Introducing Preferences in Scheduling Applications

Nitin Srinath[a,*], I. Ozan Yilmazlar[a], Dr. Mary E. Kurz[a], Dr. Kevin Taaffe[a]

[a]*Department of Industrial Engineering, Clemson University, Clemson SC, 29634*

Abstract

In some applications like fabric dying, the machines used to process jobs must be set up and serviced frequently. The setup processes and associated setup times between jobs often depend on the jobs and the sequence in which jobs are placed onto machines. That is, the scheduling of jobs to machines must account for the sequence-dependent setup times as well. These setup times can be a major factor in operational cost. Also, the sequence in which jobs are processed is important for quality, i.e., there is a strong preference to run a certain type of job after another. There are also preferences associated with scheduling jobs on specific machines. In this paper, a mixed integer linear program has been presented to schedule jobs and account for setup times. The sequencing preferences have been modeled as two new objectives along with traditionally used objectives such as makespan, tardiness and total number of setups. The MILP is found to be useful only up to a certain level of problem sizes. To tackle larger problems, a problem size breakdown method has been developed. This method incorporates an MILP to reduce the possible machines on which a job can be processed, segmenting the problem into several smaller subproblems that can be solved to optimality. Next, in order to obtain solutions quickly, multiple construction heuristics have been developed based on SPT/LPT/EDD/LFJ rules and tested on a variety of industrially-inspired data sets. To produce schedules which can do well on multiple objectives at the same time, three additional tailored heuristics have been developed. Both the original and tailored heuristics are compared to the optimal solutions for small-sized problems. For larger problems, the heuristics are compared among themselves as well as against the problem size reduction method.

Keywords: Scheduling, Sequence-dependent setup times, MIP based heuristics, Preferences, Manufacturing optimization

1. Introduction

Sequencing and scheduling decisions play a critical role in today's industries, whether it is manufacturing, transportation, energy, or other service industries. Effective scheduling and sequencing decisions are

*Corresponding author

necessary for optimal use of resources and labor, meeting due dates, reducing costs, maintaining quality, etc. (Pinedo 2012)

An important scheduling problem is one of sequencing jobs with sequence dependent setup times (SDSTs). Setups are sometimes required to ensure machines are ready to run the jobs. Setups can also be viewed as loading-unloading actions within transportation problems (Zhu and Wilhelm 2006). Sometimes, the time required for these setups is dependent on the sequence of jobs scheduled. This is especially true for scheduling applications in fabric processing, wafer probing, circuit assembly, packaging, last mile delivery logistics, pulp and paper processing, biomass handling, shoe manufacturing, etc (Allahverdi 2015).

This paper addresses a scheduling and sequencing problem in the fabric dying industry with SDSTs, due dates and machine eligibility constraints. To reduce defects in the product, there are also sequence eligibility constraints that ensure certain products are not scheduled after others. For example, white fabrics should not be scheduled right after black fabrics on the same machine, due to the difficulty in removing all the black dye from the machine. To address these preferences, a preference matrix among the colors is available. For example, it is preferable to dye a purple fabric after a blue rather than a yellow after a blue. Also, when a schedule is generated, it is preferable to have some machines run lighter fabrics consistently and others run darker fabrics. Even if they move from one shade to another, it is preferable to have a smooth transition within the shades of the products scheduled on the same machine. These sequence preferences are also set to maintain quality. These sequence preferences, scheduling preferences and sequence eligibility constraints make this problem very unique. The preferences are modeled as objectives and the eligibility restrictions are modeled as constraints. The problem is inherently multi-objective due to the need to reduce delays, reduce imbalance among the machines in the schedule, reduce setups and maximize preferences.

In this paper, we develop a mixed integer linear program (MILP) that can handle any of the following objectives: minimize makespan, minimize setup times, minimize lateness, maximize scheduling preferences and minimize schedule inconsistencies. The MILP also has constraints to model sequence eligibility. The underlying scheduling problem is NP-hard and can be very time consuming to solve for even a small set of jobs (Sharma and Jain 2016). For larger problems, a problem size reduction methodology has been developed. In this method, we exploit the graph properties of our scheduling and assignment problem such that the original set of jobs and machines can be solved as several smaller sub-problems with the least amount of information being lost. These smaller problems are solved to optimality for any given objective, without a guarantee of optimality for the larger problem. In many ways, this method is similar to other construction heuristic methods such as SPT/LPT rule-based heuristics.

Heuristics have often been used in many scheduling applications because they provide feasible, good quality solutions even if they do not guarantee optimality. In the next section of this paper, multiple construction heuristics have been tested. The first set of heuristics include shortest processing time (SPT), longest processing time (LPT), earliest due date (EDD), least flexible job (LFJ) and a combination of them. These basic heuristics performed poorly especially on the new objectives. We then developed three new construction heuristics, which try to do well on all the objectives at the same time. Along with the problem size reduction method, a total of 11 heuristic methods are compared against each other and with respect to the optimal solutions for small-sized problems. For large-sized problems, the heuristics are compared against each other and with the semi-optimal solutions from the problem size reduction method. Overall, the new heuristics perform well on the new objectives while being just as good on the traditional objectives like makespan and setups. Their design makes them more useful for larger problems as well.

2. Literature Review

The literature in scheduling can be classified in many ways (Abedinnia et al. 2017):

1. Deterministic or stochastic based on the variability of the parameters involved,
2. Dynamic or static based on how the parameters and conditions change through time,
3. Based on the type of machine settings - single machine, parallel machine, flow shop etc.,
4. Based on the need for setup times - sequence dependent or independent,
5. Based on objectives - minimizing total completion time, minimizing make span, minimizing setup times, minimizing costs, minimizing tardiness or lateness, etc.,
6. Based on constraints they need to incorporate such as setup times, machine eligibility, due dates, etc.,
7. Based on the type of methods employed: exact methods such as MILPs, Branch and bound (BnB) or branch and cut techniques, or approximate methods such heuristic algorithms etc.

The problem that we address in this project is a parallel machine, deterministic, static scheduling problem with constraints of machine eligibility, due dates and sequence dependent setup times. The objectives of the methods developed are to reduce setups, reduce makespan, reduce lateness and obtain schedules that maximize the sequence and schedule preferences.

There are several papers in the literature that have addressed the problem of sequence dependent setup times on parallel machines. The following literature review is structured based on the methods used (both exact and heuristic) as well as by meeting objectives within the methods.

Most authors who consider scheduling problems with SDSTs to optimality have developed MILPs and solved them using commercially available solvers such as CPLEX or Gurobi. Others have developed branch and bound algorithms which are fed a partially feasible solution and work through the steps of the algorithm to reach the optimal solution.

Most of the papers that have proposed MILPs have either minimized tardiness (Anderson et al. 2013; Lin and Hsieh 2014; Naderi-Beni et al. 2014; Kim, Song, and Jeong 2019) or reduced the makespan (Kurz and Askin 2001; Avalos-Rosales, Angel-Bello, and Alvarez 2015; Hou and Guo 2013; Hu and Yao 2011a; Hu and Yao 2011b; Hamzadayi and Yildiz 2017; Afzalirad and Rezaeian 2016). The sum of completion times of all jobs (total completion time) is also a commonly-used objective (Ruiz and Andrés-Romano 2011; Fan and Tang 2006; Gokhale and Mathirajan 2012; Tsai and Tseng 2007). Akyol and Bayhan (2008) and Toksarı, Oron, and Güner (2010) have proposed MILPs with weighted sums of earliness and tardiness as objectives.

While earliness and tardiness are seen from a customer-standpoint (due-date based), makespan or total completion time objectives are resource-conserving approaches (Abedinnia et al. 2017). To address these competing goals in a scheduling context, a common approach is to introduce a multi-objective MILP. Gharehgozli, Tavakkoli-Moghaddam, and Zaerpour (2009) uses a weighted sum of total completion time and total tardiness. Tavakkoli-Moghaddam et al. (2009) has used the same two objectives but minimizes the objectives one after another in lexicographical order. Li et al. (2012) uses a goal programming approach to multi-objective problems where one of the objectives is converted as a constraint while optimizing over the other. This paper uses makespan and total tardiness as the two objectives. Rocha et al. (2008) uses an unweighted sum of tardiness and makespan as a single objective. Wang, Wang, and Chen (2013) uses a similar approach but instead uses a weighted sum of total number of late jobs and makespan as objectives.

