle time restriction is met. Thus, activity start times must be adjusted to achieve new solutions with a lower total labour cost. Therefore, the RPRM is utilised to assess the potential decrease in the maximum number of assigned employees of each profile via adjustments to the starting times of activities. For every solution found by the ants, the RPRM is executed. It progressively decreases the number of workers used per profile in a solution by one and then attempts to reschedule the activities using SSGS to obtain a new solution. The pseudo-code of the RPRM is shown in Fig. 4.

```
1  S← S';
2  For all profile o  do:
3    S'← Reduce_used_resource(o);
4    S' ← SGSS_scheme();
5      If f (S') improves f (S) AND S' is feasible then:
6      S ← S'
7      EndIf
8  EndFor
9  Return (S)
```

Fig. 4. Title: Algorithm 4: Resource Peak Reduction mechanism
Fig. 4. Description: Illustration presenting the programming code of the Resource Peak Reduction mechanism.

Fig. 5 illustrates the application of the RPRM. Activities A and B have no precedence relationship and are scheduled concurrently using the SSGS as the solution meets the maximum number of available workers of a given profile (Fig. 5a). Therefore, six workers are required to perform these activities. After implementing the RPRM, the maximum number of

workers used in the solution is reduced by one. When implementing the SSGS to obtain a new schedule, the start time of activity B is delayed satisfying resource constraints. As a result, subsequent activities in the solution are affected. This example shows that reducing the number of available workers affects the calculation of the new activity start times. Nevertheless, the solution maintains the same makespan value. Note that both schedules (Fig. 5a and 5b) share a common solution representation and therefore, the second schedule would never be achieved by only applying a SSGS.

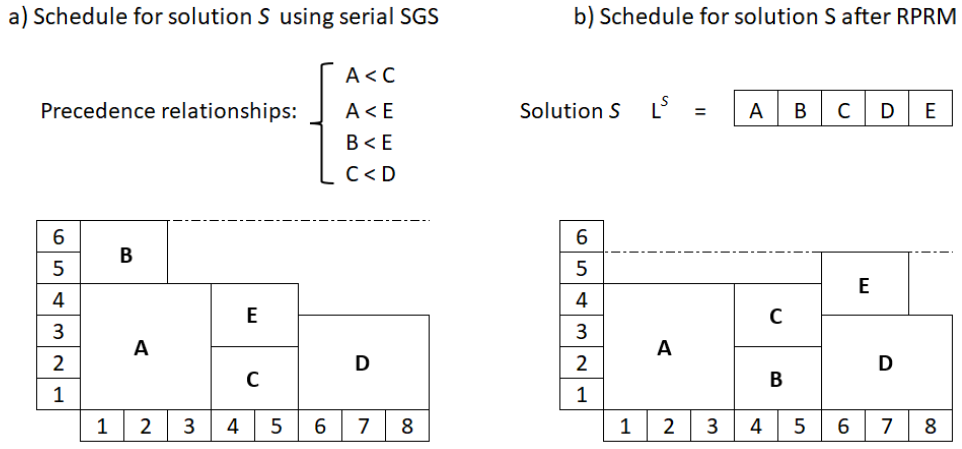

Fig. 5. Title: Example of RPRM application
Fig. 5. Description: Illustration showing an example of how the Resource Peak Reduction mechanism reduces the maximum number of workers required in the solution by one unit.

4.2. Local search procedure for M-ACS

The Variable Neighbourhood Search (VNS) is a metaheuristic originally proposed by Mladenović and Hansen (1997) that includes three phases: shaking, local search and move. Since a local optimum for a given type of move (neighbourhood structure) is not necessarily the same for another, the basic idea of VNS is to change the neighbourhood structure during the search to escape from local optima.

This paper focuses on its simplest version, Variable Neighbourhood Descent (VND), which arises when the search is performed in a deterministic manner (Duarte et al. 2018). The VND algorithm functions as the local search phase in M-ACS. and it is only applied to the ant's best solution during each iteration.

```
1  Define a set of neighborhood structures N_λ, λ=1, 2,...,λ_max;
2  S← S';
3  λ=1 ;
4  While (λ ≤ λ_max) do:
5     S' ← Best_Improvement_Search (S, λ);
6     If f(S') improves f(S) then:
7        S ← S'; λ ← 1;
8     Else:
9           λ= λ+1;
10    EndIf
11 EndWhile
12 If f(S') improves f(S) then:
13    S ← S';
14 EndIf
15 Return (S)
```

Fig. 6. Title: Algorithm 5: VND Procedure.
Fig. 6. Description: Illustration presenting the programming code of the Variable Neighbourhood Descent Procedure

Specifically, the VND starts by defining a set of neighbourhood structures N_λ ($\lambda=1,\ldots,\lambda_{max}$). The VND then starts from an initial solution s and a local search based on the best improvement search is performed to determine a new

solution *s'* in the neighbourhood N_λ. The neighbourhood of *s* in N_λ is defined by all the solutions it can be transformed into by performing a predefined move. If *s'* improves the objective function of the current solution *s,* then *s* is replaced by s' and the search returns to N_1; otherwise, the search continues with the next neighbourhood structure $N_{\lambda+1}$ until all neighbourhood structures have been examined ($\lambda=\lambda_{max}$).

