



Combinatorial optimization methods for yarn dyeing planning

Ege Duran¹ · Cemalettin Ozturk² · M. Arslan Ornek³

Accepted: 15 April 2024
© The Author(s) 2024

Abstract

Managing yarn dyeing processes is one of the most challenging problems in the textile industry due to its computational complexity. This process combines characteristics of multidimensional knapsack, bin packing, and unrelated parallel machine scheduling problems. Multiple customer orders need to be combined as batches and assigned to different shifts of a limited number of machines. However, several practical factors such as physical attributes of customer orders, dyeing machine eligibility conditions like flote, color type, chemical recipe, and volume capacity of dye make this problem significantly unique. Furthermore, alongside its economic aspects, minimizing the waste of natural resources during the machine changeover and energy are sustainability concerns of the problem. The contradictory nature of these two makes the planning problem multi-objective, which adds another complexity for planners. Hence, in this paper, we first propose a novel mathematical model for this scientifically highly challenging yet very practical problem from the textile industry. Then we propose Adaptive Large Neighbourhood Search (ALNS) algorithms to solve industrial-size instances of the problem. Our computational results show that the proposed algorithm provides near-optimal solutions in very short computational times. This paper provides significant contributions to flexible manufacturing research, including a mixed-integer programming model for a novel industrial problem, providing an effective and efficient adaptive large neighborhood search algorithm for delivering high-quality solutions quickly, and addressing the inefficiencies of manual scheduling in textile companies; reducing a time-consuming planning task from hours to minutes.

Keywords Batching scheduling · MIP · ALNS · Dyeing

Cemalettin Ozturk and M. Arslan Ornek have contributed equally to this work.

Extended author information available on the last page of the article

1 Introduction and problem description

The textile industry plays a crucial role in the global economy, providing essential goods for daily life, such as clothing, blankets, and household items (Research 2020). Textile production is a complex process that involves six primary stages as shown in Fig. 1, including yarn spinning and dyeing, warp making, starching, weaving, finishing and dyeing, and cutting and sewing (Karacapilidis and Pappis 1996).

The textile dye industry produces about 3600 different types of products (Kant 2012), which can make scheduling and planning easily intractable. Advanced algorithms are needed to tackle this complexity.

In this paper, the first stage of the textile manufacturing process, yarn dyeing will be considered. Yarn dyeing occurs after the fibres have been spun into yarn, where the dye penetrates the strands to the centre of the yarn. This process aims to produce coloured yarns, which can be used to make striped knit or woven textiles, solid-dyed yarn fabrics, and sweaters. The study focuses on planning the spinning and dyeing process, the first step of textile production. Yarn is coloured as packages, which are small reels fixed on hollow spindles. Package dyeing, which involves dyeing yarn coiled on perforated cores, is one of the most widely used yarn dyeing procedures. The dye passes through to the yarn package with the aid of the tube package's designed perforations. Once the carrier of coloured yarn has been fully depleted, it is removed from the vessel. Readers are referred to textiletutorials.com (2023) for more technical details of the process with visual material.

Figure 2 provides a schematic representation of the Yarn Dyeing Planning (YDP) described by the industry experts during the development of this manuscript and validated with literature (Karacapilidis and Pappis 1996). Assume that there are 13 customer orders (jobs) to be assigned to one of the three shifts ($P = 3$) of K eligible Dyeing Machines (DM) on each day of the planning period of G days. Each machine can work at only one of the predefined *levels* which is chosen based on the attributes of jobs (total weight, volume, etc.) assigned to it. Furthermore, to reduce DM cleaning times water and chemical usage between consecutive batches on a machine, the color code in consecutive batches must be in ascending order while respecting the weight and volume capacity of DMs in each batch. This way, not only the cost but the environmental impact of the dyeing process is minimized. YDP has two objectives lexicographically ordered based on their priority: maximizing the number of jobs processed and minimizing the number of DMs used. These priorities are defined by the industry experts during the consultancy. The industry employs over 8000 types of chemicals in various textile manufacturing processes, including dyeing and printing. Many of these chemicals harm human health, either directly or indirectly. Textile processing, dyeing, and printing are water-intensive operations, with an average-sized textile mill, consuming approximately 1.6 million liters of water, of which 16% is

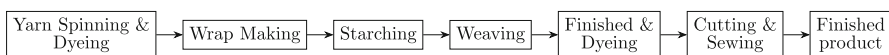


Fig. 1 The textile manufacturing process (Karacapilidis and Pappis 1996)

used in dyeing processes. Specifically, yarn dyeing requires about 60 *lt. of water per kilogram of yarn, and the dyeing section contributes to 15–20% of the total wastewater flow* (Kant 2012). As a result, the optimization goal was set to reduce the amount of dye machines used. The idea behind this method is that by lowering the total number of dye machines in use, chemical and water usage will also drop.

Figure 2 shows the assignment of those 13 jobs into three shifts of machines *machine*₁ and *machine*₃ in days 1 and *g* respectively. Selected levels of *machine*₁ are 1, 3, and 2 while 1, 2 and 3 for *machine*₃. As an example, the solution depicts that jobs 2, 4, and 8 are assigned to the second batch of *machine*₁ which works in *level*₃ based on the total weight, volume, and flotte of the jobs assigned to this machine.

The problem demonstrated above has several similarities with multidimensional knapsack (Kellerer et al. 2004), bin packing (Delorme et al. 2016), and unrelated parallel machine scheduling problems (Fleszar and Hindi 2018) in the literature:

- As in the multidimensional knapsack problem, YDP has several capacities (weight, volume, number) and flotte conditions that provide an upper bound for the number of jobs assigned to each DM in each batch.
- The number of available machines is known as in knapsack problems; however, the second objective of YDP is minimizing the number of DMs which converges the problem to bin packing problems.
- The color code of the consecutive batches (shifts) of each DM each day must be in non-decreasing order, bringing a sequencing dimension as in an unrelated parallel machine scheduling problem.

Therefore, YDP can be reduced to these three well-known combinatorial optimization problems; hence, it is an NP-hard problem.

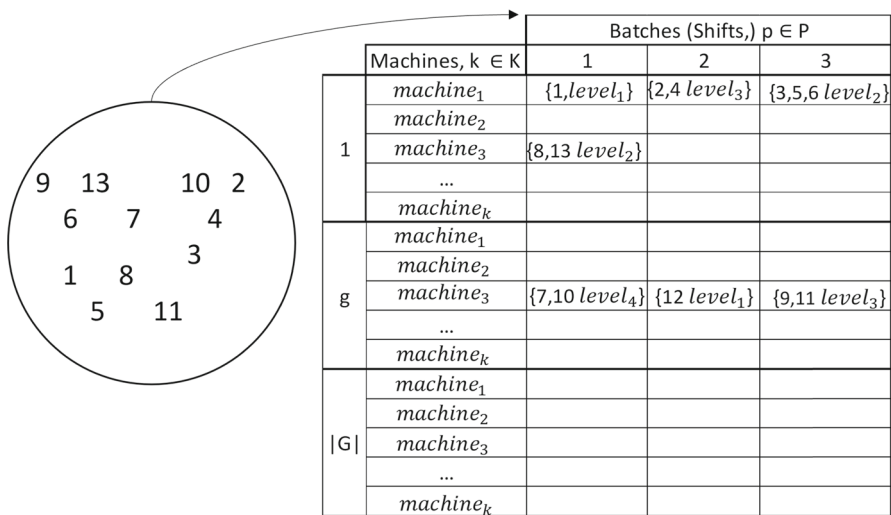


Fig. 2 An illustration of Yarn Dyeing Planning problem

However, the following conditions of YDP make it different from classical multidimensional knapsack, bin packing, and unrelated parallel machine scheduling problems:

- Capacities of a DM (volume, weight, number, and flotte) are also decision variables and selected from technologically predefined levels according to the assigned jobs into that DM.
- There are nonlinear eligibility conditions between jobs assigned to the same batch of the same DM.
- Sequencing colour code of consecutive batches requires time-bucket (Swamidass 2000) based formulation as in lot sizing problems (Oztürk and Ornek 2010) in which each bucket corresponds to a batch in a DM on a given day. This property makes YDP different from unrelated parallel machine scheduling where disjunctive scheduling-based formulations are needed to sequence individual jobs.

Therefore, this paper will describe YDP using a mixed integer programming (MIP) model. However, the industrial size of the YDP is intractable to solve by using exact solution methods such as mathematical programming in terms of time and space complexity. Hence, we propose an adaptive large neighbourhood search algorithm (ALNS) (Pisinger and Ropke 2019). ALNS exploits both systematic search capability and scalability of local search. This gives the power to deal with optimization problems with extremely large instances with close to optimal solution quality. The basic principle of ALNS is finding an initial solution, generating sub-problems, and solving them iteratively until a stopping condition is met. In the proposed ALNS we will demonstrate two types of initial solution finding mechanisms; one relies on solving the proposed MIP model until the first feasible solution is found and the other is a matheuristic (Fischetti and Fischetti 2016) algorithm. The selection of the subproblems in ALNS has a significant impact on the performance of the overall algorithm. Hence, we propose three subproblem selection (destroy) methodologies; job group-based, DM based, and ad-hoc (random). Dynamically changing subproblem sizes in each iteration provides adaptability to the proposed large neighbourhood search algorithm.

In contrast to previous publications in this domain, this paper brings the following contributions: (1) a novel mixed integer programming model formulation for a novel problem from the textile industry (2) a very fast adaptive large neighborhood search algorithm providing high-quality solutions in a very short computation time. (3) In practice, the textile company, that we worked for during the development of this study, often relies on manual scheduling and planning processes, which can be time-consuming and inefficient, requiring half a day and several people to develop a feasible plan. In contrast, the ALNS algorithm provides a highly efficient solution, reducing this task to a matter of minutes requiring just a single planning staff. (4) extensive experiments with publicly available realistic instances for further research.

The paper is structured into five sections, with Sect. 2 providing a literature review. Section 3 explains the YDP formally with a mixed integer programming model and a lexicographic solution scheme for the model. An adaptive large neighbourhood algorithm to solve large instances of YPD is proposed in Sect. 4. The computational results are presented in Sect. 5, and the paper concludes by elaborating on the results and discussing future studies in Sect. 6.

2 Literature review

This study focuses on the initial step of textile production, which is the spinning and dyeing process of yarn. Yarn is dyed as small packages fixed on hollow spindles, using the widely used package dyeing method where the dye penetrates through the perforations of the yarn package. Scheduling textile production is complex, involving planning multiple products and manufacturing phases using Dyeing Machines with different capacities and levels. The varied nature of the textile production system makes production management challenging, with different planning horizons and output characteristics for each process. Scheduling problems have gained attention due to their wide application in industry and computational complexity, and researchers have focused on multi-stage and complex scheduling to find the most appropriate quantity and timing of production orders based on customer demands and resource capacities (Sáenz-Alanís et al. 2016; Öztürk and Ornek 2014).

Researchers have extensively studied scheduling problems involving multi-stage and complex manufacturing processes. Various mathematical formulations, heuristic algorithms, and decomposition methods have been proposed to solve these problems.

The problem of scheduling the yarn dyeing process involves parallel machines and batch processing. Various studies have been conducted to address similar issues, such as single-machine batch scheduling with job families and setup requirements (Ghosh and Gupta 1997), parallel machine batch scheduling with sequence-dependent setup times in a brewing company (Sáenz-Alanís et al. 2016), scheduling of parallel batching machines with non-identical machine capacities and inclusive processing set constraints (Li 2017), and single batch-processing machine problem with non-identical task sizes and release dates (Zhou et al. 2018). These studies have proposed different mathematical formulations, heuristic algorithms, and optimisation techniques to minimize maximum lateness or makespan, which are critical objectives in manufacturing companies. Overall, scheduling problems are complex and require different approaches depending on the production system's characteristics and requirements.

Li et al. (2009) developed an Ant Colony Optimization (ACO) algorithm to solve the parallel batch processing machines problem with incompatible job families, dynamic job arrivals, and sequence-dependent setup time constraints.