While these papers address the importance of having jobs ready on time (tardiness) or balancing the available resources (makespan) in a sequence dependent setup environment, they do not consider reducing the setup time or cost as a primary objective. Setup times are included within the constraints to ensure that the schedules are feasible. In that direction, a vehicle routing problem-based MILP has been developed to reduce setup times (Dinh and Bae 2012). They use a sum of total tardiness and total setup time as their objective. But in this case, the setup times are dynamic and are used as variables that depend on past sequences and are not determined by what job runs after the another. In most cases, the setup times between jobs are considered as parameters (s_{ij}) as the setup time between job i and job j if j is immediately processed after j. Only one paper (Joo and Kim 2012) has proposed a MILP model to minimize a weighted sum of setup times and tardy times, but they have concluded that the model is not tractable for any more than 10 jobs.

Those who have not used MIPs/MILPs with solvers have developed their own branch and bound algorithms. These algorithms make use of unique properties of the problem to develop better bounds and reach optimal solutions faster. Bajestani and Tavakkoli-Moghaddam (2009) uses a BAB algorithm to minimize tardiness and Gacias, Artigues, and Lopez (2010) have used a BAB algorithm to minimize total completion times. Rocha et al. (2008) have used a BAB algorithm along with their MILP formulation to minimize a weighted sum of tardiness and makespan. However, the method developed by Rocha et al. (2008) can only perform well for 30 or fewer jobs. The BAB algorithm at its worst can evaluate all the possible solutions of a combinatorial problem. Also, the computational efficiency of a BAB algorithm also depends on the starting solution fed to it and its ability to compute better bounds. Branch and bound and branch and cut algorithms are generally used by MILP solvers as one of the steps in solving MILPs.

In the last ten years, mixed integer linear programs (MILPs) have not been discussed in the literature of scheduling due to their observed ineffectiveness (Abedinnia et al. 2017). From the above section, it is evident that exact methods currently available are not tractable for large sets of jobs. So, most authors also have developed a plethora of heuristic or approximate algorithms which are computationally way more efficient than exact methods while guaranteeing solutions that are at least close to optimality.

The objective of reducing setup times alone has not been addressed. While reducing the makespan or total completion time does help in reducing setups to an extent, in some cases, it is important to reduce setups further at the expense of increasing the makespan or total completion. In that direction, Joo and Kim (2012) has used a genetic algorithm to reduce a weighted sum of tardiness and setup times. Other than this work, there is no evidence of work done towards explicitly and directing reducing setups to the best of our knowledge. This is important to address as in some applications, the setups are the most cost-adding/ labor intensive component of the process. Joo and Kim (2012) is also the only paper to have developed an MILP which minimizes setup times as an objective but they have declared that the MILP is computationally not feasible for any more than 10 jobs. Also, there has been no work done previously to address the quality-based preferences and sequencing constraints while sequencing jobs. This approach is important in scheduling applications in the fabric dying industry to maintain high quality standards.

Another common solution approach is to employ heuristic and meta-heuristic methods. Having a problem-specific set of rules to schedule jobs helps to design heuristics in some cases. Kang and Shin (2010), Logendran, McDonell, and Smucker (2007), and Xi and Jang (2012) have used specific dispatching rules are used to schedule jobs and minimize a single objective such as tardiness (Logendran, McDonell, and Smucker 2007; Xi and Jang 2012) or a mix of objectives such as late jobs, completion times etc., (Kang and Shin 2010).

The commonly used basic dispatching rules are SPT (shortest processing time), W-SPT (Weighted SPT), EDD (Earliest due date), LPT (Longest processing time), etc. There are also more complex dispatching rules such as ATC (Apparent Tardiness Cost) (Vepsalainen and Morton 1987), COVERT (Carroll 1965) and MODD (modified operation due date) (Baker and Bertrand 1982) which are specifically designed for minimizing weighted tardiness. SPT is commonly used to reduce total completion times while LPT is commonly used to reduce makespan. For a single machine scheduling problem with setups and the aim of reducing weighted tardiness, the ATCS (ATC with setups) rule has been developed (Lee, Bhaskaran, and

Pinedo 1997). Lee and Pinedo (1997) proposed a heuristic for a parallel machine problem with SDSTs to obtain a starting solution for a simulated annealing procedure using the ATCS rule with an estimated makespan as input. Their objective was to reduce weighted tardiness as well. Chen (2009) has also developed a heuristic based on ATCS and simulated annealing for a batch scheduling problem with parallel machines and SDSTs. Liao, Lee, and Tsai (2016) have proposed a heuristic to minimize makespan in which the dispatching rule uses an index based on setup times on the machine and similarity between the jobs. Since the similarity between the jobs affected the setup times between them, this index helped reduce setup times overall. Liao, Tsai, and Chao (2011) proposed a dispatching rule for a parallel machine problem with machine eligibility based on the LFJ (Least flexible job) order to reduce the makespan. This rule is used to create an initial solution for an ant colony optimization meta-heuristic. Coffman, Garey, and Johnson (1978) proposed a heuristic based on a bin-packing approach where each machine is given a certain time capacity to accept jobs, and if all machines reach that capacity, then additional time is added to each machine. This method also aims to reduce makespan. Kurz and Askin (2001) has also proposed a similar Multi-fit heuristic algorithm to minimize makespan for a problem with parallel machines, ready times, and SDSts.

Another type of construction heuristics used are insertion based heuristics which are extensively used to schedule tasks on to processors in computer systems. Insertion based scheduling policies are used in many computer task scheduling applications such as in Daoud and Kharma (2006). A few such insertion based heuristics have also been used to find solutions to vehicle routing and delivery problems (Lu and Dessouky 2006; Campbell and Savelsbergh 2004). Insertion based greedy heuristics have been used in Zeng, Che, and Wu (2018) to minimize makespan and total electricity costs. Kurz and Askin (2001) has also proposed an insertion based heuristic for a scheduling problem to reduce makespan.

Greedy algorithms have been used in (Arango, Giraldo, and Castrillon 2013; Lin, Lu, and Ying 2011; Ying and Cheng 2010) to minimize tardiness and in (Kampke, Arroyo, and Santos 2010) to minimize total completion time. In Kim, Park, and Lee (2017), greedy algorithms have been used to minimize flow time in job shops with SDSTs.

With better computational power and improved MILP solvers, there has been a recent trend of using MILP based heuristics to solve these problems. Gestrelius, Aronsson, and Peterson (2017) has proposed a MILP based heuristic for a train timetabling problem where problems are trains are added in batches and solved to optimality. Any infeasible boundary solutions are corrected by rule-based heuristics. Servare Junior et al. (2020) has developed a LP relaxation based heuristic where the integer program is relaxed as an LP, and constraints added iteratively to approximate the values of the variables. Basán et al. (2020) has proposed a decomposition based method to minimize makespan in a flexible manufacturing environment. In this method, an initial solution is constructed using a rule-based heuristic. This solution is improved iteratively using an MILP which re-solves for a small subset of jobs from the initial solution. Umetani, Fukushima, and Morita (2017) has proposed a linear relaxation based heuristic for obtaining electric vehicle charging schedules which minimize electricity costs. In this method, the values of the LP relaxation are rounded to the nearest integer and if the solution obtained by the approximation is infeasible, additional rules are incorporated to fix the values of variables in such solutions.

Meta-heuristic methods have often been used to quickly and efficiently generate good-enough solutions. This is evident in the large number of papers that have developed or utilized these methods. The most commonly used meta-heuristic methods are genetic algorithms, tabu search, simulated annealing, randomized neighborhood search and ant colony/ bee colony optimization.

From the literature review above, it can be seen that the objective of reducing setup times alone has not been addressed. In that direction, (Joo and Kim 2012) has used a genetic algorithm to reduce a weighted sum of tardiness and setup times. Other than this work, there is no evidence of work done towards minimizing setups to the best of our knowledge. This is important to address as in some applications, the setups are the most cost-adding/ labor intensive component of the process. (Joo and Kim 2012) is also the only paper to have developed an MILP which minimizes setup times as an objective but they have declared that the MILP is computationally not feasible for any more than 10 jobs. We are also unaware of work to address the quality-based preferences and sequencing constraints while sequencing jobs. This approach is important in scheduling applications in the fabric dying industry to maintain high quality standards. Key contributions of this paper are the introduction of novel exact and heuristic methods that can effectively incorporate these newly formulated objectives and constraints.

3. Problem definition

The problem addressed in this paper is a scheduling problem with parallel machines. The objectives include minimizing makespan, minimizing total lateness, minimizing setup times, maximizing color preferences and minimizing shade inconsistencies on the machines. The constraints include machine eligibility, sequence dependent setup times, and sequence dependent restrictions.

3.1. Parameters

The parameters for the formulation include the machine eligibility ($\alpha_{i,j}$) and setup times between jobs (β_{i_1,i_2}), preference between jobs (γ_{i_1,i_2}), due dates (d_i) and processing times (d_i) for every job, ready times for every machine (r_j), shade values for every job (s_i) and based on their shade values, they are grouped into shade groups (s_i^g).