The VND scheme implemented in this paper varies between two neighbourhood structures ($\lambda_{max}=2$) which are defined by two different operators. Firstly, N_1 is defined by a mode mutation operator, which aims to generate a solution modifying the assigned activity mode. Secondly, N_2 is defined by a relocate operator, which removes an activity from a position in the solution and inserts it into another feasible position according to its precedence relationships. All activity modes are also explored in the insertion. The pseudo-code of the VND algorithm is presented in Fig. 6.

5. Experimental approach

This section outlines the computational experiments undertaken to validate the effectiveness of the two algorithms developed, ACS and memetic ACS. The algorithms have been coded in C++ and executed on a 2.60 GHz Intel®Core (TM) i7-9750H CPU with 16 GB of RAM. Firstly, Section 5.1 sets out the process for creating a new set of benchmark problem instances for the MRCPSP-Z. Subsequently, Section 5.2 explains how the algorithm was tested to determine the optimal parameter values. Finally, this section presents a comparison of computational results between the proposed algorithms for MRCPSP-Z and MRCPSP.

5.1. Assembly line workstation instances generation

The Project Scheduling Problem Library (PSPLIB), as proposed by Kolisch and Sprecher in 1997, and MMLIB (Van Peteghem & Vanhoucke, 2014), comprise a series of test cases that allow the evaluation of the efficiency of new methods and techniques for solving the RCPSP or MRCPSP. However, this library is inadequate for the MRCPSP-Z due to its omission of several crucial attributes, including multi-mode based on the allocation of multiple workers per activity and worker proficiency levels, labour costs, and spatial constraints.

In order to generate relevant instances that are fully equivalent to real industrial cases, the main characteristics of the AAL workstation are taken into account when designing the sets to be used in the experimentation: (1) the number of precedence constraints is low, most of the tasks have only one direct predecessor; (2) most of the precedence constraints are general precedence constraints; (3) most of the tasks can only be performed by operators with a single skill, regardless of the skill level; (4) the number of operators that can perform an activity is limited, usually not exceeding three; and (5) there are usually not more than four operator profiles per platform (Borreguero et al., 2015).

Initially, RanGen2 network generator (Vanhoucke et al. 2008) is used to produce the RCPSP instances. A critical parameter in RanGen2 is the serial-parallel indicator (I2), which is set to 0.8 to create networks with a reduced number of precedence constraints (features 1 and 2). Secondly, the RanGen2 tasks generated utilise multiple resources, thus necessitating post-processing to adapt them to single-skill tasks (feature 3) and scale the required workforce within the range of [0-4] (feature 4). Each task is randomly assigned to a specific zone, and the maximum number of workers per zone varies between [4, 5] (feature 5). The cycle time ranges from 80% to 100% of the total time for the generated RanGen2 tasks.

The resources available in RanGen2 must be distributed among predetermined profiles while adhering to the initial availability (as shown in Fig. 7a). These profiles are determined by combining the numbers of resources with the proficiency levels of workers. Each profile incurs a unique cost, which increases with resource expertise level (as detailed in Fig. 7b).

The list of activity modes is generated by considering all possible combinations of available workers for each required profile. The requested number of workers may vary, resulting in an increase in possible combinations and consequently, a larger number of modes per task. Fig. 7c depicts a scenario in which an activity can be executed in four distinct modes, with workers of four different profiles. Each mode has varying processing times (dur) based on the number of workers and their profile. The higher the proficiency and number of allocated workers, the shorter the processing time.

a) Resource availability per profile in benchmark instance

```
**************************
RESOURCE AVAILABILITIES:
P1      P2      P3      P4
2       2       3       1
**************************
```

b) Cost per profile in benchmark instance

```
**************************
UNITARY COSTS:
P1      P2      P3      P4
35      28      25      20
**************************
```

c) Mode activity list in benchmark instance

```
************************************************
REQUESTS/DURATIONS:
jobnr.  mode    dur     P1      P2      P3      P4
------------------------------------------------
4       1       4       0       0       2       0
        2       5       0       0       1       1
        3       2       0       0       3       0
        4       3       0       0       2       1
```

Fig. 7. Title: Sections of a benchmark instance example.
Fig. 7. Description: Illustration of three sections that belong to a benchmark instance data file.