Crauwels et al. (2006) addressed a batch scheduling problem with parallel machines, job families, and family-dependent setup times. They partitioned jobs into families and required a family-specific setup time at the beginning of each period and batch. They developed an integer program to minimize the number of overloaded periods and total overtime, and the program provided benchmark results for the heuristic method.

Although there has been extensive research on scheduling problems in various industries, there has been limited research on scheduling problems specific to the textile production process. This is partly due to the complex nature of the textile industry and the difficulty in developing software systems that can handle all aspects of textile production (Öztürk and Ornek 2014). However, some studies have been conducted on the programming of textile production.

According to Periyasamy and Militky (2020) and Kant (2012), the textile industry is a major contributor to environmental sustainability issues, with the dyeing process having a significant impact on pollution, water usage, and energy consumption. Hynes

et al. (2020) note that Industry 4.0 technologies can be used to implement multi-objective optimization techniques for dyeing processes, enabling the minimization of water consumption, treatment costs, and environmental pollution by optimizing manufacturing resources.

Zhang et al. (2017) investigate a multi-objective ABC algorithm that focuses on solving the production scheduling problem in fabric dyeing processes. They view the problem as a model for parallel batch machine scheduling. In a separate study, Hsu et al. (2009) examine the scheduling strategy for producing yarn-dyed textiles, which includes multi-stage production, hierarchical product structure, sequence-dependent setup times, and group delivery requirements. The aim is to minimize the total tardiness of customer orders. Li et al. (2021) tackle a parallel machine scheduling problem with several color families, sequence-dependent setup times, and machine eligibility constraints. They create an integer programming model to minimize the overall delay. Huynh and Chien (2018) researched related to Industry 4.0 migration and developed a multi-subpopulation genetic algorithm with embedded heuristics (MSGGA-H) to optimize the textile batch dyeing scheduling process. The goal of the study was to minimize the makespan, which is the time required to complete a batch, and improve the efficiency of the dyeing process, which is a bottleneck in textile production. For a more detailed study on setups in the dyeing process, see (Gomes et al. 2021). Demir and Kemal (2022) have created an integer linear model and iterative greedy-based heuristics for bobbin boiler planning in the textile industry. EROĞLU et al. (2014) have developed a genetic algorithm to address loom scheduling with sequence-dependent setup times. A data-driven, simulation-based method to optimize the scheduling of operations to increase the manufacturing process' sustainability is provided by Pirola et al. (2021). In another research, Zhou et al. (2020), authors presented an optimization model for scheduling dyeing processes in knitting companies. It utilizes a hybrid genetic algorithm with variable neighborhood search to improve production schedules, and minimize delay and switching costs. Finally, Karacizmeli and Ogulata (2017) have formulated a mixed-integer programming model for scheduling textile finishing plants.

3 Problem formulation and lexicographic solution scheme

This section presents a mixed integer programming (MIP) formulation to model and solve the multi-period multi-objective yarn dyeing planning problem (YDP) described in the previous section. The formulation structure is shown conceptually in Fig. 3, in which input parameters and variables related to DMs and jobs, constraints, and objectives are included.

3.1 Multi-objective mixed integer programming model for YDP

This section formally defines the multiobjective YDP using a MIP formulation.

Sets and Indices

I : set of job orders, $i \in I = \{1, 2, \dots, |I|\}$

P : set of batches, $p \in P = \{1, 2, \dots, |P|\}$

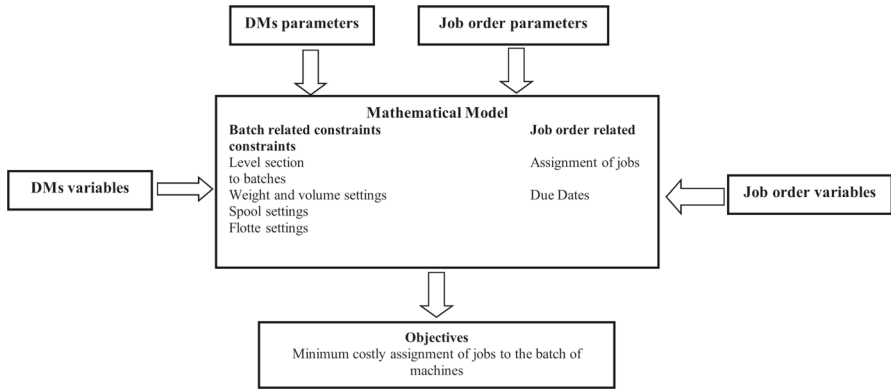


Fig. 3 Structure of the MIP model

G : set of days, $g \in G = \{1, 2, \dots, |G|\}$
 K : set of dye machines, $k \in K = \{1, 2, \dots, |K| + 1\}$, dye machine $|K| + 1$ is dummy DM in which any job that can not be assigned to any real DM is assigned
 S_k : set of levels in dye machine k , $m \in S_k = \{1, 2, \dots, |level_{S_k}|\}$
 F : set of flotte intervals, $f \in F = \{1, 2, \dots, |F|\}$
 R : set of chemical recipe code, $r \in R = \{1, 2, \dots, |R|\}$
 C : set of colour percentage, $c \in C = \{1, 2, \dots, |C|\}$
 $machine_i$: eligible dye machines for job order i
 $machine_i = K - \{k \in K \mid ((reactive_i=1) \vee (lyc_i=1)) \wedge (special_k=1)\}$

Parameters

Job-related parameters

$orderno_i$: number of the job order
 kg_i : weight of the job order
 $orderrecipe_i$: chemical recipe of the customer order
 $ordercolour_i$: colour percentage of the job order
 $duedate_i$: due date of the job order
 $latest_i$: latest possible time for finishing the job order
 $orderflotte_{i,f} = \begin{cases} 1 & \text{if flotte interval of job } i \text{ is } f \\ 0 & \text{otherwise} \end{cases}$
 $reactive_i = \begin{cases} 1 & \text{if job } i \text{ is reactive} \\ 0 & \text{otherwise} \end{cases}$

$lyc_i = \begin{cases} 1 & \text{if job } i \text{ is spandex} \\ 0 & \text{otherwise} \end{cases}$

$spool$: weight of a spool

Dye machine-related parameters

$machinenok$: index of dye machine
 $special_k = \begin{cases} 1 & \text{if machine } k \text{ is eligible for reactive or spandex job orders} \\ 0 & \text{otherwise} \end{cases}$
 $Level_{S_k}$: total number of possible levels of dye machine k

$volumecapacity_m$: volume capacity of machine k at level m where $m \in S_k$
 $maxquantity_m$: number of spools capacity of machine k at level m where $m \in S_k$
 $minkg_m$: minimum processing weight of machine k at level m where $m \in S_k$
 $maxkg_m$: maximum processing weight of machine k at level m where $m \in S_k$
 $initialcolour_k$: initial colour percentage of dye machine k at the beginning of the

planning period

$maxflotte_f$: maximum allowed flotte value for flotte interval f

$minflotte_f$: minimum allowed flotte value for flotte interval f

Note that flotte intervals are defined for both jobs and batches and the flotte value of a dye machine is defined as the ratio of volume to the weight of dyestuff

In the following, we define big-M values used to linearise several constraints. We define different values to ensure the tightest possible right-hand side values for each constraint where a big M is needed.

M : a big number

M_2 : a big number for colour constraints ($\max_{\forall i} \{ordercolour_i\}$)

M_3 : a big number for receipt constraints ($\max_{\forall i} \{orderrecipe_i\}$)

M_4 : a big number for machine kg constraints ($\max_m \{maxkg_m\}$)

M_5 : a big number for a number of spool constraints ($\max_m \{maxquantity_m\}$)

Variables

Variables related to dye machines

$weight_{g,p,k}$: total weight of batch p at dye machine k on day g , $weight_{g,p,k} \in \mathbb{R}^+$

$volume_{g,p,k}$: total volume of batch p at dye machine k on day g , $volume_{g,p,k} \in \mathbb{R}^+$

$quantity_{g,p,k}$: total number of spools of batch p at dye machine k on day g , $quantity_{g,p,k} \in \mathbb{Z}^+$

$colour_{g,p,k}$: colour percentage of batch p at dye machine k on day g , $colour_{g,p,k} \in \mathbb{R}^+$

$recipe_{g,p,k}$: colour code of batch p at dye machine k on day g $recipe_{g,p,k} \in \mathbb{R}^+$

$flotte_{g,p,k,f} = \begin{cases} 1 & \text{if batch } p \text{ at dye machine } k \text{ on day } g \text{ is in flotte interval } f \\ 0 & \text{otherwise} \end{cases}$

$level_{g,p,m} = \begin{cases} 1 & \text{if level of batch } p \text{ at dye machine } k \text{ on day } g \text{ is level } m \\ 0 & \text{otherwise} \end{cases}$

$usage_{g,p,k} = \begin{cases} 1 & \text{if a job order is assigned to batch } p \text{ at dye machine } k \text{ on day } g \\ 0 & \text{otherwise} \end{cases}$

$machineusage_k = \begin{cases} 1 & \text{if dye machine } k \text{ is used} \\ 0 & \text{otherwise} \end{cases}$

Variables related to job orders

$assignment_{g,p,k,i} = \begin{cases} 1 & \text{if job } i \text{ is assigned to batch } p \text{ of dye machine } k \text{ at day } g \\ 0 & \text{otherwise} \end{cases}$

α : the number of jobs that are not assigned to any DMs

β : the number of DMs used

Now, we propose the complete mixed integer programming model:

$$\begin{aligned}
 \text{Minimize } \alpha &= \sum_{g \in G} \sum_{p \in P} \sum_{i \in I} \text{assignment}_{g,p,|K|+1,i} \\
 \text{Minimize } \beta &= \sum_{k \in K | k \leq |K|} \text{machineusage}_k
 \end{aligned} \tag{1}$$

where $\alpha \preceq \beta$

The lexicographic objective function 1 first maximizes the number of jobs processed by minimizing the number of jobs assigned to the dummy machine $|K| + 1$. Then it minimizes the number of DMs used.

Subject to

Constraints for assigning jobs to different batches of dye machines

$$\sum_{g \in G, p \in P, k \in |K|+1} \text{assignment}_{g,p,k,i} = 1, \forall i \in I \tag{2}$$

Constraint (2) ensures that each job order ($\forall i \in I$) has to be assigned to exactly one of the batches ($\forall p \in P$) of a DM ($\forall k \in \text{machine}_i$) in a day ($\forall g \in G$).

Constraints for selecting a level for each batch

$$\sum_{m \in S_k} \text{level}_{g,p,m} \leq 1 \forall g \in G, \forall p \in P, \forall k \in K | k \leq |K| \tag{3}$$

$$\begin{aligned}
 \sum_{i \in I | k \in \text{machine}_i} \text{assignment}_{g,p,k,i} &\leq M * \text{usage}_{g,p,k} \\
 \forall g \in G, \forall p \in P, \forall k \in K | k \leq |K| &\tag{4}
 \end{aligned}$$

$$\text{usage}_{g,p+1,k} \leq \text{usage}_{g,p,k} \forall g \in G, \forall p \in P, \forall k \in K (p < |P|) \wedge (k < |K|) \tag{5}$$

$$\text{usage}_{g,p,k} = \sum_{m \in S_k} \text{level}_{g,p,m} \forall g \in G, \forall p \in P, \forall k \in K | k < |K| \tag{6}$$

Constraint (3) guarantees that each DM ($\forall k \in K$) can be used at one of its levels ($\forall m \in s_k$) in each batch ($\forall p \in P$) on any day ($\forall g \in G$).