The values for γ come from a color preference matrix among the jobs and the preference scale goes from 0 to 1, 1 being the most preferred. The shade values for each job range from 0 to 100, 100 being the lightest.

The shade values of the jobs are used to group them into shade groups. There are 4 groups: Very dark (1-25), dark (26-50), light (51-75), very light (76-100).

$$\alpha_{i,j} = \begin{cases} 1, & \text{if job } i \text{ can be dyed on jet } j \\ 0, & \text{otherwise} \end{cases}$$

β_{i_1,i_2} = setup time required between jobs i_1 and i_2

γ_{i_1,i_2} = preference between jobs i_1 and i_2

d_i = due date for job i

p_i = amount of time needed to dye job i

r_j = ready time for jet j (or) finish time of current job in jet j

s_i^g = shade group for job i

s_i = shade value for job i

3.2. Variables

The variables for this formulation include the scheduling variables ($x_{i,j}$), start and end variables ($x_{i,j}^{strt}$ and $x_{i,j}^{end}$), sequencing variables (y_{i_1,i_2}) which are all binary. The continuous variables include finish times for each job, makespan of the schedule, lateness or delay for each job and shade and shade group differences.

$$x_{i,j} = \begin{cases} 1, & \text{if job } i \text{ is processed on jet } j \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i,j}^{strt} = \begin{cases} 1, & \text{if job } i \text{ is the first job on jet } j \\ 0, & \text{otherwise} \end{cases}$$

$$x_{i,j}^{end} = \begin{cases} 1, & \text{if job } i \text{ is the final job on jet } j \\ 0, & \text{otherwise} \end{cases}$$

$$y_{i_1,i_2} = \begin{cases} 1, & \text{if job } i_1 \text{ is processed immediately before } i_2 \\ 0, & \text{otherwise} \end{cases}$$

c_i = finish time for job i

c_{max} = makespan of the schedule

u_i = lateness for job i

s_{i_1,i_2}^d = shade difference between successive jobs i_1 and i_2

s_j^{gd} = shade group difference from first to last job on machine j

4. Formulation - OPS

In this section, we propose an MILP formulation, referrred to as OPS, since the focus of our work is to introduce new **O**bjectives and constraints to tackle the **P**references and **S**equence-based restrictions unique to this problem.

4.1. Objectives

There are five objectives in this problem: Balance the workload on the different machines, minimizing delays on jobs to meet service levels, reducing setups in order to reduce setup costs associated with cleaning chemicals and loss of time, making preferable color transitions among jobs on the same machine and running consistent shades on machines, in order to maintain good quality of output.

In order to balance the machines, the makespan objective is used. The lateness objective is used to minimize delays and the setup objective to reduce setups between the jobs. These three objectives have been explored by researchers in the past. They have been listed in equations (1), (2), (3).

$$\text{Makespan:} \quad \min c_{max} \tag{1}$$

$$\text{Lateness:} \quad \min \sum_{i} u_i \tag{2}$$

$$\text{Setup times:} \quad \min \sum_{i_1,i_2} \beta_{i_1,i_2} y_{i_1,i_2}/(n-m) \tag{3}$$

This paper introduces two new objectives to address the preferences.

Average color preference: This objective has been introduced to handle the transition of colors in the scheduling of jobs. For example, it is preferred to schedule an orange colored fabric right after a red fabric instead of a blue fabric.

If there are two jobs i_1, i_2 being processed on the same machine with i_1 immediately preceding i_2, the color preference of moving from i_1 to i_2 must be counted into the new objective. If there were n jobs and m machines, then the total number of such transitions would be $n-m$. For a pair of jobs of i_1, i_2 which are run in immediate succession on the same machine, and the color preference involved in the transition is γ_{i_1,i_2}, then the objective is defined as in equation (4).

$$\text{Average Color Preference:} \quad \max \sum_{i_1,i_2} \gamma_{i_1,i_2} y_{i_1,i_2} \tag{4}$$

Average shade difference: This objective has been introduced to this problem to handle two issues. First, in the fabric dying industry, it is important to schedule similar shades of colors together on machines. Second, it is important to schedule machines to consistently process darker colors or lighter colors. The color preference objective is not enough to handle this aspect of scheduling because there might be two blue cloths for example, one of which is a light shade but the other is a dark shade. Scheduling these two colors one after another would be acceptable with respect to the color preference objective but there might be impact on quality when the lighter blue is scheduled immediately after the darker blue.

This objective is formulated as a sum of two parts: The first part handles transitions among the jobs with respect to the shades, while the second part handles the shade differences on a machine from the first to the last scheduled job on that machine.

If two jobs i_1 and i_2 are scheduled on the same machine, i_2 immediately after i_1, then let their shade difference be $s^d_{i_1,i_2}$. For the entire schedule, There are $n - m$ such pairs of jobs which are run in immediate succession.

The second part of this objective rewards schedules with machines consistently running similarly shaded fabrics. For example, if the first job scheduled on a machine is very dark and the last job on the same machine is light, then the shade group difference for that machine is $3 - 1 = 2$. Let the shade group difference for every machine j be s^{gd}_j. Then, the objective is defined as in equation (5). The denominators of 200 and 6 correspond to the maximum possible difference in shades and shade groups of 100 and 3 respectively, along with a 1/2 weight for each part of the objective.

$$\text{Average Shade Consistency:} \quad \min \sum_{i_1,i_2} s^d_{i_1,i_2}/200(n-m) + \sum_j s^{gd}_j/6m \qquad (5)$$

4.2. Constraints

The constraints ensure that the solutions returned from solving the formulation are feasible. Constraint set (6) ensures that a job can only be processed on a machine for which it is eligible. Constraint set (7) ensures that if a job is the starting job on a machine, then it is being processed on that machine. Constraint set (8) ensures that if a job is the ending job on a machine, then it is being processed on that machine. Constraint sets (7) and (8) link the x_{strt}, x_{end} and the x variables. Constraint sets (9) and (10) ensure that there is only one starting job and ending job for each machine. Constraint set (11) ensures that each job is processed by a machine. Constraint set (12) ensures that no job precedes itself. Constraint set (13) ensures that for every pair of jobs, there are only three scenarios: the first job can immediately precede the second job on the same machine, the second job can immediately precede the first job on the same machine, or they are not in immediate succession on the same machine. Constraint set (14) ensures that if two jobs are on different machines, they cannot immediately precede or succeed each other on the same machine. Constraint set (15) prevents two jobs that immediately precede or succeed each other on the same machine from overlapping. Constraint set (16) ensures that the finish time for each job is greater than its processing time. Constraint set (17) ensures that the makespan is the maximum completion time among all jobs. This constraint works in combination with the makespan objective. Constraint set (18) ensures that the delay for each job is greater than or equal to 0. If a job is processed ahead of time, there is no negative delay which adds incentive to the lateness objective. Constraint set (19) ensures that the delay is greater than the difference between the finish times and due dates for the jobs. This constraint works in combination with the lateness objective. Constraint sets (20) and (21) ensures that the shade difference between two jobs that are in immediate succession on the same machine is taken into account when there is transition from a darker shade to a lighter shade. This type of transition is not preferable so these constraints ensure a penalty when combined with the shade consistency objective. Constraint sets (22) and (23) ensure that the shade transition between jobs is limited to a value of 60 for this paper. Constraint sets (24) and (25), in combination with the shade consistency objective, ensure that the shade group difference between the first and the last jobs on each machine is taken into consideration when computing the objective score for the schedule. Constraint sets (26)-(29) ensure that the binary restrictions on the variables are satisfied.