To obtain benchmark instances for the MRCPSP-Z, three sets of 26, 60, and 90 tasks with 24, 16 and 16 instances were generated. Each set ranges from 2 to 4 worker skills with 2 to 3 levels of proficiency resulting in a range of 4 to 16 different worker profiles.

5.2. Parameters settings

Selecting parameters is crucial for achieving good solutions. This section focuses on identifying the optimal parameters for the recommended ACS and M-ACS. In ant colony algorithms, the Taguchi method (Taguchi et al., 2005) has demonstrated its efficacy in parameter selection (Vinay & Sridharan, 2013). A range of representative problems of varying sizes were solved using different parameter combinations for the selection process, with the best possible configuration chosen as the one yielding the most optimal outcomes. By adopting this method, the entire combination of experiments is determined by a range of parameters. Five levels of the six parameters are considered, as represented in Table 1: the pheromone factor α, the heuristic information factor β, the evaporation rate ρ, the random variable selection parameter q_0, the population size M as a percentage of the number of tasks, and the maximum execution time (*time_max*). The final value is expressed in seconds and is computed using Eq. 28, considering several factors such as the number of tasks, resources, and levels. The last parameter to be incorporated in the selection is a proportional time factor (*TF*).

$$time\_max = TF \cdot (N.Task)^2 \cdot N.Resources \cdot N.Proficiencty\ levels \tag{28}$$

Table 1
Parameters levels

Level \ Parameter	α	β	ρ	q_0	M	TF
Level 1	1	1	0,1	70	10	0,01
Level 2	1,5	1,5	0,2	75	15	0,02
Level 3	2	2	0,3	80	25	0,03
Level 4	2,5	2,5	0,4	85	50	0,04
Level 5	3	3	0,5	90	75	0,05

To ensure parameters are effectively set, this study did not consider all parameter combinations (6^5), as it would result in significant computing time. The first characteristic of the Taguchi method is to use orthogonal arrays to reduce the number of experiments while ensuring representativeness. In this study, an L25 orthogonal array was employed, consisting of 25 experiments. One instance of each generated set was selected and solved five times, resulting in a total of 375 experiments.

The second feature of the Taguchi method centres on assessing the impact of parameters on the ultimate performance. To achieve this goal, the signal-to-noise ratio (SNR) is employed (Eq. 29), with i denoting the experiment number, u standing for the instance number, N representing the total number of instances in the experiment and y indicating the response of the algorithm execution proposed for a specific replication. A higher SNR value will result in reduced noise affecting the evaluated system, thereby improving its performance.

$$\mathrm{SNR}_i = -10 \log\left(\sum_{u=1}^{N} \frac{y_u^2}{N}\right) \tag{29}$$

Due to variations in the scale of the objective function values across selected instances and the stochastic nature of the metaheuristics, Eq. (30) was used to normalise the values of y_u. Here, $f(u, j)$ represents the objective function value of an instance u during the independent run j, whereas $best_{f(u)}$ indicates the minimum objective function value obtained for instance u across all calibration experiments.

$$\mathrm{y_u} = \frac{1}{5} \cdot \sum_{\mathrm{j=1}}^{5} \frac{f(u,j) - best_{f(u)}}{best_{f(u)}} \tag{30}$$

Table 2
Experiments parameter combination and SNR results.

Exp#	α	β	ρ	q_0	M	TF	SNR	Exp#	α	β	ρ	q_0	M	TF	SNR
1	1	1	0,1	70	10	0,01	19,32	16	2,5	1	0,4	75	75	0,03	28,84
2	1	1,5	0,2	75	15	0,02	28,54	17	2,5	1,5	0,5	80	10	0,04	33,11
3	1	2	0,3	80	25	0,03	34,72	18	2,5	2	0,1	85	15	0,05	39,42
4	1	2,5	0,4	85	50	0,04	51,09	19	2,5	2,5	0,2	90	25	0,01	47,6
5	1	3	0,5	90	75	0,05	72,68	20	2,5	3	0,3	70	50	0,02	28,66
6	1,5	1	0,2	80	50	0,05	34,11	21	3	1	0,5	85	25	0,02	38,8
7	1,5	1,5	0,3	85	75	0,01	39,38	22	3	1,5	0,1	90	50	0,03	50,1
8	1,5	2	0,4	90	10	0,02	59,64	23	3	2	0,2	70	75	0,04	25,61
9	1,5	2,5	0,5	70	15	0,03	29,14	24	3	2,5	0,3	75	10	0,05	28,39
10	1,5	3	0,1	75	25	0,04	33,56	25	3	3	0,4	80	15	0,01	32,52
11	2	1	0,3	90	15	0,04	40,91								
12	2	1,5	0,4	70	25	0,05	26,08								
13	2	2	0,5	75	50	0,01	31,96								
14	2	2,5	0,1	80	75	0,02	34,59								
15	2	3	0,2	85	10	0,03	44,83								