Constraint (4) is the linear formulation of logical proposition $\sum_{i \in I | k \in \text{machine}_i} \text{assignment}_{g,p,k,i} \geq 1 \implies \text{usage}_{g,p,k} = 1$ and mean that If at least one job order is assigned to a batch ($\forall p \in P$) of a DM ($\forall k \in K$) on any day ($\forall g \in G$), that dye machine is marked as used. Constraint (5) guarantees that batches ($\forall p \in P$) assigned to a machine ($\forall k \in K$) are used consecutively. In other words, a job can be assigned to a successor batch $p+1$ if and only if the batch p is used. Since the cost of producing in any batch of a given day is indistinguishable, constraint (5) is formulated as valid inequalities and used to break symmetries. Constraint (6) is the linear formulation of logical proposition $\text{usage}_{g,p,k} \Leftrightarrow \sum_{m \in S_k} \text{level}_{g,p,m}$ and ensure that if a dye machine ($\forall k \in K$) is used in a batch ($\forall p \in P$) of any day ($\forall g \in G$), it

can be used in one of its predefined levels ($m \in S_k$). In addition to constraints (3) and (4), if any job is not assigned to a batch of a dye machine on a particular day, there is no need to make a level decision for the same machine in that batch of the day. However, if at least one job is assigned to a level has to be selected by the model. We diversify $usage_{g,p,k}$ and $level_{g,p,j}$ variables to allow for the easier formulation of logical propositions (a) minimum weight condition is not compulsory if none of the job orders is assigned to that batch and (b) flotte selection for each batch which is explained in detail in constraints below.

Weight and volume constraints for each batch

$$weight_{g,p,k} = \sum_{i \in I | k \in machine_i} (kg_i * assignment_{g,p,k,i}) \quad \forall g \in G, \forall p \in P, \forall k \in K \quad (7)$$

$$weight_{g,p,k} \leq maxkg_m + M_4 * (1 - level_{g,p,m}) \quad \forall g \in G, \forall p \in P, \forall k \in K, \forall m \in S_k | k \leq |K| \quad (8)$$

$$weight_{g,p,k} \leq minkg_m - M_4 * (1 - level_{g,p,m}) \quad \forall g \in G, \forall p \in P, \forall k \in K, \forall m \in S_k | k \leq |K| \quad (9)$$

$$volume_{g,p,k} = \sum_{m \in S_k} (volumecapacity_m * level_{g,p,m}) \quad \forall g \in G, \forall p \in P, \forall m \in S_k | k \leq |K| \quad (10)$$

Constraint (7) returns the total weight of each batch ($\forall p \in P$) of every dye machine ($\forall k \in K$) for each day ($\forall g \in G$) as the sum of the weights of assigned job orders which must be in between the minimum and maximum weight interval of the selected level of that batch with the help of constraints (8) and (9). Similarly, constraint (10) ensures that the volume of a batch equals the volume of the selected level of that batch. Note that if the total volume of assigned jobs is not enough to fill the volume of the chosen level, scrap yarns are used.

Number of spools-related constraints for each batch

$$quantity_{g,p,k} \geq \frac{weight_{g,p,k}}{spool}, \quad \forall g \in G, \forall p \in P, \forall k \in K | k \leq |K| \quad (11)$$

$$quantity_{g,p,k} \leq \frac{weight_{g,p,k}}{spool} + 1, \quad \forall g \in G, \forall p \in P, \forall k \in K | k \leq |K| \quad (12)$$

$$quantity_{g,p,m} \leq maxquantity_m * level_{g,p,m} + M_5 * (1 - level_{g,p,m}) \quad \forall g \in G, \forall p \in P, \forall m \in S_k \forall k \in K | k \leq |K| \quad (13)$$

$$quantity_{g,p,m} \geq maxquantity_m * level_{g,p,m} + M_5 * (1 - level_{g,p,m})$$

$$\forall g \in G, \forall p \in P, \forall m \in S_k \forall k \in K |k| \leq |K| \tag{14}$$

The total number of spools in a batch of a dye machine in a day is equal to $quantity_{g,p,k} = \lceil \frac{weight_{g,p,k}}{bobin} \rceil$ which is linearized (making it integer) by constraints (11) and (12). The integer $quantity_{g,p,k}$ variable is greater than or equal to the float $\frac{weight_{g,p,k}}{bobin}$ (11) and less than or equal to float $\frac{weight_{g,p,k}}{bobin} + 1$ (12). For example, if $\frac{weight_{g,p,k}}{bobin} = 2.3$, the number of spools in that batch will be three because it is the only integer value between $2.3 \leq amount_{g,p,k} \leq 3.3$. Constraints (13) and (14) ensure that the number of spools in a batch is within the predefined minimum and the maximum number of spools of the assigned level. In this formulation, we assume that different levels of a dye machine are sorted in ascending order of the maximum number of spools allowed. For example, if there are two levels with spool capacity $maxamount_m=20$ and $maxamount_{m-1}=15$, the number of spools assigned to a batch with level j must be greater than 15 (constraint 13) and less than 20 (constraint 14).

Colour carryover constraints for each batch

$$colour_{1,1,k} \geq initialcolour_k, \forall k \in K |k| \leq |K| \tag{15}$$

$$colour_{g,1,k} \geq colour_{g-1,|P|,k}, \forall g \in G, \forall k \in K |(g > 1) \wedge (k \leq |K|) \tag{16}$$

$$colour_{g,p+1,k} \geq colour_{g,p,k}, \forall g \in G, \forall k \in K |(p > |P|) \wedge (k \leq |K|) \tag{17}$$

$$colour_{g,p,k} \leq M_2 * (initialcolour_k + \sum_{l \in P |l| \leq p} usage_{g,l,k})$$

$$\forall g \in G, \forall k \in K |(p > |P|) \wedge (k \leq |K|) \tag{18}$$

Because of the high cost and time for changeovers, a setup carryover restriction enforces that from shift to shift, the colours must be processed in the increasing degree of darkness, i.e., in technical terms, the colour percentage of the batch increases. Hence, while constraint (15) guarantees that the colour percentage of a dye machine at the beginning of the planning period is greater than or equal to the initial colour percentage of that machine, constraint (16) ensures that the colour percentage of the first batch of a successor day is greater than or equal to the colour percentage of the same dye machine in the last batch of the previous day. Similarly, constraint (17) sorts colour percentages of successive batches in a dye machine in ascending order for each dye machine. Constraint (18) is used as a valid inequality such that the color percentage variable is set to 0 if a machine batch is not used.

Flotte selection for each batch

$$usage_{g,p,k} = \sum_{f \in F} flotte_{g,p,k,f}, \forall g \in G, \forall p \in P, \forall k \in K |(k \leq |K|) \tag{19}$$

$$volume_{g,p,k} \geq weight_{g,p,k} * minflotte_f - M * (1 - flotte_{g,p,k,f}),$$

$$\forall g \in G, \forall p \in P, \forall f \in F, \forall k \in K |k| \leq |K| \tag{20}$$

$$\begin{aligned} volume_{g,p,k} &\leq weight_{g,p,k} * maxflotte_f + M * (1 - flotte_{g,p,k,f}), \\ \forall g \in G, \forall p \in P, \forall f \in F, \forall k \in K | k \leq |K| \end{aligned} \quad (21)$$

The flotte value of a batch is calculated by the nonlinear equation $\frac{volume_{g,p,k}}{weight_{g,p,k}}$ and must be in between predefined technologically possible intervals. Hence, first, we assign a flotte interval to each batch if it is used as in constraint (19) which is a linear formulation of a logical proposition $usage_{g,p,k} = 1 \Leftrightarrow \sum_{f \in F} flotte_{g,p,k,f}$. Since $minflotte_f \leq \frac{volume_{g,p,k}}{weight_{g,p,k}} \leq maxflotte_f$ is a non-linear constraint, it is converted into linear constraints in (20) and (21).

Constraints for ensuring consistency between technological requirements of jobs and batches of dye machines

$$\begin{aligned} orderflotte_{i,f} &\leq flotte_{g,p,k,f} + M * (1 - assignment_{g,p,k,i}), \\ \forall g \in G, \forall p \in P, \forall f \in F, \forall k \in K | (orderflotte_{i,f} = 1) \wedge (k \leq |K|) \end{aligned} \quad (22)$$

$$\begin{aligned} orderflotte_{i,f} &\geq flotte_{g,p,k,f} - M * (1 - assignment_{g,p,k,i}), \\ \forall g \in G, \forall p \in P, \forall f \in F, \forall k \in K | (orderflotte_{i,f} = 1) \wedge (k \leq |K|) \end{aligned} \quad (23)$$

$$\begin{aligned} ordercolour_i &\leq colour_{g,p,k} + M_2 * (1 - assignment_{g,p,k,i}), \\ \forall g \in G, \forall p \in P, \forall i \in I, \forall k \in K | k \leq |K| \end{aligned} \quad (24)$$

$$\begin{aligned} ordercolour_i &\geq colour_{g,p,k} - M_2 * (1 - assignment_{g,p,k,i}), \\ \forall g \in G, \forall p \in P, \forall i \in I, \forall k \in K | k \leq |K| \end{aligned} \quad (25)$$

$$\begin{aligned} orderrecipe_i &\leq recipe_{g,p,k} + M_3 * (1 - assignment_{g,p,k,i}), \\ \forall g \in G, \forall p \in P, \forall i \in I, \forall k \in K | k \leq |K| \end{aligned} \quad (26)$$

$$\begin{aligned} orderrecipe_i &\geq recipe_{g,p,k} - M_3 * (1 - assignment_{g,p,k,i}), \\ \forall g \in G, \forall p \in P, \forall i \in I, \forall k \in K | k \leq |K| \end{aligned} \quad (27)$$

$$recipe_{g,p,k} \leq M_3 * usage_{g,p,k} \quad \forall g \in G, \forall p \in P, \forall k \in K | k \leq |K| \quad (28)$$

Constraint (22) and (23) ensure that the flotte interval of a job order is the same with the selected flotte interval of the assigned batch and consequently with other jobs assigned to the same batch of the same dye machine on the same day. Similarly, constraints between (24)–(27) guarantee the consistency of colour percentage and recipe of individual job orders with assigned batches and other job orders assigned to the same batch of the same dye machine. Constraint (28) is used as a valid inequality such that if a batch of a machine is not used on a particular day, the recipe variable is set to 0.

Domains of variables

$$usage_{g,p,|K|+1} = 0, \forall g \in G, \forall p \in P \tag{29}$$

$$recipe_{g,p,|K|+1} \leq 0, \forall g \in G, \forall p \in P \tag{30}$$

$$colour_{g,p,|K|+1} \leq 0, \forall g \in G, \forall p \in P \tag{31}$$

$$level_{g,p,m} = 0, \forall g \in G, \forall p \in P \forall m \in S_k |k = |K| + 1 \tag{32}$$

$$weight_{g,p,m} \leq 0, \forall g \in G, \forall p \in P \forall m \in S_k |k = |K| + 1 \tag{33}$$

$$volume_{g,p,m} \leq 0, \forall g \in G, \forall p \in P \forall m \in S_k |k = |K| + 1 \tag{34}$$

$$amount_{g,p,|K|+1} = 0, \forall g \in G, \forall p \in P \tag{35}$$

$$flotte_{g,p,|K|+1,f} = 0, \forall g \in G, \forall p \in P, \forall f \in F \tag{36}$$

$$weight_{g,p,k}, volume_{g,p,k}, colour_{g,p,k}, recipe_{g,p,k} \geq 0$$

$$usage_{g,p,k} \in \{0, 1\}, quantity_{g,p,k} \leq \text{and integer}$$

$$\forall g \in G, \forall p \in P, \forall f \in F \tag{37}$$

$$flotte_{g,p,k,f} \in \{0, 1\}, \forall g \in G, \forall p \in P, \forall k \in K \forall f \in F \tag{38}$$

$$level_{g,p,m} \in \{0, 1\}, \forall g \in G, \forall p \in P, \forall m \in S_k \tag{39}$$

$$assignment_{g,p,k,i} \in \{0, 1\}, \forall g \in G, \forall p \in P, \forall k \in K, \forall i \in I \tag{40}$$

$$\sum_{g \in G} \sum_{p \in P} \sum_{i \in I} assignment_{g,p,k,i} \leq M * machineusage_k, \forall k \in K |k \leq |K| \tag{41}$$

$$machineusage_k \in \{0, 1\} \forall k \in K |k \leq |K| \tag{42}$$

All variables on the dummy dye machine are set to 0 with constraints (30)–(37) where the domains are given in (38)–(40). Constraint (41) indicates if there is any assignment to any of the DMs in any shift (batch) of any day, that machine is marked as being used, $machineusage_k = 1$. And finally constraint (42) gives the domain of machine usage variables.