$$x_{i,j} \leq \alpha_{i,j} \quad \forall i,j \tag{6}$$

$$x_{i,j}^{strt} \leq x_{i,j} \quad \forall i,j \tag{7}$$

$$x_{i,j}^{end} \leq x_{i,j} \quad \forall i,j \tag{8}$$

$$\sum_i x_{i,j}^{strt} = 1 \quad \forall j \tag{9}$$

$$\sum_i x_{i,j}^{end} = 1 \quad \forall j \tag{10}$$

$$\sum_j x_{i,j} = 1 \quad \forall i \tag{11}$$

$$y_{i,i} = 0 \quad \forall i \tag{12}$$

$$y_{i_1,i_2} + y_{i_2,i_1} \leq 1 \quad \forall i_1 < i_2 \tag{13}$$

$$x_{i_1,j} + \sum_{j_1 \in J/j} x_{i_2,j_1} + y_{i_1,i_2} + y_{i_2,i_1} \leq 2 \quad \forall i_1, i_2, j \tag{14}$$

$$c_{i_1} \leq c_{i_2} - p_2 - \beta_{i_1,i_2} + M(1 - y_{i_1,i_2}) \quad \forall i_1, i_2, j \tag{15}$$

$$c_i \geq p_i \quad \forall i \tag{16}$$

$$c_{max} \geq c_i \quad \forall i \tag{17}$$

$$u_i \geq 0 \quad \forall i \tag{18}$$

$$u_i \geq c_i - d_i \quad \forall i \tag{19}$$

$$s_{i_1,i_2}^d \geq (s_{i_2} - s_{i_1}) y_{i_1,i_2} \quad \forall i_1, i_2 \tag{20}$$

$$s_{i_1,i_2}^d \geq 0 \quad \forall i_1, i_2 \tag{21}$$

$$s_{i_1,i_2}^d \leq 60 \quad \forall i_1, i_2 \tag{22}$$

$$(s_{i_1} - s_{i_2}) y_{i_1,i_2} \leq 60 \quad \forall i_1, i_2 \tag{23}$$

$$s_j^{gd} \geq \sum_i s_i^g x_{i,j}^{strt} - \sum_i s_i^g x_{i,j}^{end} \quad \forall j \tag{24}$$

$$s_j^{gd} \geq \sum_i s_i^g x_{i,j}^{end} - \sum_i s_i^g x_{i,j}^{strt} \quad \forall j \tag{25}$$

$$x_{i,j} \in \{\, 0, 1\} \quad \forall i,j \tag{26}$$

$$y_{i_1,i_2} \in \{\, 0, 1\} \quad \forall i_1, i_2 \tag{27}$$

$$x_{i,j}^{strt} \in \{\, 0, 1\} \quad \forall i,j \tag{28}$$

$$x_{i,j}^{end} \in \{\, 0, 1\} \quad \forall i,j \tag{29}$$

5. Problem Size Reduction - PSR

The OPS formulation was first tested for different sizes of problems. We observed that we could only solve OPS for small sized problems (up to 50 jobs and 5 machines). This conclusion is in line with other work in literature where such problems have been difficult to solve with exact methods due to the NP-hard nature of the problems (Sharma and Jain 2016). In this section, we segment the original OPS formulation into smaller manageable problems through what we call the Problem Size Reduction (PSR) method. We do this by breaking down the original, single connected graph into smaller unconnected graphs, removing as few edges as possible in the process. Each of these smaller problems can be solved to optimality, the results of which can be aggregated to form a feasible but possibly sub-optimal solution.

Consider a set of jobs N which must be scheduled on a set of machines M. Let the bipartite graph representing the machine eligibility for the jobs be $G(V,E)$, where V represents the set of nodes corresponding to the jobs and the machines, and E represents the set of edges, with an edge being present between a job node and a machine node if the job is eligible to be processed on that machine. If this graph is too large, it is hard to solve OPS to optimality. But, if we remove a few edges in this graph, we can tackle the different disconnected subgraphs of this altered graph as separate problems. We can create optimal schedules for each of the subgraphs and combine them to get a complete schedule which is feasible as well. This is an instance of a graph partitioning problem with constraints. This is proved to be NP-Hard in Ji (2004).

As an additional restriction, there must be at least one edge out of every node, i.e., every job must have at least one machine eligible after the reduction. The PSR method divides the given graph into two disconnected subgraphs by removing some edges from the given graph and may be iteratively applied to the larger remaining subgraph.

5.1. Variables

The variables include the v variables that determine the side of the partition on which a node is present, and the w variables that determine which edges are removed.

$$v_i = \begin{cases} 1, & \text{if node i is in subgraph 1} \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in V$$

$$w_{i,j} = \begin{cases} 1, & \text{if edge (i,j) is removed} \\ 0, & \text{otherwise} \end{cases} \qquad \forall (i,j) \in E$$

5.2. Objective

The objective of this MILP (30) is to reduce the number of edges removed from the given graph. The fewer the edges removed, the less the loss of information that might affect solution quality.

$$\min \sum_{(i,j)\in E} w_{i,j} \tag{30}$$

5.3. Constraints

Let Q represent the maximum allowable size of the subgraph. For this paper, the maximum value of Q is set to 60, which is the maximum size of problems the MILP has been able to solve. If the MILP in section 4 can be improved to handle larger sized problems, then Q can be larger. It then follows that constraint set (31) ensures that there are exactly Q nodes in the first subgraph and the rest of the nodes are in the second subgraph. Constraint sets (32) and (33) ensure that if two nodes with an edge between them are in different subgraphs, then that edge is removed from the given graph. Constraint set (34) ensures that each node has at least one other node connected to it. This ensures that every job has at least one machine on which it can be processed. Constraint sets (35) and (36) ensure binary restrictions are satisfied for the variables. Note that the w variables need not be constrained to be binary due to the enforcement of constraint sets (32) and (33).

$$\sum_{i\in V} v_i = Q \tag{31}$$

$$w_{i,j} \geq v_i - v_j \quad \forall (i,j)\in E \tag{32}$$

$$w_{i,j} \geq v_j - v_i \quad \forall (i,j)\in E \tag{33}$$

$$\sum_{j\in V-i} w_{i,j} \leq |G(i) - 1| \quad \forall i \tag{34}$$

$$0 \leq w_{i,j} \leq 1 \quad \forall (i,j)\in E \tag{35}$$

$$v_i \in \{\, 0, 1\} \quad \forall i \in V \tag{36}$$

5.4. Example

Let us look at an example with 500 jobs and 50 machines. Note that this problem when translated to a graph as described above has 550 nodes. Let Q be 50 for this case. This graph is first broken down into two subgraphs containing 50 and 500 nodes each. The 50 node subgraph can now be independently solved to optimality for any of the objectives. The 500 node subgraph is again divided into 50 and 450 node subgraphs. Similarly, the entire 550 node original graph is broken down in iterations into 11 subgraphs of 50 nodes each of which can be solved independently using the formulation proposed in section 4. Once the problems are all solved, the solutions are combined to obtain a feasible solution (sub-optimal) to the original problem.

6. Construction Heuristics

The initial aim of this work was to develop methods that could help an industrial client with creating schedules. The methods had to be fast and flexible, without a significant loss in solution quality. The MILP was not fast and could not produce optimal solutions for large problems. Next, the problem size reduction MILP was proposed as a way to provide semi-optimal solutions in a reasonable amount of time. However, this method was also not fast enough for problems with over 300 nodes. Moreover, our industrial client was skeptical about its ability to maintain a solution based on an optimization approach. So, there was a need to develop methods which were quick and could be altered slightly to return solutions which were reasonably good on all five objectives. Considering that the problem size reduction method is similar to a construction heuristic, we started with the simplest and most common types of scheduling heuristic algorithms.

6.1. Construction Heuristic 1: H1 (Job First Approach)

Without machine eligibility constraints and sequence-dependent setup times, the Shortest Processing Time (SPT) rule is optimal for the objective of reducing total completion time. Similarly, for the makespan objective, Longest Processing Time (LPT) rule is optimal. Similarly, Earliest Due Date (EDD) rule is optimal for the lateness objective. When there are machine eligibility constraints, but all processing times are equal, the Least Flexible Job (LFJ) rule is optimal (Pinedo 2012). The motivation behind the first set of heuristics is that even with the machine eligibility, sequence-dependent setups and sequence-dependent restrictions among the jobs, these rules might give good-enough solutions. So, in heuristic H1, these rules are used to first sort the jobs based on their processing times, due-dates or flexibility. To obtain balanced solutions across multiple objectives, a combined sort of processing time or flexibility and due dates was also used. Once the list of sorted jobs was ready, the jobs would be placed on the machine which could finish the job first. When the list of machines available for the jobs is obtained, the machine eligibility rules and sequence-dependent restrictions are taken into consideration. For example, if a job is not eligible on the first machine, the first machine will not be listed for that job. Also, if a job is a very light color, then a machine which is currently running very dark colors is not listed because of the large shade difference. Once the machine is decided for the job, the machine ready-time is updated and the next job is selected for placement. Algorithm 1 is detailed below. Given that the method loops through all the jobs, its worst case running time is $O(nm)$.

Algorithm 1 Construction Heuristic H1

```
Obtain list of jobs
for every job $i$ in list of jobs do
    Obtain list of eligible machines
    for every machine $j$ in list of eligible machines do
        Finish time for job $i$ on machine $j$ = $\begin{cases} \text{finish time of current job on machine } j + \\ \text{setup time between current job and job } i + \\ \text{processing time for job } i \end{cases}$
    end for
    Place job $i$ on the machine where it finishes first
    Update current job and finish time on the chosen machine
end for
```

6.2. Construction Heuristic 2: H2 (Machine First Approach)

When developing the first set of heuristics H1, we did not explicitly tackle the setup or the preference objectives. This resulted in solutions that were poor with respect to these objectives. Also, to better tackle the preference objectives, the industrial client suggested a method that looked at machines selecting the next job as opposed to jobs selecting the next machine. With that concept in mind and the need to address the new objectives, heuristic H2 has been developed. In this heuristic, the list of jobs is not sorted. To start the algorithm, the machine which is ready first is selected. Based on the machine eligibility and sequence-dependent restrictions, the list of jobs which can be received by this machine is obtained. For each job on that list, a score is given based on the setup time, preference and shade difference between the current job on the machine and the incoming job.