The signal-to-noise ratio (SNR) results for every experiment are displayed in Table 2. The average SNR for each level-parameter combination, calculated from five experiments while considering the level of the parameter, is presented in Table 3. The parameter with the highest SNR is associated with the optimal level. α and β are assigned values of 1 and 3, respectively, to give greater significance to heuristic information in arc selection. The value of ρ is set at the highest level of 0,5, leading to a maximum reduction in pheromone on the arcs. Additionally, q_0 is set to the maximum level of 90, resulting in a decrease in diversification during the ant search phase. Finally, the quantity of ants in the experiment is defined by two parameters: *M* set to 75 and *time_factor* set to 0.05.

Table 3
SNR per level-parameter and selection.

Level \ Parameter	α	β	ρ	q_0	M	TF
1	41,270	32,396	35,396	25,760	37,056	34,156
2	39,164	35,440	36,138	30,258	34,107	38,045
3	35,673	38,270	34,411	33,809	36,148	37,525
4	35,526	38,160	39,635	42,705	39,184	36,855
5	35,082	42,450	41,135	54,183	40,221	40,135
max SNR	41,270	42,450	41,135	54,183	40,221	40,135
Value	**1**	**3**	**0,5**	**90**	**75**	**0,05**

5.3. Computational results

This section details the comparison of the proposed methods to exact method solutions for small-sized instances of the MRCPSP-Z. Additionally, it compares the solutions found by the proposed methods for medium and large-sized instances of the MRCPSP-Z.

5.3.1. Comparison with exact methods in small size instances

This section presents the computational results of ACS and M-ACS obtained from the MRCPSP-Z small size instances. The algorithms' performance is assessed through a comparison with the solutions acquired from resolving the MILP model suggested in Section 3, restricted to a CPU time limit of 5 hours. In this regard, the Optimisation Modelling Software for Linear Programming LINGO was utilised to resolve the MILP model. Twenty-four scenarios with a range of 26 tasks were created, involving 2 to 4 resources and requiring 2 to 3 levels of proficiency. For each scenario, algorithmic processes were performed five times. The best solutions were

identified, and the relative deviations are displayed in Table 4. Results were categorised according to whether they were equal to (=), improved (<), or worsened (>) the solutions obtained by LINGO. An asterisk (*) signifies that LINGO was stopped due to a CPU time limit of 5 hours.

Both ACS and M-ACS provide effective solutions, matching 91.67% of the solver solutions, 22 out of 24 in total, and finding better solutions in 2 out of 22 cases. The average total relative deviation is -0.18%. In the worst-case scenario, where the solver solution is not found by the algorithms, the relative deviation is 2%. In contrast, in the best-case scenario where the algorithms improve solver solutions, the relative deviation is -5.43%. The results suggest that the algorithms acquire optimal or almost-optimal solutions.