3.2 Lexicographic solution scheme for the multi-objective YDP MIP model

MIP formulation in Sect. 3.1 has two objectives lexicographically ordered; minimizing the number of jobs assigned to dummy DM (α) with a minimum number of DMs, β . Based on the industry experts' feedback, there is a clear distinction between the priority of these two objectives, and hence a lexicographic solution method fits best for the resolution which also does not require normalizing goals as in the weighted sum method. The methodology relies on decomposing the multi-objective problem into many single objective problems and solving sequentially in which the solution (i.e., objective function value) of the upstream problem(s) is added as an upper bound (for minimization problems) or lower bound (for maximization problems) into the downstream problem(s). The optimal solution for all objectives is found when the last instance is solved, which also guarantees a Pareto optimal solution. Readers are referred to Zykina (2004) and Arora (2012) for details of the lexicographic method.

The same methodology is used for YDP problem formulation in Sect. 3.1: presented model is first solved by minimizing only the number of jobs assigned to the dummy machine, we call this model as *MIP* α and then the objective function value α^* is fed into the second MIP model, which we call as *MIP* β , as an upper bound for unassigned jobs while minimizing the number of DMs used, β . When optimal solutions are found for both models, the minimum number of unprocessed jobs, α^* and the minimum number of machines used β^* are ensured.

Figure 4 shows the lexicographic solution scheme. The formulation of *MIP* α and *MIP* β are detailed in the following subsections.

3.2.1 *MIP* α

$$\text{Minimize } \alpha = \sum_{g \in G} \sum_{p \in P} \sum_{i \in I} \text{assignment}_{g,p,|K|+1,i} \quad (43)$$

subject to

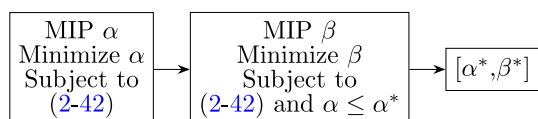
Constraints (2–42)

The solution of *MIP* α , is stored as α^* to be used as an upper bound for the number of unassigned jobs in the downstream model *MIP* β .

3.2.2 *MIP* β

Additional input parameters

Fig. 4 Lexicographical solution scheme



α^* : the number of jobs that are not assigned to any DMs based on the solution of *MIP* α

$$\text{Minimize } \beta = \sum_{k \in K} \text{machineusage}_k \tag{44}$$

subject to
(2–42) and

$$\sum_{a \in A | a.\text{machine} = |K| + 1} \text{assignment}_{g,p,k,i} \leq \alpha^* \tag{45}$$

Constraint (44) ensures that any solution to *MIP* β can not yield a worse solution for *MIP* α than α^* .

Optimal solutions of first the *MIP* α and then *MIP* β models result from the minimum number of jobs that are not processed, α^* and β^* [α^*, β^*]

In addition to the NP-hard computational complexity of the proposed MIP models, the size complexity of the model is bounded with $\mathcal{O}(|G| \times |P| \times |K| \times |I|)$ both for the number of variables and constraints due to $\text{assignment}_{g,p,k,i}$ variables and equation (27).

Nonlinear relations between variables make developing good-quality lower bounds for objective functions difficult. However, a practical lower bound can be formulated for β as in the following.

Recall that the colour percentage of a batch is one and only one and equal to the colour percentage of all jobs assigned to (15–18 and 24, 25). Assume five different colour percentages (groups of jobs) exist in the set of jobs to be processed, and only three batches are available. They need to be assigned to minimum ($\lceil \frac{5}{3} \rceil = 2$) machines. Therefore, $\sum_k \text{machineusage}_k \geq \lceil \frac{n}{b} \rceil$ where n and b represent the number of groups and the total number of batches in a day, respectively is a valid lower bound for β .

As presented in Sect. 5 in detail, the lexicographic solution scheme proposed above is intractable due to its size and computational complexity. Hence, in the next section, an Adaptive Large Neighborhood Search (ALNS) heuristic will be proposed to get good-quality solutions in a reasonable time for large instances. The ALNS heuristic differs from the conventional large neighbourhood search (LNS) heuristic by employing multiple destroy and repair methods in dynamically changing size during the search.

4 Adaptive large neighbourhood search algorithm for YDP

Divide-and-Conquer is the most practical method to attack sizeable combinatorial optimization problems, dividing the problem into computationally tractable but independent sub-problems and iteratively solving till a stopping criterion is met. Large neighbourhood search (LNS) proposed in Shaw (1998) is the primarily used divide-and-conquer method for combinatorial optimization problems from many domains. The basic working principle of a typical LNS is finding an initial solution, selecting

a subproblem, and iteratively destroying and repairing the current solution. As an example, for a mixed integer programming model-based initial solution, these phases would be selecting a set of variables to be destroyed (subproblem selection) at random or with a heuristic selection method, relaxing values of these variables (destruction) while fixing values of other variables, then solving (repair) the mixed integer programming model only for those whose values are relaxed. Since a combinatorial problem is solved in every repair phase, the method is called “large” compared to “local” search methods, where polynomial repair methods are used. If the size of the subproblems changes dynamically during the search, the overall process is called an adaptive large neighbourhood search (ALNS), Ropke and Pisinger (2006). Readers are referred to Taillard (2023) for a detailed explanation of generic LNS and ALNS procedures.

In this paper, we implemented the ALNS method into the Yarn Dyeing Planning (YDP) problem, demonstrating the hierarchy of steps in Fig. 5. Having a structural formulation of the problem through a mixed integer programming model provides a background for employing ALNS as it can be used for having a good quality initial solution and as a repair operator and promises quickly converging solutions compared to other metaheuristic methods. Developed ALNS uses two approaches to find an initial solution; a matheuristic method based on the YDP MIP model presented in Sect. 3.1 and the other based on executing the lexicographic scheme presented in Sect. 3.2 only until the first feasible solution is found. Once an initial solution is found, three alternative subproblem generation heuristics are executed; job and machine-based or random. A subproblem is selected based on descending or ascending ordering heuristics. The following subsection will explain the design of these steps demonstrated in Fig. 5.

4.1 Design of the ALNS algorithm

The ALNS metaheuristic uses a neighborhood defined implicitly by a destroy method and a repair method. The destroy method breaks down a part of the current solution,

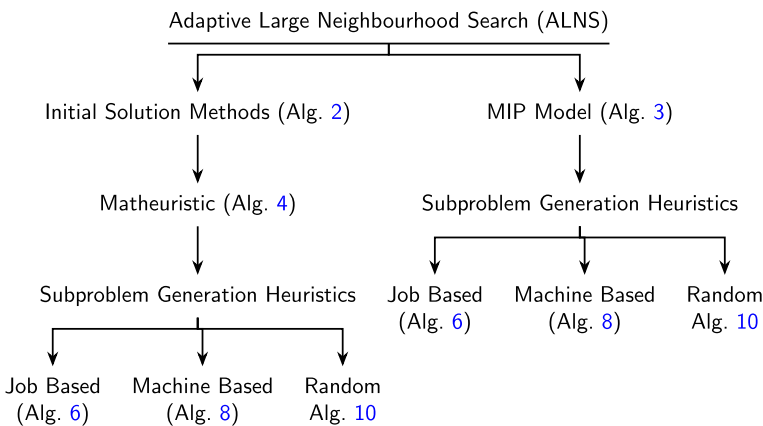


Fig. 5 Hierarchy and ALNS steps for Yarn Dyeing Problem

while the repair method rebuilds the destroyed part with the hope of a better objective. For avoiding to stuck in a local optima, the destroy method usually incorporates a degree of randomness so that a different part of the current solution is destroyed in each iteration. The function $d(\cdot)$ represents the destroy method, while $r(\cdot)$ represents the repair method. In more detail, when the function $d(x)$ is applied to a solution x , it returns a partially destroyed copy of x . On the other hand, using the function $r(\cdot)$ to a partially destroyed solution results in a repaired, feasible solution constructed from the destroyed one (Pisinger and Ropke 2019). Algorithm 1 shows the framework of ALNS.

Algorithm 1 ALNS (*machines,jobs,Reverse*)

```

1: RT=timlim
2: S=InitialSol(machines,jobs,Heuristic,timlim)
3:  $S^b = S$ 
4: while RT > 0 do
5:   if  $\alpha > 0$  then
6:      $S^t \leftarrow r(d^1(S, RT))$ 
7:     if  $S^t \prec S^b$  then
8:        $S^b \leftarrow S^t$ 
9:     end if
10:     $S^t \leftarrow r(d^2(S, RT))$ 
11:    if  $S^t \prec S^b$  then
12:       $S^b \leftarrow S^t$ 
13:    end if
14:  else
15:     $S^t \leftarrow r(d^2(S, RT))$ 
16:    if  $S^t \prec S^b$  then
17:       $S^b \leftarrow S^t$ 
18:    end if
19:  end if
20:   $S^t \leftarrow r(d^3(S, RT))$ 
21:  if  $S^t \prec S^b$  then
22:     $S^b \leftarrow S^t$ 
23:  end if
24: end while
25: Return  $S^b$ 

```

Algorithm 1 provides the adaptive large neighbourhood search algorithm's pseudo-code. ALNS starts with finding an initial solution (Line 2) with α which refers to the number of unassigned jobs, lexicographically the first objective function. Suppose there exists at least one unassigned job (line 5). In that case, ALNS applies $r(d^1(S,RT))$ (lines 6–9) that first destroys the current solution based on the job-based destroy operator and then repairs the solution, which updates the current best solution if the new solution is lexicographically better. If not, ALNS invokes $r(d^2(S,RT))$ (lines 10–13) and destroys the current solution based on the machine based destroy operator and repair with the hope of finding a better solution with fewer machines (β) used. If all jobs are assigned (line 14), the algorithm invokes $r(d^2(S,RT))$ (Lines 15–19) to

minimize β . Once all job and machine-based destroy operators are tried, if there is still time remaining, ALNS calls a random destroy operator and repairs the destroyed solution $r(d^3(S,RT))$ (lines 20–23). Once the time limit is reached, ALNS stops (line 23) returning the current best solution (line 25). Details of Algorithm 1 are explained in the following subsections.

4.1.1 Finding an initial solution

Two methods have been employed to find an initial solution as shown in Algorithm 2. The first method uses the lexicographic solution scheme presented in Sect. 3.2 with limited time. And the second one is using the matheuristic method to search for larger solution space in a given limited time. These initial solution algorithms are selected with binary “Heuristic” parameters in line 2 of Algorithm 1.

Algorithm 2 InitialSol(machines,jobs,Heuristic)

```

if Heuristic=1 then
  S  $\leftarrow$  Lex(machines,jobs)
else
  S  $\leftarrow$  Matheuristic(machines,jobs)
end if
Return S

```

Algorithm 3 is used to find the first feasible solution using the lexicographic scheme in Sec 3.1 which is denoted as *solve* method. Since the problem is NP-Hard, time restriction is used to find the first feasible solution in a reasonable time.

Algorithm 3 Lex(machines,jobs,RT)

```

S  $\leftarrow$   $\emptyset$ 
start=current_time
if RT>0 or FirstSolTim>0 then
  S  $\leftarrow$  solve(machines,jobs, $\emptyset$ , $\emptyset$ , $\emptyset$ )
  end=current_time
  RT=timlim-(end-start)
  FirstSolTim=timlim-(end-start)
end if
Return S

```

In Algorithm 4, a matheuristic is applied to find an initial solution. First, jobs are divided into Job_groups using StaticJobGroup algorithm, Algorithm 5. For each group in Job_groups, g is a subproblem for matheuristic and solved by the lexicographic

method presented in Sect. 3.1 with resulting objective function values α_g and β_g . Overall objective function values α and β are updated with values from those groups. And finally, the assignments in matheuristic are returned to Algorithm 1.