For example, consider a machine which is currently running a light blue job, and there is an incoming light green job. In this case, there is no need for a setup. Also, the preference is 0.5, considering the change in color and the shade difference is also 0. We calculate a "job score" to represent the unweighted normalized average of setup times, preferences and shade differences. In such a case, the score for the incoming job would be $[(0/10) + (0.5/1) + (0/100)]/3$ if the maximum setup time across all jobs was 10 hours. Once the score for each jobs is obtained, the machine receives the job with the best score and earliest due date, and the ready-time and current job on the machine is updated. Then, the next machine in the list is selected to receive a job.

How does this method tackle all the objectives? The rule for selecting the machine is earliest ready times. This rule intuitively ensures a balance of jobs on the machines. The score function incorporated into the algorithm aims to improve the three objectives of setup times, color preferences and shade consistency. The ties broken using due-dates help improve the lateness objective. Algorithm 2 is presented below. Since this algorithm looks at all jobs in each iteration in the worst case, the running time is $O(n^2)$.

Algorithm 2 Construction Heuristic H2

```
Obtain list of jobs to be placed
Obtain list of machines
while Number of jobs to be placed $> 0$ or Number of eligible machines $> 0$ do
  Select the machine with the earliest ready time
  Obtain list of eligible jobs that can be placed on chosen machine
  if Number of eligible jobs=0 for chosen machine then
    Remove chosen machine from list of machines
    Continue
  end if
  for every job $i$ in the list of eligible jobs do
    Record the finish time of job $i$ if it is placed on chosen machine after the last job
  end for
  Select the job which has the least finish time among all the eligible jobs
  Update the ready time for chosen machine
  Remove chosen job from list of jobs to be placed
end while
```

6.3. Construction Heuristic 3: H3 (Machine First Insertion Approach)

Algorithm H2 worked well to improve the quality of schedules over H1. However, when the schedules were being inspected, it was evident that improvements were possible. When a schedule is being created, the machines in the factory would be processing some jobs. These jobs cannot be altered on the schedule since they are already being processed. But, when the schedule is being created for new jobs, we need not place jobs one after another. We can also insert jobs in between other jobs so that the final schedule is even better on the various objectives. This is the reasoning behind the development of heuristic H3. In this heuristic also, the machine first approach is used. However, when a machine is going to receive a job, it does not necessarily receive a job at the end of its schedule. The jobs can be inserted between other jobs at the time of creation of the schedule.

Once a machine is selected, the available slots on the machine is obtained. If no job has been placed on the machine yet (at the beginning of schedule creation), then there is only one slot available. Stated formally, if a machine has x jobs, then there will be $x + 1$ available slots on that machine. Then, each job-slot pair gets a job-slot score. This score depends on the setup times, preferences and shade differences between the job in question and the jobs adjacent to the slot on the schedule. The job and slot with the best score is selected and the machine ready time is updated. The schedule is also updated.

Algorithm 3 is detailed below, with a computation time complexity for this algorithm is $O(n^3)$ if $n > m$.

Algorithm 3 Construction Heuristic H3

```
Obtain list of jobs to be placed
Obtain list of machines
while Number of jobs to be placed ¿ 0 or Number of eligible machines ¿ 0 do
  Select the machine with the earliest ready time
  Obtain list of eligible jobs that can be placed on chosen machine
  if Number of eligible jobs=0 for chosen machine then
    Remove chosen machine from list of machines
    Continue
  end if
  Obtain all the eligible slots for chosen machine
  for every job $i$ in the list of eligible jobs do
    for every available slot $k$ in list of eligible slots for chosen machine do
      Obtain the slot-job score of job $i$ if it is placed on slot $k$ chosen machine
    end for
  end for
  Select the job and slot pair which has the best slot-job score
  Update the ready time for chosen machine
  Remove chosen job from list of jobs to be placed
end while
```

6.4. Construction Heuristic 4: H4 (Machine Filling Approach)

The algorithm H3 provided benefit across all the objectives but lateness, and the makespan values and schedule were both acceptable to the industrial client. However, there was still opportunity for improvement in the setup and preference objectives. If the makespan values from H3 were acceptable, we could use those values to continuously place jobs using the same job-slot rules on the same machine until the makespan value for that machine is reached. This helps keep similar jobs together on the same machine and thus improves the setup times and preference objectives. This is the reasoning behind the development of heuristic H4, where a machine is selected and filled with jobs until a pre-determined capacity is reached. Then the algorithm moves to the next machine. Algorithm 4 is detailed below, with a computational complexity for this algorithm also is also $O(n^3)$ if $n >> m$.

Algorithm 4 Construction Heuristic H4

```
Obtain list of jobs to be placed
Obtain list of machines
From the schedule created by H3, obtain total running times for all machines
Assign current running time for every machine = 0
while Number of jobs to be placed > 0 or Number of eligible machines > 0 do
    if All machine current running times > total running times then
        Add one unit of time to total running times for all machines
    end if
    for Every machine $j$ in the list of machines do
        Obtain list of eligible jobs that can be placed on $j$
        if Number of eligible jobs=0 for chosen machine then
            Remove chosen machine from list of machines
            Continue
        end if
        while current machine time for $j$ < total running time for $j$ do
            Obtain list of eligible jobs that can be placed on $j$
            Obtain all the eligible slots for $j$
            for every job $i$ in the list of eligible jobs do
                for every available slot $k$ in list of eligible slots for chosen machine do
                    Obtain the slot-job score of job $i$ if it is placed on slot $k$ on $j$
                end for
            end for
            Select the job and slot pair which has the best slot-job score
            Update the current running time for $j$
            Remove chosen job from list of jobs to be placed
        end while
    end for
end while
```

7. Computation results for PSR

For the problem sizes of interest (≤ 50machines, ≤ 500jobs), the construction heuristics could all solve within 30 seconds. So, they were not subjected to a thorough computational analysis. However, the PSR method involves the process of solving two MILPs multiple times. For larger problems, it takes a significant computational effort. For this reason, the PSR method was tested using several combinations of machines and jobs. Each splitting process was limited to 10 minutes. The results are detailed in Figures 1-3. From Figure 1, it can be seen that on average, when the number of nodes in the graph (machines+jobs) is large, the splitting MILP takes longer to solve. This is due to increased complexity with more constraints and variables. This does not provide the entire picture, because for the same number of nodes, the solution times can be different. From Figure 2, it is evident that in most cases, when the number of machines is the same, the solution time does not necessarily increase or decrease with the increasing number of jobs. For very large problems, however, we can conclude that with increasing number of jobs, the solution time increases. From Figure 3, we observe a consistent trend in which solution quality increases as the number of jobs increases, given the same number of machines. This is because when the number of jobs increases, the number of splits also increases. This is especially true for the initial splits that result from very large problems. For example, if there are 500 jobs and 50 machines, then the total number of splits is 11 (50 nodes in each split). So, in the first iteration, the algorithm splits the graph into 50 nodes and 450 nodes, which is a very skewed split. Even though the gap cannot reach 0 by the imposed time limit for problems with many nodes, the skewed nature of the split ensures a lower optimality gap. However, when a graph of 200 nodes has to be split into 100 nodes each, then it is harder to solve such a problem. Consider problem instances where the number of machines is the same. When the number of jobs in each split is varied, the solution time and gap increases with increasing number of jobs when the number of machines is the same. For example, a 120 node graph and a 100 node graph both require 2 splits. So, in that case, the 120-node graph is harder to split than the 100-node graph, when the number of machines is the same. So, the main contributors to solution gap is the number of jobs in each split and the skewness of the splits. The solution time is also dependent on the number of times the splitting algorithm is run. For example, when there are 550 nodes, there is a need for 11 splits, which means the splitting algorithm is implemented 10 times. Given that the solution time is the total time needed to run the algorithm for all of the splits, the solution time increases with increasing number of nodes (generally).