Table 4
Results for Set26 instances

Instance name	Number of tasks	Number of resources	Number of proficiency levels	Number of profiles	Maximum zones occupation	Solver Solutions		ACS			M-ACS		
						TLC	Time elapsed (seconds)	TLC	Time elapsed (seconds)	Relative deviation	TLC	Time elapsed (seconds)	Relative deviation
Set_26.R2.L2	26	2	2	4	4	133	23	133 (=)	135	0,00%	133 (=)	135	0,00%
Set_26.R2.L2	26	2	2	4	4	171	22	171 (=)	135	0,00%	171 (=)	135	0,00%
Set_26.R2.L2	26	2	2	4	5	154	31	154 (=)	135	0,00%	154 (=)	135	0,00%
Set_26.R2.L2	26	2	2	4	5	144	22	144 (=)	135	0,00%	144 (=)	135	0,00%
Set_26.R2.L3	26	2	3	6	4	147	18000*	147 (=)	203	0,00%	147 (=)	203	0,00%
Set_26.R2.L3	26	2	3	6	4	147	18000*	122 (<)	203	-5,43%	122 (<)	203	-5,43%
Set_26.R2.L3	26	2	3	6	5	152	81	152,5	203	0,33%	152,5	203	0,33%
Set_26.R2.L3	26	2	3	6	5	145	16	145 (=)	203	0,00%	145 (=)	203	0,00%
Set_26.R3.L2	26	3	2	6	4	208	715	208 (=)	203	0,00%	208 (=)	203	0,00%
Set_26.R3.L2	26	3	2	6	4	283	122	283 (=)	203	0,00%	283 (=)	203	0,00%
Set_26.R3.L2	26	3	2	6	5	269	42	269 (=)	203	0,00%	269 (=)	203	0,00%
Set_26.R3.L2	26	3	2	6	5	237	389	237 (=)	203	0,00%	237 (=)	203	0,00%
Set_26.R3.L3	26	3	3	9	4	200	51	204 (>)	304	2,00%	204 (>)	304	2,00%
Set_26.R3.L3	26	3	3	9	4	262	1438	262 (=)	304	0,00%	262 (=)	304	0,00%
Set_26.R3.L3	26	3	3	9	5	255	28	255 (=)	304	0,00%	255 (=)	304	0,00%
Set_26.R3.L3	26	3	3	9	5	179	18000*	179 (=)	304	0,00%	179 (=)	304	0,00%
Set_26.R4.L2	26	4	2	8	4	281	26	281 (=)	270	0,00%	281 (=)	270	0,00%
Set_26.R4.L2	26	4	2	8	4	258	18000*	258 (=)	270	0,00%	258 (=)	270	0,00%
Set_26.R4.L2	26	4	2	8	5	283	183	283 (=)	270	0,00%	283 (=)	270	0,00%
Set_26.R4.L2	26	4	2	8	5	267	22	267 (=)	270	0,00%	267 (=)	270	0,00%
Set_26.R4.L3	26	4	3	12	4	170	37	170 (=)	406	0,00%	170 (=)	406	0,00%
Set_26.R4.L3	26	4	3	12	4	249,5	18000*	245 (<)	406	-1,21%	245 (<)	406	-1,21%
Set_26.R4.L3	26	4	3	12	5	246	1190	246 (=)	406	0,00%	246 (=)	406	0,00%
Set_26.R4.L3	26	4	3	12	5	179	31	179 (=)	406	0,00%	179 (=)	406	0,00%
Average values						209,15	3936	208,1 (<)	6084	-0,18%	208,1 (<)	6084	-0,18%

* Solver was stopped at CPU time limit (5 hours)

5.3.2. Medium and large instance experimentation

Further testing with larger instances is necessary to validate the algorithm's performance in real industrial scenarios, as the results obtained in small instance sets were similar. The generated sets comprised 60 and 90 tasks, with proficiency levels ranging from 2 to 3 and resources ranging from 3 to 4. The algorithm was executed five times for each instance, and Table 5 displays the average and best solution discovered. Using the same system employed in the previous section to classify M-ACS as equal, improved or worsened when compared to ACS solutions, the results reveal a slightly better performance of M-ACS. The total average relative deviation is -0.23%, with ACS solutions being improved in 7 cases, presenting a maximum relative deviation of -1.8%. On the other hand, worsened performance was observed in only one case, with a relative deviation of 0.6%. In the remaining 24 instances, the solutions were found to be equal. The level of repetitiveness observed in the best solution found during the five trials is useful for assessing the robustness of the algorithms. ACS reveals an average repetitiveness of 85% in the best solution found, whereas M-ACS indicates 83.1%. This feature can be assessed by comparing the average TLC of both algorithms. M-ACS outperforms ACS in 3 cases but worsens in 1 case. Therefore, both algorithms exhibit comparable performance, with M-ACS demonstrating slightly better results.

The inclusion of the local search requires analysis. Both algorithms were run with identical CPU time per instance. M-ACS used an average of 53.14% of the time on the local search procedure (intensification phase) rather than on ant evolution (search phase). In Set60 instances, the VND application yielded a success rate of 23.63% in identifying intermediate best solutions during a trial. Similarly, in Set90 instances, this success rate amounted to 22.49%. Out of all executed instances, the VND application only managed to find the best solution in 33.75% of them. Moreover, the VND did not produce any alternative solutions when compared to the ACS results. Selecting TLC as the objective function results in a small and limited set of solutions with different objective function values for the MRCPSP-Z. Improving solutions during the intensification phase of the M-ACS is difficult because VND can only enhance solutions by identifying a neighbour that utilises less costly worker profiles in a particular activity, while ensuring the resource requirements of the remaining activities. This observation implies that the exploration phase is more efficient than the intensification phase for this problem.