Algorithm 4 Matheuristic(machines,jobs)

```

job_groups=StaticJobGroups(machines,jobs)
 $\alpha = 0, \beta = 0$ 
 $S \leftarrow \emptyset$ 
for  $g$  in job_groups do
   $\alpha_g, \beta_g \leftarrow \text{solve}(\text{machines}, \text{jobs}, \emptyset, \emptyset, \emptyset)$ 
   $\alpha+ = \alpha_g$ 
   $\beta+ = \beta_g$ 
end for
 $S \leftarrow S(\alpha, \beta)$ 
Return  $S$ 

```

Algorithm 5, groups jobs according to their features such as flotte, recipe, and colour percentage. Each group is appended to job_groups as a set for use in Algorithm 4. Note that Algorithm 5 is called static because jobs are grouped based on physical attributes and also to distinguish it from a dynamic grouping scheme during the search to be explained in the following subsection.

Algorithm 5 StaticJobGroups(machines,jobs)

```

job_groups  $\leftarrow \emptyset$ 
for  $j$  in jobs do
   $g \leftarrow \emptyset$  ▷ create new group  $g$ 
   $g \leftarrow g \cup j$ 
  jobs  $\leftarrow \{\text{jobs}\} - j$ 
  for  $k$  in jobs do
    if  $\text{recepte}_j = \text{recepte}_k \ \& \ \text{flotte}_j = \text{flotte}_k \ \& \ \text{perc}_j = \text{perc}_k$  then
       $g \leftarrow g \cup \{k\}$ 
      jobs  $\leftarrow \text{jobs} - \{k\}$ 
    end if
  end for
  job_groups  $\leftarrow \text{job\_groups} \cup \{g\}$ 
end for
Return job_groups

```

4.1.2 Destroy and repair operators

In this subsection, we explain the details of the job (Algorithm 6 and machine-based (Algorithm 8) and random (Algorithm 10) destroy and repair methods mentioned in Fig. 5.

Job-based destroy and repair operator presented in Algorithm 6 starts with initializing parameters in lines 1–3. DynamicJobGroup algorithm is called (Algorithm 7) in line 4 to create two lists of job groups: assigned and unassigned. While assigned groups indicate a group of jobs assigned to a machine, unassigned jobs groups refer to job groups where at least one job is not assigned to any machine. Then algorithm 6 goes through each assigned group in descending order of weight and chooses the one, lines 5–6, and creates two sets of assignment variables, V^r and V^f , ones to be relaxed and fixed respectively, line 7. To escape from local optima, all assignment variables in group a and the first unassigned groups are relaxed, line 8, while the rest of the variables are kept as they are, line 9. Then, the lexicographic method presented in Sect. 3.1 is executed with the current solution S , and a set of relaxed and fixed variables to find a new temporary solution S^t , line 10. As discussed in the numerical experiments section, the subproblem optimized in this stage is small enough to find optimal solutions in all instances. Suppose the temporary solution is better than the current best solution (line 11), the best solution is updated (line 12), a set of assigned and unassigned groups of jobs are re-created based on the new best solution (line 13), and the algorithm starts from scratch (line 14). If not, the algorithm continues by switching to the next group of jobs in the unassigned group (line 18). When all unassigned groups are traversed without further improvement, the algorithm returns the current best solution and remaining time parameter to Algorithm 1. The worst-case number of calls for Algorithm 6 will be $O(IJ)$ if each individual job indicates an unassigned group itself.

Algorithm 6 $r(d^1(machines, jobs, S, RT))$

```

1: count=0
2: start=current_time
3: job_groups=StaticJobgroups(machines,jobs)
4: unassigned_groups, assigned_groups=DynamicJobGroups(machines,jobs,job_groups,S)
5: while count < |unassigned_groups| do
6:   for a in assigned_group do
7:      $V^r \leftarrow \emptyset, V^f \leftarrow \emptyset$            ▷ Initializing set of relax and fix variables
8:      $V^r \leftarrow$  set of variables in a and unassigned_groups[count] to be relaxed
9:      $V^f \leftarrow$  set of remaining variables to be fixed
10:     $S^t \leftarrow solve(machines, jobs, S, V^f, V^r)$ 
11:    if  $S^t \prec S^b$  then
12:       $S^b \leftarrow S^t$ 
13:      unassigned_groups, assigned_groups=DynamicJobGroups(machines,jobs,job_groups,S)
14:      count = 0
15:      break
16:    end if
17:  end for
18:  count+=1
19: end while
20: end = current_time
21: RT=time_limit-(end-start)
22: Return  $S^b, RT$ 

```

Since `DynamicJobGroups` referred to in lines 4 and 13 of Algorithm 6 are also being initialized in other destroy and repair operators, we provide its details below. Algorithm 7 starts with initializing two lists of groups of jobs, for unassigned and assigned groups in lines 1 and 2, and then invokes Algorithm 5 to group jobs according to their physical attributes in line 3. Then, each job group is checked (lines 4 and 5) and added to the list of unassigned (line 6) groups if there exists at least one job not assigned to any machine or assigned list of groups otherwise (line 8).

Algorithm 7 *DynamicJobGroups(machines,jobs,job_groups,S)*

```

1: unassigned_groups ← ∅
2: assigned_groups ← ∅
3: job_groups=StaticJobGroups(machines,jobs)
4: for j in job_groups do
5:   if There exist at least one job in j not assigned to any machine then
6:     unassigned_groups.append(j)
7:   else
8:     assigned_groups.append(j)
9:   end if
10: end for
11: Return unassigned_groups, assigned_groups

```

The second destroy and repair operator is machine based and explained in Algorithm 8, prioritizing the second objective, minimizing the number of machines used. The Algorithm starts with initializing input parameters in lines 1 and 2. In line 3, Algorithm 9 is invoked to create the list of used machines in the current based solution which is ordered in ascending (`Reverse=1`) or descending (`Reverse=0`) order based on the weight of jobs assigned to each. Starting from line 4, the algorithm traverses each machine in the used machine list. It initializes the list of variables to be relaxed and fixed, line 6, and the group of jobs assigned to the selected machine, line 7. Then, the algorithm checks all job groups, line 8, and add all of them to the list of Machine group if at least one job is assigned to the selected machine, lines 9 and 10. Job groups in `assigned_groups` are in descending order based on their weight and Job groups in `unassigned_groups` are sorted in descending order in terms of the number of unassigned jobs. All assignment variables related to the machine group are added to the list of variables to be relaxed and the rest are kept as fixed, lines 13 and 14. Then, the algorithm invokes the lexicographic solution scheme in Sect. 3.2 by considering the relaxed and fixed variables in line 15. If the current solution is better than the best solution in line 16, the best solution is updated in line 17, the list of used machines is initialized in line 18, and the algorithm starts from the beginning, line 19. Otherwise, the algorithm continues with the following machine in the list of used machines, line 21. Algorithm 8 terminates by returning the best solution and the remaining runtime, line 26.

Algorithm 8 $r(d^2(S, RT, Reverse))$

```

1: start = current_time
2: job_groups = StaticJobGroup(machines, jobs)
3: used_machine = CreateMachineGroup(S, Reverse)
4: u = 0
5: while  $u < |used\_machine|$  do
6:    $V^r \leftarrow \emptyset, V^f \leftarrow \emptyset$   $\triangleright$  Initialization set of variables to be relaxed and fixed
7:   MachineGroup  $\leftarrow \emptyset$ 
8:   for g in job_groups do
9:     if There exist at least one job in g assigned used_machine[u] then
10:      MachineGroup  $\leftarrow$  MachineGroup  $\cup$  g
11:    end if
12:  end for
13:   $V^r \leftarrow$  MachineGroup  $\triangleright$  Set of all variables to be relaxed
14:   $V^f \leftarrow$  set of all remaining assignment variables to be fixed
15:   $S^t \leftarrow$  solve(machines, jobs, S,  $V^f$ ,  $V^r$ )
16:  if  $S^t \prec S^b$  then
17:     $S^b \leftarrow S^t$ 
18:    used_machine = CreateMachineGroup(S, Reverse)
19:    u = 0
20:  else
21:    u + = 1
22:  end if
23: end while
24: end = current_time
25: RT = time.limit - (end - start)
26: Return  $S^b, RT$ 

```

In the following, Algorithm 9, we demonstrate how machine groups are created, which are used in Algorithm 8. Algorithm 9 traverses each machine, line 2, and if at least one job is assigned, the total weight of jobs is computed, and the machine is added to the list of used machines, lines 3–5. The list of used machines is sorted in descending (*Reverse* = 0) or ascending (*Reverse* = 1) order based on the total weight of assigned jobs before it is returned to Algorithm 8, lines 8 and 9.

Algorithm 9 CreateMachineGroup(*S*, *Reverse*)

```

1: used_machine  $\leftarrow \emptyset$ 
2: for i in machines do
3:   if any job assigned to i then
4:     c  $\leftarrow$  Calculate weight of machine i
5:     used_machine.append((i, c))
6:   end if
7: end for
8: used_machine.sort(c, Reverse)
9: Return used_machine

```

Finally, after job based and machine-based destroy and repair operators explained above, if there is still available time, Algorithm 1 uses a completely random destroy

and repair operator, Algorithm 10 as explained below. After initializing the set of relaxed and fixed variables, in line 1, randomly selected $b\%$ of assignment variables (chosen as 30%) are fixed, and $(1-b)\%$ are relaxed, lines 2 and 3. The lexicographic solution scheme is invoked with the given fixed and relaxed variables, line 4 and if the new solution is better than the current best, the incumbent solution is updated, lines 5–7.

Algorithm 10 $r(d^3((S,b,RT)))$

```

1:  $V^r \leftarrow \emptyset, V^f \leftarrow \emptyset$  ▷ Initialization set of relax and fix variables
2:  $V^r \leftarrow$  set of  $\%(100-b)$  of variables to be relaxed
3:  $V^f \leftarrow$  set of  $\% b$  of variables to be fixed
4:  $S^t \leftarrow$  solve(machines,jobs,S, $V^f, V^r$ )
5: if  $S^t \prec S^b$  then
6:    $S^b \leftarrow S^t$ 
7: end if
8:  $end = current\_time$ 
9:  $RT = time\_limit - (end - start)$ 
10: Return  $S^b, RT$ 

```

In the following subsection, we provide a numerical example to demonstrate the execution of Algorithm 1.

4.2 A numerical example

This paper proposes an ALNS with three different destroy and repair operators to tackle multi-objective YDP. Three variants of ALNS are proposed. This subsection presents a minimal example of how these operators are executed. We generate an instance with three dyeing machines (DM) and 17 jobs, as shown in Tables 1 and 2.