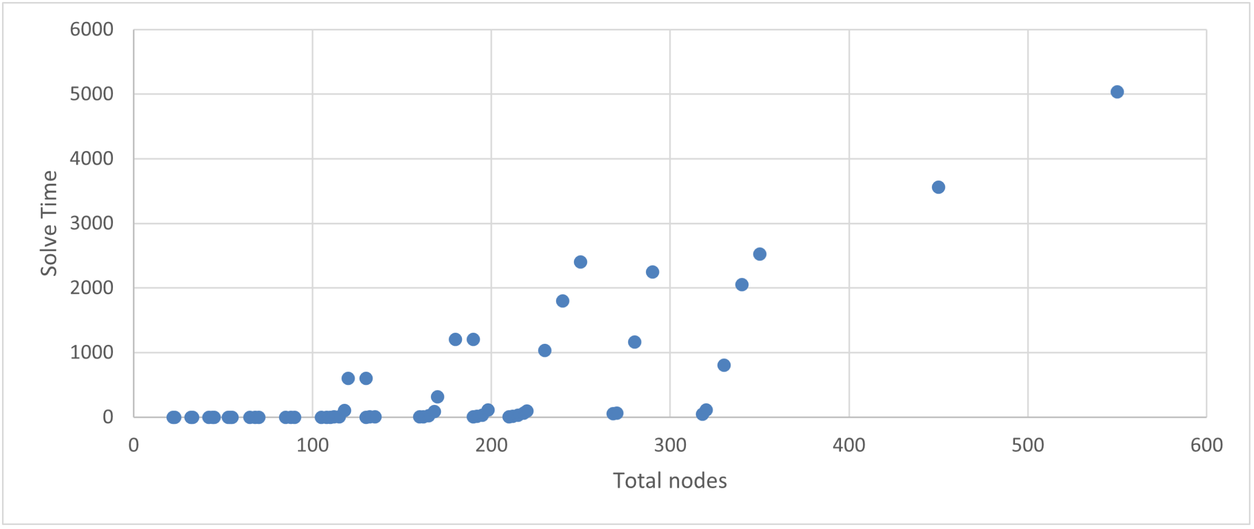

Figure 1: PSR Solution time vs number of nodes

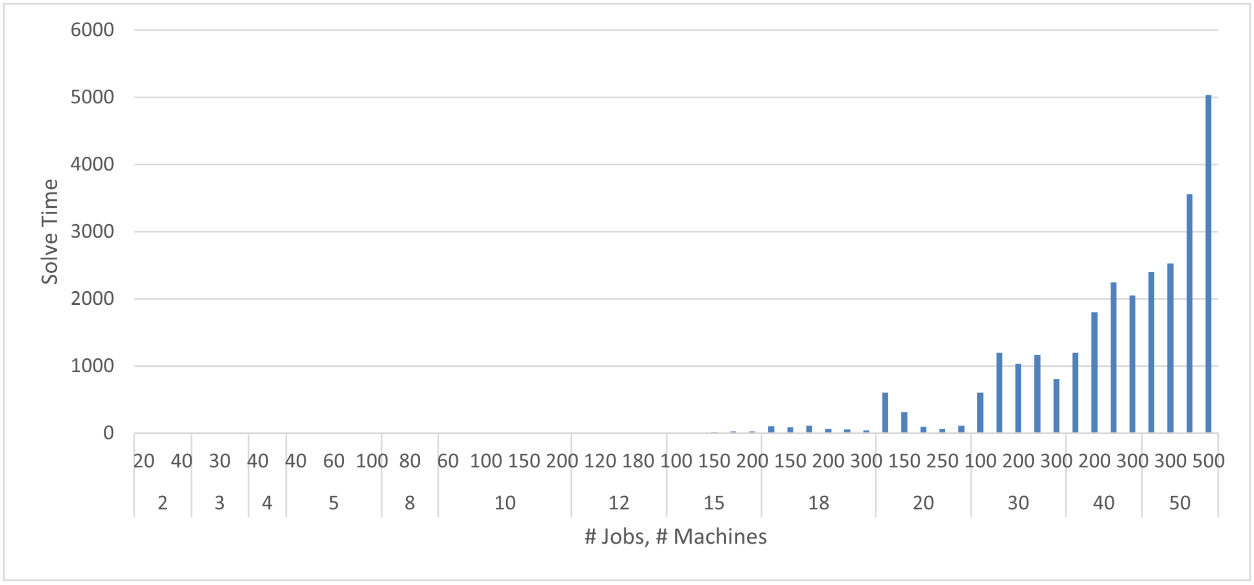

Figure 2: PSR Solution time vs number of machines, number of jobs

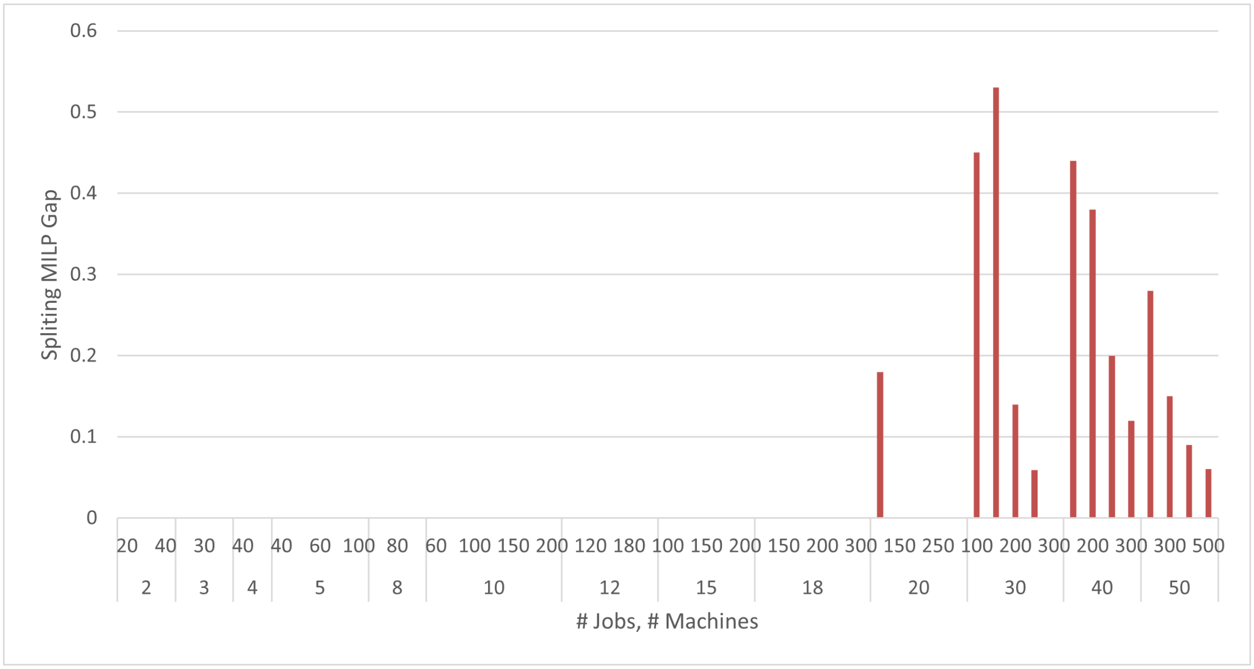

Figure 3: PSR Splitting MILP optimality gap vs number of machines, number of jobs

8. Comparison of Construction Heuristics

In this section, we compare the construction heuristics and PSR among themselves as well as with respect to the optimal solutions obtained from the MILP on all five objectives.

8.1. Design of Experiments

In order to understand how well a particular heuristic or the problem size reduction (PSR) method performs with respect to an objective, it is important to obtain optimal solutions (if possible) and compare them against the solutions obtained by the heuristic. Our MILP (OPS) is able to handle problems with up to 50 jobs and 5 machines. In the first set of experiments, we use small problems and solve them to optimality. Then the same problems are also solved by the heuristics and the solutions are compared for optimality gap.

To test the PSR method and its performance with respect to optimal values on all the objectives, the smaller sized problems were broken up into 2, 3, and 4 groups of equal size, each group solved to optimality and combined to form a single schedule. For example, if there are 3 machines and 20 jobs, then there are 23 nodes. If dividing this problem into two groups, then there would be 12 nodes in one group and 11 in the other group. These two groups would then be solved to optimality individually.

For this set of experiments, we test configurations with 2, 3, 4, 5 machines and 10, 20, 30, 40, 50 jobs. The colors, shades, due dates, processing times, machine eligibility and setup times between the jobs are randomly assigned. The color is randomly selected among 10 colors. Each color has a range of possible shades, and the shade is selected from that range randomly with same probability. The due dates are randomly selected using a uniform distribution with a range of [-1 week, +3 weeks]. The processing times are randomly selected using a uniform distribution with a range of [3 hours, 15 hours]. Machine eligibility is selected based on each machine having equal probability of being able to or not being able to process a job. Setup times between jobs are assigned based on their shades and colors and can take either 0, 5 or 10 hours. Each configuration (ex: 2 machines, 10 jobs) is tested for 10 different datasets to ensure consistency of the results. For each configuration, the gaps (percent above optimal) are averaged across the 10 datasets and the results are reported.