Table 5
Results for Set60 and Set90 instances

Instance name	Number of tasks	Number of resources	Number of proficiency levels	Number of profiles	Total modes	Maximum zones occupation	Time elapsed (seconds)	ACS		M-ACS			
								Min TLC	Average TLC	Min TLC	Average TLC	Rel. dev. from Min TLC ACS	Time used in LS (%)
Set_60.R3.L2_1	60	3	2	6	211	4	1080	203	203	203 (=)	203 (=)	0,0%	43%
Set_60.R3.L2_2	60	3	2	6	233	4	1080	277	277	272 (<)	272 (<)	-1,8%	75%
Set_60.R3.L2_3	60	3	2	6	182	5	1080	235	235	235 (=)	235 (=)	0,0%	39%
Set_60.R3.L2_4	60	3	2	6	182	5	1080	234	234	234 (=)	234 (=)	0,0%	43%
Set_60.R3.L3_1	60	3	3	9	333	4	1620	269,5	270,7	266,5 (<)	268,6 (<)	-1,1%	59%
Set_60.R3.L3_2	60	3	3	9	306	4	1620	175,5	175,5	175,5 (=)	175,5 (=)	0,0%	66%
Set_60.R3.L3_3	60	3	3	9	339	5	1620	236	236	236 (=)	236 (=)	0,0%	49%
Set_60.R3.L3_4	60	3	3	9	319	5	1620	225	225	221 (<)	224,2 (<)	-1,8%	48%
Set_60.R4.L2_1	60	4	2	8	189	4	1440	184	184	184 (=)	184 (=)	0,0%	42%
Set_60.R4.L2_2	60	4	2	8	182	4	1440	286	286	286 (=)	286 (=)	0,0%	42%
Set_60.R4.L2_3	60	4	2	8	203	5	1440	282	282	282 (=)	282 (=)	0,0%	43%
Set_60.R4.L2_4	60	4	2	8	192	5	1440	279	279	279 (=)	279 (=)	0,0%	44%
Set_60.R4.L3_1	60	4	3	12	319	4	2160	251,5	251,5	251,5 (=)	251,5 (=)	0,0%	50%
Set_60.R4.L3_2	60	4	3	12	313	4	2160	268,5	269,4	264,5 (<)	265,3 (<)	-1,5%	71%
Set_60.R4.L3_3	60	4	3	12	320	5	2160	283,5	288,4	283,5 (=)	283,5 (<)	0,0%	80%
Set_60.R4.L3_4	60	4	3	12	352	5	2160	277	277	275 (<)	276 (<)	-0,7%	73%
Set_90.R3.L2_1	90	3	2	6	294	4	2430	254	254	254 (=)	254 (=)	0,0%	43%
Set_90.R3.L2_2	90	3	2	6	272	4	2430	255	255	255 (=)	255 (=)	0,0%	43%
Set_90.R3.L2_3	90	3	2	6	292	5	2430	212	212	212 (=)	212 (=)	0,0%	42%
Set_90.R3.L2_4	90	3	2	6	269	5	2430	176	176	176 (=)	176 (=)	0,0%	43%
Set_90.R3.L3_1	90	3	3	9	445	4	3645	182,5	182,5	182,5 (=)	182,5 (=)	0,0%	71%
Set_90.R3.L3_2	90	3	3	9	448	4	3645	174,5	174,5	174,5 (=)	174,5 (=)	0,0%	48%
Set_90.R3.L3_3	90	3	3	9	557	5	3645	270,5	272,5	270,5 (=)	272,3 (<)	0,0%	72%
Set_90.R3.L3_4	90	3	3	9	545	5	3645	220	220	220 (=)	220 (=)	0,0%	51%
Set_90.R4.L2_1	90	4	2	8	324	4	3240	289	289,4	289 (=)	290,6 (>)	0,0%	43%
Set_90.R4.L2_2	90	4	2	8	295	4	3240	336	336	336 (=)	336 (=)	0,0%	72%
Set_90.R4.L2_3	90	4	2	8	297	5	3240	299	299	299 (=)	299 (=)	0,0%	42%
Set_90.R4.L2_4	90	4	2	8	307	5	3240	246	247,8	246 (=)	247,2 (<)	0,0%	42%
Set_90.R4.L3_1	90	4	3	12	515	4	4860	260	260	260 (=)	260 (=)	0,0%	50%
Set_90.R4.L3_2	90	4	3	12	479	4	4860	246,5	246,5	244,5 (<)	245,3 (<)	-0,8%	73%
Set_90.R4.L3_3	90	4	3	12	544	5	4860	323,5	323,7	323 (<)	324 (>)	-0,2%	53%
Set_90.R4.L3_4	90	4	3	12	466	5	4860	245,5	246,4	247 (>)	247 (>)	0,6%	46%
							Average values	248,64	249,03	248,05 (<)	248,47 (<)	-0,23%	53,14%

6. Conclusions

This paper introduces the MRCPSP-Z to respond to the particularities of an AAL workstation, where the main objective is to minimise the total labour cost given a fixed cycle time per station. The problem has been defined as a variant of the MRCPSP, considering additional workers for each activity, different workers' proficiency and spatial constraints in the work zones.