Table 1 DM data for the minimal example

Dye machine ($machine_i$)	Level ($level_{s_k}$)	Volume (Lt.) ($volumacapacity_m$)	Max spool ($maxquantity_m$)	Min Kg. ($minkg_m$)	Max Kg. ($maxkg_m$)
DMS01	1	250	20	0	17
DMS01	2	300	25	0	20
DMS01	3	350	30	0	25
KRN03	1	320	30	0	25
KRN03	2	400	48	30	40
THS01	1	1600	180	123	149
THS01	2	1800	210	150	174
THS01	3	2100	270	175	224
THS01	4	2400	300	225	249

Table 2 Job data for the minimal example

Job ($ordernoi_t$)	Weight (kg_t)	Recipe ($orderrecipe_i$)	Colour (%) ($ordercolour_i$)	Reactive ($reactive_i$)	Lyc (lyc_t)	Min flote ($minflotte_i$)	Max flote ($maxflotte_i$)
1	10	30,609,630	0.197	0	0	15	16
2	8	30,609,630	0.197	0	0	15	16
3	5	30,609,630	0.197	0	0	15	16
4	26	50,612,562	0.4883	0	0	9	11
5	24	50,612,562	0.4883	0	0	9	11
6	16	50,612,562	0.4883	0	0	9	11
7	15	50,612,562	0.4883	0	0	9	11
8	12	50,612,562	0.4883	0	0	9	11
9	10	50,612,562	0.4883	0	0	9	11
10	13	50,513,411	0.72	0	0	9	11
11	7	50,513,411	0.72	0	0	9	11
12	5	50,513,411	0.72	0	0	9	11
13	15	50,513,411	0.72	0	0	9	11
14	70	30,513,496	1.22	0	0	9	11
15	50	30,513,496	1.22	0	0	9	11
16	15	30,513,496	1.22	0	0	9	11
17	30	30,513,496	1.22	0	0	9	11

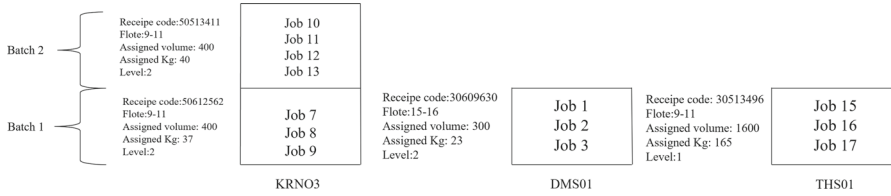


Fig. 6 Initial solution for the minimal data

Overall ALNS algorithm, Algorithm 1 starts with an initial solution, in which, one of the variants of finding an initial solution, Algorithm 4 requires the classification of jobs first. Algorithm 5, jobs are classified into four groups: group 1={1,2,3}, group 2={4,5,6,7,8,9}, group 3={10,11,12,13}, group 4={14,15,16,17}. An initial solution is shown in Fig. 6

In the initial solution, there are four unassigned jobs: jobs 4,5,6, and 14. Then ALNS invokes Algorithm 6 which first creates two groups of jobs based on the current solution. Suppose all jobs in a group previously created in Algorithm 5 are assigned to a machine. In that case, that group is put into the assigned_group set, otherwise into the unassigned_group set(see. Algorithm 7). For this demonstrated example, assigned_group: {[10, 11, 12, 13], [1, 2, 3]} which is ranked in descending order based on their total weight; and unassigned_group: {[4, 5, 6, 7, 8, 9], [14, 15, 16, 17]} which is sorted in descending order based on the number of unassigned jobs in each group. Next, Algorithm 6 relaxes all variables in the first elements of both sets and invokes the lexicographic solver by fixing the assignment variables for the rest as fixed. The new solution found is illustrated in Fig. 7.

Then, in the next iteration of Algorithm 6, two more jobs are assigned; jobs 5 and 6. However, since job 4 from group 2 and 14 from group 4 are still not assigned to any machine, the assigned and unassigned set of groups of jobs stays the same::unassigned_group: {[4, 5, 6, 7, 8, 9], [14, 15, 16, 17]}; assigned_group: {[10, 11, 12, 13], [1, 2, 3]}. However, since the total number of unassigned jobs is reduced by two with the new solution, the best solution is updated and Algorithm 6 starts from the beginning. The next step will be the same as the previous one: all variables of the first element of both sets are relaxed: {[4, 5, 6, 7, 8, 9], [10, 11, 12, 13]} which results

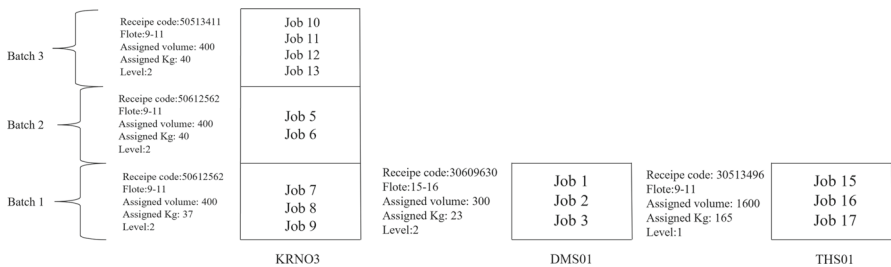


Fig. 7 Demonstration of the improved solution by Algorithm 6 after the first iteration

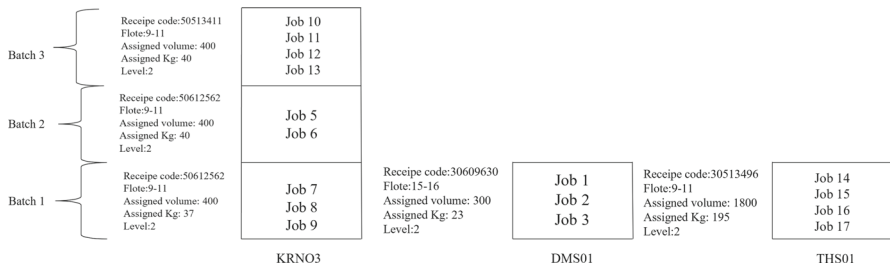


Fig. 8 Final result of Algorithm 6

in no improvement. Then the next component of the assigned_group and the first element in the unassigned_group. ($\{[4, 5, 6, 7, 8, 9], [1, 2, 3]\}$) are chosen without no improvement as well. At this point, because all possible pairs of unassigned/assigned groups of jobs are tried with the first element in the unassigned_group has been tried, Algorithm 6 proceeds with pairing the next element in the unassigned_group which is $[14, 15, 16, 17]$ and the first element in the assigned_group $[10, 11, 12, 13]$ to create a set of variables to relax (create a subproblem). The solution of this new subproblem results in an improvement by assigning one more job, job 14, as demonstrated in Fig. 8.

As the algorithm achieves a better solution, a set of assigned and unassigned jobs are updated, unassigned_job: $\{[4, 5, 6, 7, 8, 9]\}$, assigned_job: $\{[14, 15, 16, 17], [10, 11, 12, 13], [1, 2, 3]\}$ Algorithm 1 continues with Algorithm 6 till trying all pair of assigned/unassigned groups of jobs as long as the time limit is not exceeded. In this small example, we do not see further improvement with the Algorithm 6 operator and as there is still time to search, ALNS continues with Algorithm 8. Algorithm 8 first creates used machine sets. In this illustrative example, three machines are used which are DMS01, KRNO3, and THS01. As an input parameter, the algorithm sorts them based on their weight in descending or ascending order (see. Alg. 1). In this example, descending order is used which orders machines as THS01, KRNO3, and DMS01, respectively. If at least one job exists in any job group assigned to THS01, then all variables of jobs in that group are relaxed. In that case, the assignment of jobs in group 4, job 14–17, is relaxed. However, there will be no improvement since all jobs in this group are already assigned. Next, machine KRNO3 is selected to which Group 2 and Group 3 are assigned. Hence, all variables of all jobs in groups 2 and 3 are to be relaxed the resulting subproblem is solved again. If there will be no more improvement after Algorithm 8 and there is still solution time available, ALNS will continue with Algorithm 10 which randomly chooses the set of jobs to be fixed and relaxed. In this study, 30% of the randomly selected job assignments are kept fixed in every iteration. Algorithm 10 continues until the stopping criteria are met. In the next section, computational experiments to test the proposed ALNS algorithm over a vast number of realistically generated instances are presented.

5 Computational result

In this section, we test the performance of developed ALNS and compare it with the best results obtained from the solution of the lexicographic MIP scheme over extensive instances generated based on a real industrial case.

The industrial dyeing system consists of 33 dyes (DM) ($|K| = 33$) where 26 of them are eligible for reactive and lycra job orders $Special_k = 1$ and none of them is ready for any of the colour percentages ($initialcolour_k = 0$). Each DM has at least one, at most, four different dyeing levels based on its features. Volume capacity, minimum and maximum weights, and the number of spools allowed for a DM in a defined level vary between the lower and upper bounds as shown in Table 3. Similarly, there are four different and predefined flotte intervals as given in Table 4. The Spool weight, *spool* is assumed as 0.850 kg.

For numerical experiments, 45 instances are generated by varying the number of jobs from 15 to 100 and the number of machines is fixed to 33 in each as in the real industrial case. For each number of jobs (5 different numbers of jobs: 15, 40, 60, 80, and 100), nine instances are generated by changing flotte values and colors which in total results $5 \times 9 = 45$ instances. Instances can be accessed from corresponding author’s GitHub repository (Duran). The instances utilized in this study were derived from actual data obtained from a textile company. Due to privacy concerns and confidentiality agreements, the specific real data and company identity cannot be disclosed. Table 5 shows the number of different flotte values and colors for each five groups.

The ALNS algorithm and the lexicographic MIP scheme in Sect. 3.2 are implemented in Python 3.7 (Foundation 2023) and executed on an ASUS ZenBook 13 computer with Intel core i7 8th gen processor, 16 GB RAM running on Windows 10 operating system. IBM ILOG CPLEX (IBM 2023) is used as a solver for MIP models in Sect. 3.2 and to repair the destroyed subproblems in the ALNS algorithm.

Table 3 Lower and upper bounds of dye machine parameters

Parameter	Lower bound	Upper bound
<i>volumcapacity_j</i> (litres)	14	4800
<i>maxnumber_j</i> (spools)	2	600
<i>minkg_j</i> (kg)	0	450
<i>maxkg_j</i> (kg)	1.7	500

Table 4 Predefined flotte intervals of batches and jobs

Flotte $f \in F = \{1, \dots, F \}$	1	2	3	4
<i>minflotte_f</i>	9	11	15	17
<i>maxflotte_f</i>	11	12	16	20

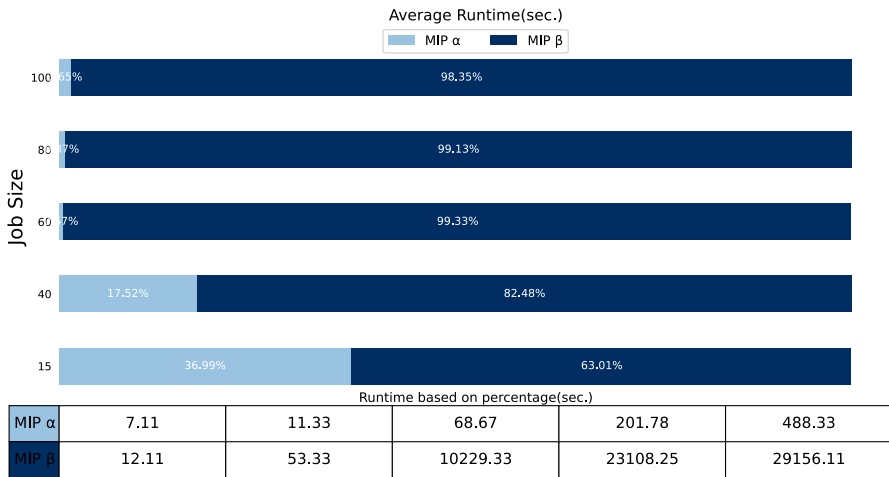


Fig. 9 Runtime of instances with lexicographic scheme in Sect. 3.2

Table 5 Number of different flotte and colour percentage for each group

# of different flotte	# of different colour
1	3
2	3
3	3
1	7
2	7
3	7
1	10
2	10
3	10

First, we present the performance of the lexicographic solution scheme in Sect. 3.2 on the generated instances that are limited to 10 h of runtime. Figure 9 shows what percentage of the overall solution time is used for solving MIP α and MIP β respectively. Results show that MIP α consumes less time in larger instances. This behaviour is because, in larger instances, the set of feasible assignments is limited and hence. In contrast, the assignment problem (MIP α) is solved in less time and the remaining time is used to find a solution with less machine usage (MIP β). The individual runtime of each instance will be provided in the following results table. Among the factors that affect the runtime, the number of jobs is prominent. However, increasing the number of colors and flotte has not been found to establish a strong correlation with the runtime.

Table 6 Results for small instances

J*F*C	Instances	Reverse	Initial solution methods						Best lower bound		
			Lex			Matheuristic			α	β	Time
			α	β	Time	α	β	Time			
15*1*3	1	0	0	3	1,92	0	3	2,83	0	3	17
		1	0	3	1,92	0	3	2,83			
15*2*3	2	0	1	4	1,83	1	4	2,76	1	4	27
		1	1	4	1,83	1	4	2,76			
15*3*3	3	0	2	4	1,69	2	4	2,71	2	4	20
		1	2	4	1,69	2	4	2,71			
15*1*7	4	0	8	4	1,39	8	4	5,40	8	4	17
		1	8	4	1,39	8	4	5,40			
15*2*7	5	0	5	5	1,33	5	5	5,38	5	5	18
		1	5	5	1,33	5	5	5,38			
15*3*7	6	0	7	3	1,37	7	3	7,04	7	3	17
		1	7	3	1,37	7	3	7,04			
15*1*10	7	0	5	4	1,35	5	4	7,84	5	4	19
		1	5	4	1,35	5	4	7,84			
15*2*10	8	0	7	3	1,32	7	3	8,78	7	3	20
		1	7	3	1,32	7	3	8,78			
15*3*10	9	0	11	1	1,34	11	1	7,88	11	1	18
		1	11	1	1,34	11	1	7,88			

45 instances mentioned earlier are divided into small (15 jobs), medium (40 and 60 jobs,) and large (80 to 100 jobs) to analyze the behaviour of ALNS and lexicographic solution method better which are presented in In Tables 6, 7, and 8.