In the next set of experiments, the heuristics and the PSR method are compared among themselves on all five objectives for larger datasets. These large datasets contain 10, 20, 30, 40, 50 machines and 100, 200, 300, 400, 500 jobs. These problems are too large for the MILP to solve to optimality in a reasonable amount of time. Each configuration of the dataset has random due dates, processing times, machine eligibility, and setup matrices. Each configuration in this set also has 10 different datasets. The gap is calculated with respect to the best performing heuristic (percent above best solution) and the gaps are averaged across the 10 datasets.

Both sets of experiments are conducted on the 4 construction heuristics as well as the PSR method detailed in the previous section. However, for the construction heuristic H1, the list of jobs are sorted in 7 different ways: Shortest processing time (SPT), Longest Processing time (LPT), Least flexible job (LFJ), earliest due date (EDD), SPT+EDD, LPT+EDD, and LFJ+EDD. When a combination of sorting methods are used, the jobs are first grouped based on the processing times and within the groups, they are sorted by earliest due date. Since the jobs range between 1 hour and 15 hours of processing times, they are grouped into 1-5 hour jobs, 6-10 hour jobs and 10+ hour jobs. For SPT+EDD and LPT+EDD methods, this grouping is used. For LFJ+EDD, the ties for the flexibility are broken using due dates. For the rest of the heuristics, the sorting of the jobs has little effect. For the smaller sized problems, the MILP is solved for each objective separately to obtain the optimal solution for that dataset and objective. For larger sized problems, the PSR method is tested by dividing the large dataset into groups with each group containing a maximum of 60 nodes. For example, if the dataset has 400 jobs and 20 machines, it would be divided into 7 groups of 60 nodes (jobs/ machines) each, then each group would be solved to optimality, then combined to form a single schedule. This schedule is then compared with the rest of the heuristics for each objective performance.

8.2. Comparison with respect to the optimal solutions

In this subsection, we consider the results of the first set of experiments with respect to each objective. The results are compared for optimality gaps with respect to the optimal solutions obtained from the MILP.

8.2.1. Makespan objective

For the makespan objective, the objective grid (Figure A.4) indicates that the PSR-2 split method seems to be the better performing method as its values appear closer to the optimal value. However, within the heuristics, it is not clear as to which method is better, as the values seem close to each other. When a Kruskal Wallis test was conducted on the gaps, the p-value was less than 0.05, which means that at least one of the methods is different than the others. When we look at the box plots (Figure A.5), it is evident that the PSR-2 split has the best average gap, followed by LFJ methods, H2, H3 and LPT methods. The range of values for PSR 2 split is slightly large though. This is due to some cases when the splitting of the datasets resulted in skewed sub-dataset formation. For example, if there were 5 machines and 50 jobs, there was a chance of one group having 1 machine and 25 jobs and another group having 25 jobs and 4 machines.

8.2.2. Lateness objective

For the lateness objective, objective grid (Figure A.6) indicates that EDD and PSR are better than the rest of the heuristics. This is also true on the box plots (Figure A.7). When EDD was incorporated as a secondary sort on the list of jobs, it improved the gap slightly on the lateness objective. These conclusions are expected due to EDD's strength related to the lateness objective. With the flavor of optimality in the PSR method, we expect good results as well on all objectives. However, it is interesting to note that even EDD was 67% worse on average when compared to the optimal results. Also, the difference between EDD and other methods was at most 9%.

8.2.3. Setup ratio objective

For the setup objective, the objective grid (Figure A.8) show that the PSR methods perform significantly better than the rest. This is also true on the box plots (Figure A.9) where the gaps on these methods is significantly lower. For the PSR-2 split method, the average gap with respect to the optimal results is only 6%.

Among the heuristics, the H2, H3 and H4 methods perform better. Considering that these methods were developed to improve this objective along with the preference and shade consistency, this result is expected. Also, within the algorithm, the scoring function has a setup component which tries to minimize setups.

However, even on this objective, the difference between the best method (H2) and the optimal solutions is large (about 48%).

8.2.4. Color preference objective

For the color preference objective too, the PSR method is the best followed by the methods H2, H3 and H4. This is evidenced by the objective grid and the box plots (Figures A.10 and A.11). PSR methods benefit heavily by the use of optimal methods within them. The only loss of optimality is due to the removal of some edges from the machine eligibility graph.

H4 is the best performing heuristic for this objective and the gap with respect to the optimal is on average only 13%. This is due to the binning approach followed by H4 which allows for similar jobs to stay together on the same machine.

8.2.5. Shade consistency objective

For this objective, the PSR methods are actually worse when compared to the rest of the objectives as seen in figures A.12 and A.13. The rest of the heuristics have a gap of about 16% with respect to the MILP. There is almost no difference among the heuristics for this objective. This objective comes into greater light for larger datasets with more jobs on each machine.

8.3. Comparison between the heuristics on large datasets

In this subsection, we consider the results of the second set of experiments with respect to each objective. The results are compared for gaps with respect to the best solutions found from among the different heuristic methods.

8.3.1. Makespan objective

On the makespan objective, it is hard to differentiate between the methods from the objective grid A.14). However, it is interesting to note the few cases where PSR has performed terribly. This is again attributed to the few cases where the grouping was imbalanced. Such imbalance is more likely when there are fewer machines and more jobs. To tackle this imbalance in the future, the PSR method can be made more robust to include a better balance of machines and jobs within each group.

LPT performed the best among the heuristic methods as seen in the box plots (Figure A.15). The difference between the methods is small however, with the worst performer (PSR) only about 19% worse than the best. Among the heuristics, the worst performing is H4 with about 12% gap. This may happen because each method has some mechanism to ensure that the machines are balanced. In H1, the machine which can finish the job first is picked whereas in H2 and H3, the machine with the least completion time is picked for receiving a job. On H4, the machine utilization values from H3 is used as guidance. This makes the methods perform similarly on this objective.

8.3.2. Lateness objective

On the lateness objective, once again, EDD performs the best as expected (Figures A.16 and A.17). PSR is a close second with an average gap of 11%. Combining EDD with other methods once again helped the objective, for example, the second best was SPT+EDD. Among the other methods, there was insignificant difference. The solutions from the non-EDD heuristics were about 35% worse when compared to the EDD solutions.

8.3.3. Setup ratio objective

On the setup objective, we see PSR dominate the other methods. This is evident in the objective grid (Figure A.18) as well as the box plot (Figure A.19).

Among the heuristics, H4 is by far the best. In most cases, H2 and H3 performed better than H1, but even they were outperformed by H4. Having a large number of jobs helps H4 decide better to reduce the setups on each machine. When there are fewer jobs, the number of possible options for H4 is smaller so the effect of the algorithm design is not seen. In the large datasets, however, with plenty of options to select from, H4 performs significantly better than the rest. For this objective, H4 performs about 50% better than the worst performing method (EDD). H2 and H3 perform about 20% better.

8.3.4. Color preference objective

On this objective also, PSR performs the best in all cases, closely followed by H4, with an average gap of 1.4%. H2 and H3 follow closely behind as can be seen in the objective grid and the box plots (Figures A.20 and A.21).

H1 heuristics are much worse on this objective. This is because of the placement rules embedded in H2, H3 and H4 which consider the preferences, setups and shade difference values when placing the jobs. These rules are not present in H1 heuristics. On this objective, H4 performs about 31% better than H1 and H2 and H3 perform about 26% better.

8.3.5. Shade consistency objective

On this objective also, PSR performs the best by a large margin (Figures A.22 and A.23). The closest heuristic is again H4 which is also 78% worse. The rest of the methods perform very poorly in comparison to PSR.

8.4. Summary of Heuristic Comparison and Managerial Insights

Overall, PSR performs similar to other heuristics with respect to the makespan objective. In some cases, due to imbalance in splitting, the resulting schedules are skewed. This can be improved with a few constraints ensuring a better mix in each group. PSR is also only very slightly worse than EDD on the lateness objective. But, with respect to the other three objectives, it is significantly better than the rest. When compared to the optimal values also, it is only performing poorly on the lateness objective. An average gap of about 17% is acceptable on the makespan objective and shade consistency objectives. On the preference and setup objectives, the gap is less than 6%. So, overall, PSR is the best method among the tested methods. However, it does not provide quick solutions. For very large problems with more than 400 jobs, it takes a long time to create the optimal split. This gives the heuristic methods an advantage over any optimal or semi-optimal methods.