First, the problem was developed as a MILP model. Then, since the problem is NP-hard, two algorithms based on the Ant Colony System were developed, the standard ACS and a memetic ACS that includes a VND algorithm for the local search phase. Both algorithms considered two different types of pheromones, two types of heuristic information and SSGS for solution representation. Furthermore, a resource peak reduction mechanism was implemented to find solutions that SSGS cannot reach.

As the problem has not been addressed in previous research, computational tests were performed using new benchmark instances adapted to the characteristics of the presented MRCPSP-Z. For small size sets, ACS and M-ACS solutions were compared with exact method solutions. The results show a high percentage of optimal solutions found and a reduced relative average deviation, justifying the effectiveness of the proposed solution approaches.

Medium and large size sets were also solved to demonstrate the efficiency and robustness of the two algorithms, concluding with a slightly better performance of M-ACS compared to ACS. After evaluating the effectiveness of the VND, the exploration phase results to be more efficient than the intensification phase for this problem where cost is the objective. This is due to the difficulty in finding neighbours solutions that could lead to a reduction of TLC.

Finally, this paper fills the lack of existing applications in the AAL literature that consider real factory characteristics, providing promising solutions in reasonable CPU time. The results are also relevant for researchers, as they present a new problem that has not been addressed before.

Further research could focus on developing new solution approaches to solve the MRCPSP-Z. The problem could be extended with a multi-objective approach to include additional objectives relevant to the industry, or to include multi-skilled workers and stochastic processing times that are characteristic of other industries.

Data Availability Statement

The data that support the findings of this study are openly available in ["figshare"] at http://doi.org/10.6084/m9.figshare.21400035.

References

Afshar, M. R., Shahhosseini, V., & Sebt, M. H. (2022). A genetic algorithm with a new local search method for solving the multimode resource-constrained project scheduling problem. *International Journal of Construction Management*, *22*(3), 357-365.

Ahmadpour, S., & Ghezavati, V. (2019). Modeling and solving multi-skilled resource-constrained project scheduling problem with calendars in fuzzy condition. *Journal of Industrial Engineering International*, *15*(1), 179-197.

Arkhipov, D., Battaïa, O., Cegarra, J., & Lazarev, A. (2018). Operator assignment problem in aircraft assembly lines: a new planning approach taking into account economic and ergonomic constraints. *Procedia CIRP, 76*, 63-66.

Baradaran, S., Ghomi, S. F., Ranjbar, M., & Hashemin, S. S. (2012). Multi-mode renewable resource-constrained allocation in PERT networks. *Applied Soft Computing*, *12*(1), 82-90.

Blazewicz, J., Lenstra, J. K., & Kan, A. R. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete applied mathematics*, *5*(1), 11-24.

Borreguero, T. (2019). Scheduling with limited resources along the aeronautical supply chain: from parts manufacturing plants to final assembly lines. PhD diss., E.T.S.I. Industriales (UPM).

Borreguero, T., García, A., & Ortega, M. (2015). Scheduling in the aeronautical industry using a mixed integer linear problem formulation. *Procedia engineering*,*132*, 982-989.

Cheng, H., & Chu, X. (2012). Task assignment with multiskilled employees and multiple modes for product development projects. *The International Journal of Advanced Manufacturing Technology*, *61*(1), 391-403.

Chiang, C. W., & Huang, Y. Q. (2012). Multi-mode resource-constrained project scheduling by ant colony optimization with a dynamic tournament strategy. In *2012 Third international conference on innovations in bio-inspired computing and applications* (pp. 110-115). IEEE.

Correia, I., Lourenço, L. L. & Saldanha-da-Gama, F. (2012). Project scheduling with flexible resources: formulation and inequalities. *OR Spectrum*, *34*, 635–663.

Dolgui, A., Kovalev, S., Kovalyov, M. Y., Malyutin, S., & Soukhal, A. (2018). Optimal workforce assignment to operations of a paced assembly line. *European Journal of Operational Research*, *264*(1), 200-211.

Dorigo, M., & Stützle, T. (2019). Ant colony optimization: overview and recent advances. *Handbook of metaheuristics*, 311-351.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine*, *1*(4), 28-39.

Duarte, A., Mladenovic, N., Sánchez-Oro, J., & Todosijević, R. (2018). Variable neighborhood descent. In *Handbook of Heuristics*, Springer International Publishing, 341-367.