The first column in each table J*F*C refers to (Job size)*(number of different Flotte)*(number of different Colour) of each instance respectively. The next column, “Instance” refers to the instance number which is followed by the “Reverse” column “0” indicates machines are sorted in Descending (Reverse =0) order or not, “1” in Algorithm 8 based on the total weight of jobs assigned to them. The following column demonstrates the result of the ALNS algorithm, Algorithm 1, based on two initial solution methods described in Sect. 4.1.1 with respective objective function values α , the number of unassigned jobs, β , the number of machines used with the time those solutions found within the 1800 s of runtime. Finally, the “Best lower bound” column represents the solution obtained from executing the lexicographic scheme in Sect. 3.2. For each instance, the lexicographic scheme is run for 11 h of a time limit, and the best solution obtained is reported. Please note that the results discussed in Fig. 9 use the runtime performance of the lexicographic scheme presented in this column. If the solution obtained from the lexicographic scheme is reported as optimal, they are shown as bold and italic. If not, the best resulting solution (ALNS or lexicographic

Table 7 Results for medium instances

J*F*C	Instances	Reverse	Initial solution methods						Best lower bound		
			Lex			Matheuristic			α	β	Time
			α	β	Time	α	β	Time			
40*1*3	10	0	0	3	56, 46	0	3	5, 74	0	3	62
		1	0	3	59, 30	0	3	5, 74			
40*2*3	11	0	0	3	15, 61	0	3	6, 16	0	3	101
		1	0	3	15, 61	0	3	6, 16			
40*3*3	12	0	7	3	34, 72	7	5	6, 53	7	3	31
		1	7	3	25, 97	7	5	6, 53			
40*1*7	13	0	0	4	7, 71	0	4	7, 83	0	4	31
		1	0	4	7, 71	0	4	7, 83			
40*2*7	14	0	1	4	21, 54	1	4	7, 69	1	3	81
		1	1	4	22, 73	1	4	7, 69			
40*3*7	15	0	6	4	5, 55	6	4	50, 70	6	4	26
		1	6	4	5, 55	6	4	50, 90			
40*1*10	16	0	4	6	42, 06	4	7	10, 13	4	6	202
		1	4	6	44, 39	4	7	10, 13			
40*2*10	17	0	1	6	28, 07	1	6	10, 14	1	6	29
		1	1	6	10, 96	1	6	10, 14			
40*3*10	18	0	8	8	3, 73	8	9	9, 99	8	8	19
		1	8	8	3, 73	8	9	9, 99			
60*1*3	19	0	0	3	20, 38	0	3	8, 42	0	3	241
		1	0	3	20, 38	0	3	8, 42			
60*2*3	20	0	0	12	31, 81	0	3	10, 70	0	3	174
		1	0	9	31, 65	0	3	10, 70			

Table 7 (continued)

J*F*C	Instances	Reverse	Initial solution methods						Best lower bound		
			Lex			Matheuristic			α	β	Time
			α	β	Time	α	β	Time			
60*3*3	21	0	0	5	84, 10	0	5	8, 16	<i>0</i>	<i>5</i>	<i>2008</i>
		1	0	5	79, 24	0	5	8, 16			
60*1*7	22	0	0	7	43, 47	0	7	10, 23	<i>0</i>	<i>5</i>	<i>1692</i>
		1	0	7	43, 47	0	7	10, 23			
60*2*7	23	0	0	9	81, 19	0	5	10, 35	<i>0</i>	<i>5</i>	<i>36105</i>
		1	0	8	85, 20	0	5	10, 35			
60*3*7	24	0	10	7	27, 90	10	6	9, 73	<i>10</i>	<i>6</i>	<i>5101</i>
		1	10	7	28, 15	10	6	9, 73			
60*1*10	25	0	0	6	68, 20	0	5	24, 48	<i>0</i>	<i>5</i>	<i>1159</i>
		1	0	6	54, 92	0	5	53, 12			
60*2*10	26	0	0	8	31, 01	0	8	18, 06	0	6	36089
		1	0	8	39, 12	0	8	64, 18			
60*3*10	27	0	6	7	133, 80	6	7	12, 12	6	6	10113
		1	6	7	137, 04	6	7	12, 12			

scheme) is shown as bold for each instance. The criteria for ranking a solution as the best is first, lexicographically α and β values, and if they are the same then the time that solution is reached. Note that, if lexicographically scheme is reported optimal, it is shown as italic.

At first sight, Table 6 shows that the ALNS algorithm with an initial lexicographic solution is the most efficient method to solve small instances regardless of the ordering heuristic used in the “Reverse” column. However, starting from the medium size instances, Table 7 and Table 8 indicate better performance for the ALNS algorithm with a Matheuristic based initial solution regardless of the ordering heuristic. However, when all instances are analyzed, the ALNS algorithm with a matheuristic based initial solution found the best lower bound in 27 instances out of 45. It is also concluded that

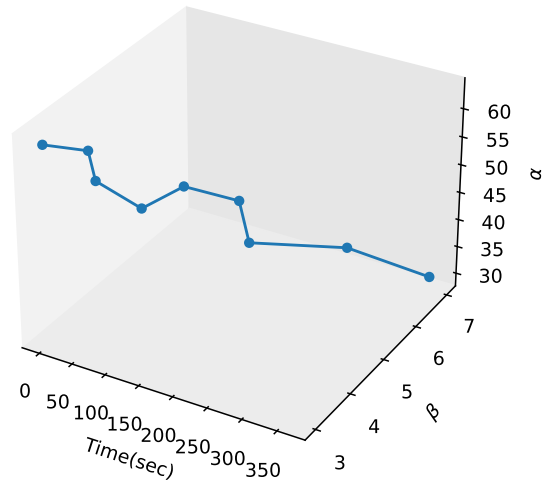
Table 8 Results for large instances

J*F*C	Instances	Reverse	Initial solution methods				Best lower bound				
			Lex		Matheuristic		α		β		Time
			α	β	Time	α	β	Time	α	β	
80*1*3	28	0	0	2	96, 40	0	3	12, 59	0	2	257
80*2*3	29	1	0	2	51, 26	0	3	12, 59	0	2	656
		0	0	5	100, 83	0	3	13, 47			
80*3*3	30	1	0	8	87, 75	0	3	13, 47	11	4	6131
		0	11	4	148, 83	11	4	12, 42			
80*1*7	31	1	11	5	89, 65	11	4	12, 42	0	4	34519
		0	0	6	158, 179	0	5	12, 68			
80*2*7	32	1	0	5	134, 53	0	5	12, 68	2	6	36159
		0	2	6	716, 75	2	6	13, 30			
80*3*7	33	1	2	6	721, 65	2	6	13, 30	6	6	36318
		0	6	7	582, 25	7	6	11, 96			
80*1*10	34	1	6	7	641, 13	7	6	11, 96	0	6	36086
		0	0	9	1711, 97	0	6	14, 86			
80*2*10	35	1	0	9	1699, 34	0	6	14, 86	0	5	36300
		0	2	6	571, 27	0	7	48, 22			
80*3*10	36	1	2	6	562, 37	0	7	31, 84	7	8	36256
		0	7	9	1231, 87	8	10	14, 47			
100*1*3	37	1	7	9	1289, 46	8	10	14, 47	0	4	824
		0	3	6	125, 35	0	5	128, 62			

Table 8 (continued)

J*F*C	Instances	Reverse	Initial solution methods						Best lower bound					
			Lex			Mathuristic								
			α	β	Time	α	β	Time	α	β	Time	α	β	Time
100*2*3	38	1	7	4	124, 70	0	5	89, 66	0	5	192, 65	0	5	10082
100*3*3	39	1	0	5	231, 19	0	5	253, 20	0	5	106, 59	17	7	36133
100*1*7	40	1	17	7	291, 61	17	7	310, 46	17	7	14, 30	0	5	36757
100*2*7	41	0	0	6	173, 16	0	6	151, 07	0	6	28, 95	0	6	36677
100*3*7	42	1	0	12	174, 94	0	8	165, 67	0	8	17, 02	19	6	36782
100*1*10	43	1	19	6	449, 21	19	6	444, 47	19	6	14, 89	0	6	36561
100*2*10	44	0	2	15	152, 63	0	8	150, 38	0	8	17, 92	1	7	36455
100*3*10	45	1	1	10	305, 21	1	10	340, 26	1	10	17, 02	18	6	36529
		1	30	7	405, 87	20	8	561, 92	20	8	17, 45			
		1	21	6										

Fig. 10 ALNS search process for Instance 45 with 100 jobs, 3 different flotte, and 10 colours



ALNS, after matheuristic, found the best lower bound in 42 instances for objective function 1, indicating the success of Algorithm 6. Overall, experiments show the ALNS algorithm's superiority in finding near-optimal solutions quickly.

Finally in Fig. 10, we demonstrate the ALNS search process for instance 45 with 100 jobs, three different flottes, and ten different colors. Figure 10 shows that during the search for the first objective function, the number of unassigned jobs, α , went down from 61 to 21 with the expense of increasing the second objective function, β , from 3 to 6 within 8 iterations.

6 Conclusions

This study is motivated by an industrial problem; multi-objective yarn dying planning. The dyeing process is vital for the overall effectiveness of an integrated textile manufacturing system because of the high setup and changeover costs/times as well the environmental impact of the process. The amount of chemicals needed to wash the DMs is reduced by using fewer of them and ordering the colors from light to dark, reducing environmental pollution. Additionally, several technological limitations and competing goals require the attention and time of production planners. To the best of the authors' knowledge, no similar research exists in the literature on formulating and solving this very practical yet computationally challenging problem.

First, this paper formulates the problem as a multi-objective mixed integer programming model. And second, an adaptive large neighbourhood search method with various initial solution methods along with alternative destroy and repair algorithms developed for large instances of the problem.

Developed mathematical formulation and ALNS method are tested with an extensive number of instances adapted from industrial data. Experimental results show that the runtime of the mathematical model grows exponentially as the instance size grows.

However, numerical computations reveal that the developed ALNS method finds optimal and near-optimal solutions for all sizes of instances in a very short computational time of fewer than 3 min at most. This is particularly important for manufacturing engineers who spend several hours finding a reasonably efficient production plan.

This work can be extended further in terms of problem definition and solution methods. The developed ALNS algorithm uses mathematical modeling-based initial solution methods that require employing a commercial or open-source solver. Developing a solver-free method for an initial solution is undoubtedly a significant extension of the ALNS method proposed in this paper. As part of another future work, the proposed model can be compared to other promising metaheuristics and matheuristics methods such as Adaptive Polyploid Memetic Algorithm (APMA) (Dulebenets 2021) and customized multi-objective hybrid metaheuristic solution algorithm (Pasha et al. 2022). Ant Colony Optimization (ACO) based methods can be considered as well (Singh and Pillay 2022) which utilizing hyper-heuristic ant colony optimisation (HACO) could be promising. Apart from applying different metaheuristics, in this article, the lexicographic method has been employed as the solution approach; however, other multi-objective methods such as the Pareto frontier (Elahi et al. 2022; Yıldırım et al. 2019) or weighted sum (Ozturk et al. 2016) can also be utilized depending on the problem definition. Exploiting machine learning methods in textile process planning showed good initial results (He et al. 2020) and can be a promising future research direction for the problem studied in this paper, especially the stochastic extension. The learning automaton (LA) (Zhao and Zhang 2020) customizes the search strategy and can be used to improve the performance of ALNS in this paper. Furthermore, stochastic batch processing times and preventive maintenance activities can be added as a problem extension as in Gholizadeh et al. (2021). Machine breakdowns, order cancellation are some other industrial extensions of the problem studied as in He et al. (2022) and Zhang et al. (2021). Adding due date constraints, limiting the earliness and tardiness to avoid inventory holding and lateness costs as a third objective function, and considering sequence-dependent setup times from one color to another in machines might give more practical use of the methods developed in this paper.

Funding Open Access funding provided by the IReL Consortium. This work was conducted with the financial support of the Science Foundation Ireland Centre for Research Training in Artificial Intelligence under Grant No. 18/CRT/6223 and Grant number 22/NCF/DR/11264, National Challenge Fund, Digital for Resilience Challenge.

Data availability The data that support the findings of this study is publicly available at Duran (2023).

Declarations

Conflict of interest The authors report there are no conflict of interest to declare.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission

directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Arora JS (2012) Chapter 17 - multi-objective optimum design concepts and methods. In: Arora JS (ed) Introduction to Optimum Design, 3rd edn. Academic Press, Boston, pp 657–679
- Crauwels H, Beullens P, Van Oudheusden D (2006) Parallel machine scheduling by family batching with sequence-independent set-up times. *Int J Op Res* 3(2):144–154
- Delorme M, Iori M, Martello S (2016) Bin packing and cutting stock problems: Mathematical models and exact algorithms. *Eur J Op Res* 255(1):1–20
- Demir Y, Kemal İ (2022) A novel approach for optimum planning of bobbin boilers in textile industry. *Text Appar* 32(1):24–36
- Dulebenets MA (2021) An adaptive polyploid memetic algorithm for scheduling trucks at a cross-docking terminal. *Inf Sci* 565:390–421
- Duran E Yarn Dyeing Planning Instances. <https://github.com/egedurann/batching-problem>
- Elahi I, Ali H, Asif M, Iqbal K, Ghadi Y, Alabdulkreem E (2022) An evolutionary algorithm for multi-objective optimization of freshwater consumption in textile dyeing industry. *PeerJ Comput Sci* 8:932
- Eroğlu DY, Özmutlu HC, Köksal SA (2014) A genetic algorithm for the unrelated parallel machine scheduling problem with job splitting and sequence-dependent setup times-loom scheduling. *Text Appar* 24(1):66–73
- Mart, R., Pardalos, P. M., & Resende, M. G. (2018). *Handbook of heuristics*. Springer Publishing Company, Incorporated.
- Fleszar K, Hindi KS (2018) Algorithms for the unrelated parallel machine scheduling problem with a resource constraint. *Eur J Op Res* 271(3):839–848
- Foundation PS Python.org. <https://www.python.org/> Accessed on June 3rd, 2023
- Gholizadeh, H., Fazlollahabtabar, H., Fathollahi-Fard, A. M., & Dulebenets, M. A. (2021). Preventive maintenance for the flexible flowshop scheduling under uncertainty: a waste-to-energy system. *Environmental science and pollution research international*. Advance online publication. <https://doi.org/10.1007/s11356-021-16234-x>
- Ghosh JB, Gupta JN (1997) Batch scheduling to minimize maximum lateness. *Op Res Lett* 21(2):77–80
- Gomes UT, Pinheiro PR, Saraiva RD (2021) Dye schedule optimization: A case study in a textile industry. *Appl Sci* 11(14):6467
- He Z, Tran KP, Thomassey S, Zeng X, Xu J, Yi C (2022) Multi-objective optimization of the textile manufacturing process using deep-q-network based multi-agent reinforcement learning. *J Manuf Syst* 62:939–949
- He Z, Tran KP, Thomassey S, Zeng X, Xu J, Yi C (2020) Multi-Objective Optimization of the Textile Manufacturing Process Using Deep-Q-Network Based Multi-Agent Reinforcement Learning. *J Manuf Syst* 62:939–949
- Hsu H-M, Hsiung Y, Chen Y-Z, Wu M-C (2009) A ga methodology for the scheduling of yarn-dyed textile production. *Expert Syst Appl* 36(10):12095–12103
- Huynh N-T, Chien C-F (2018) A hybrid multi-subpopulation genetic algorithm for textile batch dyeing scheduling and an empirical study. *Comput & Ind Eng* 125:615–627
- Hynes NRJ, Kumar JS, Kamyab H, Sujana JAJ, Al-Khashman OA, Kuslu Y, Ene A, Kumar BS (2020) Modern enabling techniques and adsorbents based dye removal with sustainability concerns in textile industrial sector-a comprehensive review. *J Clean Prod* 272:122636
- IBM: IBM ILOG CPLEX Optimization Studio. <https://www.ibm.com/products/ilog-cplex-optimization-studio> Accessed Accessed on June 3rd, 2023
- Kant R (2012) Textile dyeing industry an environmental hazard. *Nat Sci* 4(1):22–26
- Karacapilidis NI, Pappis CP (1996) Production planning and control in textile industry: A case study. *Comput Ind* 30(2):127–144
- Karacizmeli IH, Ogulata SN (2017) Energy consumption management in textile finishing plants: A cost effective and sequence dependent scheduling model. *Text Appar* 27(2):145–152
- Kellerer H, Pferschy U, Pisinger D (2004) *Multidimensional Knapsack Problems*. Springer, Berlin

- Li S (2017) Parallel batch scheduling with inclusive processing set restrictions and non-identical capacities to minimize makespan. *Eur J Op Res* 260(1):12–20
- Li L, Qiao F, Wu Q (2009) Aco-based multi-objective scheduling of parallel batch processing machines with advanced process control constraints. *The Int J Adv Manuf Technol* 44(9):985–994
- Li D, Wang J, Qiang R, Chiong R (2021) A hybrid differential evolution algorithm for parallel machine scheduling of lace dyeing considering colour families, sequence-dependent setup and machine eligibility. *Int J Prod Res* 59(9):2722–2738
- Oztürk C, Ornek AM (2010) Capacitated lot sizing with linked lots for general product structures in job shops. *Comput & Ind Eng* 58(1):151–164
- Öztürk C, Ornek AM (2014) Operational extended model formulations for advanced planning and scheduling systems. *Appl Math Modelling* 38(1):181–195
- Ozturk E, Koseoglu H, Karaboyaci M, Yigit NO, Yetis U, Kitis M (2016) Sustainable textile production: cleaner production assessment/eco-efficiency analysis study in a textile mill. *J Clean Prod* 138:248–263
- Pasha J, Nwodu AL, Fathollahi-Fard AM, Tian G, Li Z, Wang H, Dulebenets MA (2022) Exact and metaheuristic algorithms for the vehicle routing problem with a factory-in-a-box in multi-objective settings. *Adv Eng Inf* 52:101623
- Periyasamy, A. P., & Militky, J. (2020). Sustainability in textile dyeing: recent developments. *Sustainability in the textile and apparel industries: Production Process Sustainability*, 37–79.
- Pirola F, Zambetti M, Cimini C (2021) Applying simulation for sustainable production scheduling: a case study in the textile industry. *IFAC-PapersOnLine* 54(1):373–378
- Pisinger, D., & Ropke, S. (2019). Large neighborhood search. *Handbook of metaheuristics*, 99–127.
- Research GV (2020) Textile Market Size, Share & Trends Analysis Report By Raw Material (Cotton, Wool, Silk, Chemical), By Product (Natural Fibers, Nylon), By Application (Technical, Fashion), By Region, And Segment Forecasts, 2022–2030. Accessed on 01/11/2023. <https://www.grandviewresearch.com/industry-analysis/textile-market>
- Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Trans Sci* 40(4):455–472
- Sáenz-Alanís CA, Salazar-Aguilar MA, Boyer V (2016) A parallel machine batch scheduling problem in a brewing company. *The Int J Adv Manuf Technol* 87(1):65–75
- Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget J-F (eds) *Principles and Practice of Constraint Programming – CP98*. Springer, Berlin, Heidelberg, pp 417–431
- Singh E, Pillay N (2022) A study of ant-based pheromone spaces for generation constructive hyper-heuristics. *Swarm Evolut Comput* 72:101095
- Swamidass PM (ed) (2000) *Time buckets*. Springer, Berlin
- Taillard ÉD (2023) *Decomposition Methods*. Springer, Cham. https://doi.org/10.1007/978-3-031-13714-3_6
- textiletutorials.com: Yarn Dyeing in Textile. Accessed on 01/11/2023. <https://textiletutorials.com/yarn-dyeing-in-textile-hank-dyeing-package-dyeing/>
- Yıldırım, F. F., Haşçelik, B., Yumru, Ş., & Palamutcu, S. (2019). Analysis of Water consumption and potential savings in a Cotton textile dye House in Denizli, Turkey. *In Water in Textiles and Fashion* (pp. 115–134). Woodhead Publishing.
- Zhang R, Chang P-C, Song S, Wu C (2017) A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes. *Knowl-Based Syst* 116:114–129
- Zhang Z, Guo C, Wei Q, Guo Z, Gao L (2021) A bi-objective stochastic order planning problem in make-to-order multi-site textile manufacturing. *Comput & Ind Eng* 158:107367
- Zhao H, Zhang C (2020) An online-learning-based evolutionary many-objective algorithm. *Inf Sci* 509:1–21
- Zhou H, Pang J, Chen P-K, Chou F-D (2018) A modified particle swarm optimization algorithm for a batch-processing machine scheduling problem with arbitrary release times and non-identical job sizes. *Comput & Ind Eng* 123:67–81
- Zhou Y, Wang J, Zhang P, Wang P, Lu Y, Zhang J (2020) Research on dyeing workshop scheduling methods for knitted fabric production based on a multi-objective hybrid genetic algorithm. *Meas Control* 53(7–8):1529–1539
- Zykin AV (2004) A lexicographic optimization algorithm. *Autom Remote Control* 65(3):363–368

Ege Duran earned her BSc and MSc degrees from the Industrial Engineering Department at Yasar University in Izmir, Turkey. She is currently a second-year PhD student at the School of Computer Science & IT at University College Cork in Ireland and funded by Science Foundation Ireland (SFI). Her research is centered on improving algorithms and exact methods, such as mixed integer programming, constraint programming, and heuristics.

Cemalettin Ozturk is a Lecturer in Logistics and Supply Chain Management at Munster Technology University and a Principal Investigator funded by Science Foundation Ireland (SFI). He has two decades of experience in various industrial, research, and academic organizations, serving as an engineer, technical contributor, and project coordinator in the fields of smart manufacturing, supply chain management, logistics, aviation, and telecommunication. His work focuses on the development and implementation of mathematical and artificial intelligence methods, including mixed integer programming, constraint programming, heuristics, digital twins, and data analytics across diverse domains.

M. Arslan Ornek received BSc and MSc degrees from Industrial Engineering Dept., Middle East Technical University, Ankara, Turkiye, and a PhD degree from the Dept of Production Engineering, University of Wales Institute of Science and Technology (UWIST), Cardiff, Wales, UK. After working for various industries and universities, Dr Ornek is now a faculty member at Yasar University, Department of Industrial Engineering and worked as Vice Rector responsible for Academic Affairs between the years 2017 and 2023. He lectures courses on manufacturing planning and control, and Operational Research, and is actively involved in managing technology development projects with industry. His research interests include Advance Planning and Scheduling Systems, Manufacturing Execution Systems, Manufacturing Systems Dynamics and Mathematical Modelling. He has published several articles in renowned journals and presented papers at national and international meetings.

Authors and Affiliations

Ege Duran¹ · Cemalettin Ozturk² · M. Arslan Ornek³

✉ Ege Duran
e.duran@cs.ucc.ie

Cemalettin Ozturk
cemalettin.ozturk@mtu.ie

M. Arslan Ornek
arslan.ornek@yasar.edu.tr

¹ SFI Centre for Research Training in AI, School of Computer Science & IT, University College Cork, Cork, Ireland

² Department of Process, Energy & Transport Engineering, Munster Technological University, Cork, Ireland

³ Department of Industrial Engineering, Yasar University, Izmir, Turkey