Heike, G., Ramulu, M., Sorenson, E., Shanahan, P., & Moinzadeh, K. (2001). Mixed model assembly alternatives for low-volume manufacturing: the case of the aerospace industry. *International Journal of Production Economics*, *72*(2), 103-120.

Kadrou, Y., & Najid, N. M. (2006). A new heuristic to solve RCPSP with multiple execution modes and Multi-Skilled Labor. In *The Proceedings of the Multiconference on Computational Engineering in Systems Applications* (Vol. 2, pp. 1302-1309). IEEE.

Khalilzadeh, M. (2015). A honey bee swarm optimization algorithm for minimizing the total costs of resources in MRCPSP. *Indian Journal of Science and Technology*, *8*(11).

Kolisch, R., & A. Sprecher. (1997). PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. *European journal of operational research, 96*(1), 205-216.

Li, C., Wang, F., & Chung, T. (2024). Multi-mode multi-skill resource-constrained project scheduling problem with differentiated professional capabilities. *Journal of Project Management*, *9*(1), 27-44.

Li, H., & Zhang, H. (2013). Ant colony optimization-based multi-mode scheduling under renewable and non-renewable resource constraints. *Automation in construction*, *35*, 431-438.

Miralles, C., Garcia-Sabater, J. P., Andrés, C., & Cardos, M. (2007). Advantages of Assembly Lines in Sheltered Work Centres for Disabled. A Case Study. *International Journal of Production Research*, *110*(2), 187–197.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers and operations research*, *24*(11), 1097-1100.

Molina, J. C., Salmeron, J. L., & Eguia, I. (2020). An ACS-based memetic algorithm for the heterogeneous vehicle routing problem with time windows. *Expert Systems with Applications*, *157*, 113379.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, *826*, 1989.

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research, 280*(2), 395-416.

Povéda, G., Alvarez, N., & Artigues, C. (2023). Partially Preemptive Multi Skill/Mode Resource-Constrained Project Scheduling with Generalized Precedence Relations and Calendars. In *29th International Conference on Principles and Practice of Constraint Programming (CP 2023).* Leibniz International Proceedings in Informatics (LIPIcs), Volume 280, pp. 31:1-31:21

Reed, M., Yiannakou, A., & Evering, R. (2014). An ant colony algorithm for the multi-compartment vehicle routing problem. *Applied Soft Computing*, *15*, 169-176.

Ritt, M., Costa, A. M., & Miralles, C. (2016). The assembly line worker assignment and balancing problem with stochastic worker availability. *International Journal of Production Research*, *54*(3), 907-922.

Russell, A., & Taghipour, S. (2019). Multi-objective optimization of complex scheduling problems in low-volume low-variety production systems. *International Journal of Production Economics*, *208*, 1-16.

Shahnazari-Shahrezaei, P., Zabihi, S., & Kia, R. (2017). Solving a multi-objective mathematical model for a multi-skilled project scheduling problem by particle swarm optimization and differential evolution algorithms. *Industrial Engineering and Management Systems*, *16*(3), 288-306.

Shan, M., Hong, Q., & Juan, W. (2007). Multi-mode multi-project scheduling problem for mould production in MC enterprise. In *2007 International Conference on Wireless Communications, Networking and Mobile Computing* (pp. 5316-5320). IEEE.

Shen, H., & Li, X. (2013). Cooperative discrete particle swarms for multi-mode resource-constrained projects. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 31-36). IEEE.

Taguchi, G., Chowdhury, S., & Wu. Y. (2005). *Taguchi's Engineering Quality Handbook*, John Willey and Sons. *Inc, Hoboken, New Jersey*.

Van Peteghem, V., & Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, *235*(1), 62-72.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., & Tavares, L. V. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, *187*(2), 511-524.

Vinay, V. P., & Sridharan, R. (2013). Taguchi method for parameter design in ACO algorithm for distribution–allocation in a two-stage supply chain. *The International Journal of Advanced Manufacturing Technology, 64*(9), 1333-1343.

Wuliang, P., Min, H., & Yongping, H. (2014). An improved ant algorithm for Multi-mode Resource Constrained Project Scheduling Problem. *RAIRO-Operations Research*, *48*(4), 595-614.

Xu, G., Lin, H., Cheng, Y., & Li, S. (2023). An Improved Ant Colony Optimization for Solving Task Scheduling Problem in Radar Signal Processing System. *Journal of Signal Processing Systems*, 1-18.

Zhang, H. (2012). Ant colony optimization for multimode resource-constrained project scheduling. *Journal of Management in Engineering*, *28*(2), 150-159.