Among the construction heuristics, H4 is only about 11% worse on the makespan, and definitely bad on the lateness objective, But the gains on the other objectives is significant for H4. Also, on the lateness objective, it only performs very badly when compared to EDD but when compared to the rest of the methods, it is similar. So, overall, H4 seems like the best method. But if makespan is slightly more important, H3 is also a good method, considering that it performs well on the setups and color preference objectives also. However, if lateness is the most important objective, no method beats EDD. But EDD performs worse on all other objectives. Also, it performs badly when compared to the optimal values. So, there is work that has to be done to tackle this objective.

Table 8.4 shows a summary of the methods and on which objectives, they perform well. An 'x' on the table suggests that the method performs well and an 'o' suggests that it does not. From the table it is clear that on the objectives alone, PSR is the best method overall. But it suffers from poorer computation times. H4 is a fast heuristic which performs well on all objectives and only slightly worse than the rest on lateness. For lateness, EDD is the best objective. LPT and H3 are the best for makespan.

Method	Makespan	Lateness	Setups	Color	Shade	Computation
H1-EDD	x	x	o	o	o	x
H1-LFJ	x	o	o	o	o	x
H1-LPT	x	o	o	o	o	x
H1-SPT	x	o	o	o	o	x
H1-LFJ+EDD	x	o	o	o	o	x
H1-LPT+EDD	x	o	o	o	o	x
H1-SPT+EDD	x	x	o	o	o	x
H2	x	o	x	x	o	x
H3	x	o	x	x	o	x
H4	x	o	x	x	x	x
PSR	x	x	x	x	x	o

Table 1: Summary of competitiveness of the methods with respect to the five objectives and computation time

9. Conclusions and Future Work

In this paper, we have addressed a scheduling problem with parallel machines, machine eligibility, sequence-dependent setup times and sequence-dependent constraints. The objectives important to this problem are to reduce the makespan, lateness and setup times (specifically). Also, to improve the quality of production in a fabric dying industry, we have introduced two new objectives: color preference to provide smooth transitions between color changes, and shade consistency to ensure smooth shade transitions and similar shades of colors running on the machines.

We have modeled the new objectives and required constraints using an MILP (OPS). OPS was found to produce optimal solutions only for small-sized problems because of the NP-Hard nature of the problem. So, we have developed an MILP-based heuristic (PSR) which first breaks down a large problem into smaller easily solvable problems and solves each broken-down problem using the MILP in a reasonable amount of time. The method which breaks down the larger problem aims to minimize the information loss while splitting the problem. This splitting method is also a graph partitioning MILP which is also an NP-Hard problem. But, for this problem, breaking down the larger problems does not take a significant amount of time. Of course, it cannot solve all sizes of problems. We have shown that for problems with greater than 400 nodes, the splitting method does suffer from the curse of dimensionality. For this reason and because our industrial sponsor was apprehensive about using MILPs or MILP-based methods, we chose to adapt/ develop multiple construction heuristics to generate schedules quickly. We tested simple dispatching rules like SPT, EDD etc, and slightly complicated insertion based and bin-packing based heuristics.

The methods were then subject to rigorous testing to compare them against each other as well as to compare them to optimal results. The testing was split into two stages: first, the methods were compared against optimal results for small-sized problems. Then, for larger problems, the methods were compared among themselves.

For the makespan objective, PSR was competitive but certainly not the best. The best methods for the makespan objective were H2, H3 (insertion based method), LPT and LFJ. PSR struggled at this objective due to some cases where the splitting did not provide an even balance of machines and jobs. H4 was the worst on this objective.

For the lateness objective, EDD was the best but even that was significantly worse than optimal methods. PSR was second-best. The other methods were all slightly worse. However, EDD was much worse than the other methods on the other objectives.

On the other three objectives, the usage of index based insertion methods and bin-packing based methods truly paid off. PSR was also a big winner on these objectives, especially for larger problems. PSR, H2, H3 and H4 outperformed the rest on these objectives. On each of these three objectives, PSR was the best followed by H4. H4 was closely followed by H3 and then H2.

So, if a recommendation has to be made with respect an objective, we would choose H3 if makespan is important, considering the benefits it provides on other objectives. If lateness is the primary objective, EDD is definitely the way to go. However, the losses on other objectives has to be kept in mind. When it comes to the other three objectives, H4 is the best method. However, if MILP-based methods are an option and given a little more time, PSR wins hands down as the overall best method. It is competitive on the makespan-objective for most cases. It is only second-best on the lateness objective and significantly better than the rest of the methods on the other three objectives.

In this paper, we have proposed an MILP for solving the problem but have not strove to improve the MILP to be able to solve slightly larger problems also. The first direction of future work in this research is to improve the MILP to reduce computation time and optimality gaps for larger problems. The second direction is to develop heuristics to improve the lateness objective, which is the worst performing objective. Finally, while we have used index based methods to capture the multi-objective nature of the problem, we have not tackled this problem in a truly multi-objective sense. So, a large portion of future work will be dedicated towards developing/ adapting multi-objective methods such as constraint programming, multi-objective meta-heuristics, etc.

Appendix A. Objective function graphs and box plots for gap

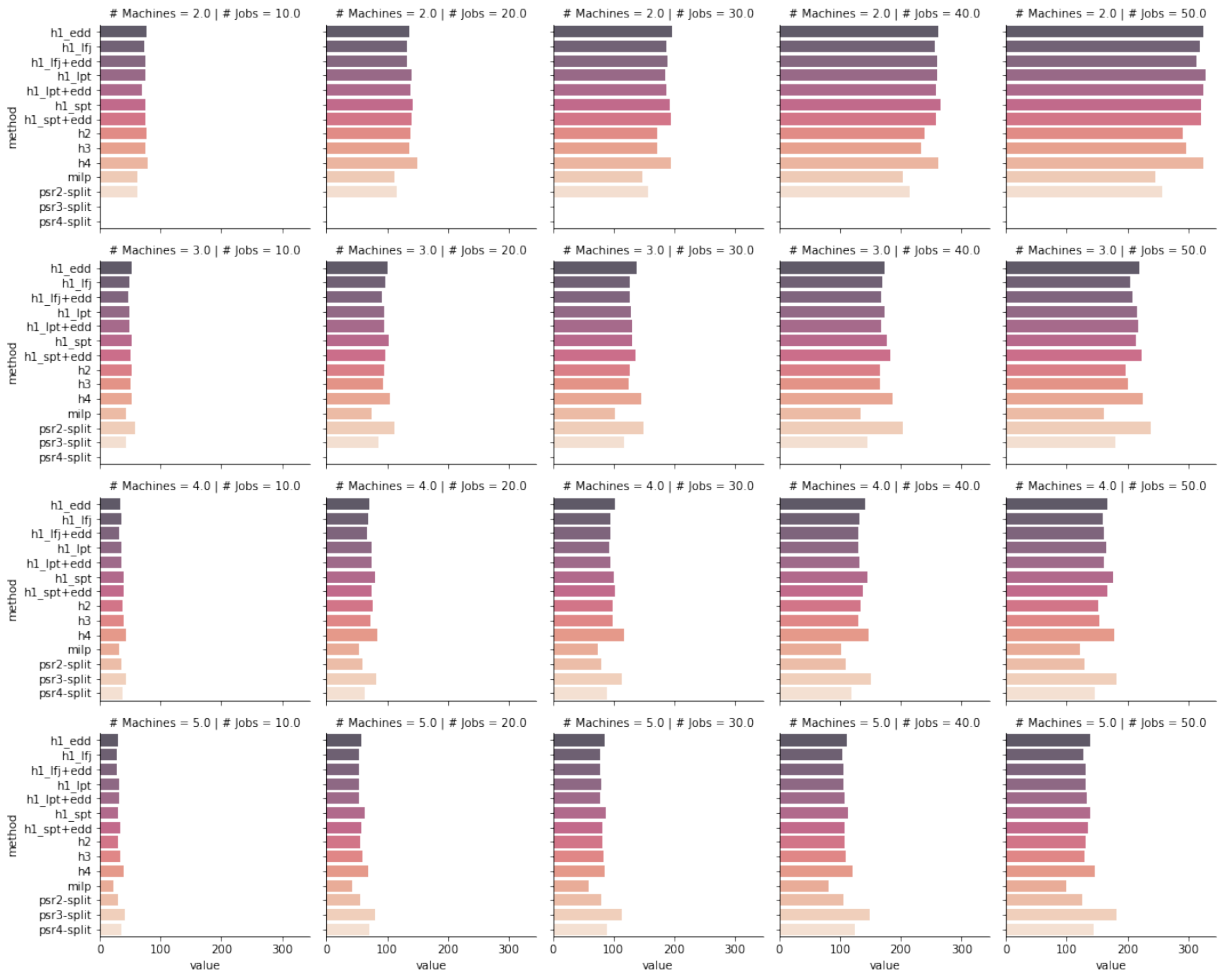

Figure A.4: Makespan small datasets values

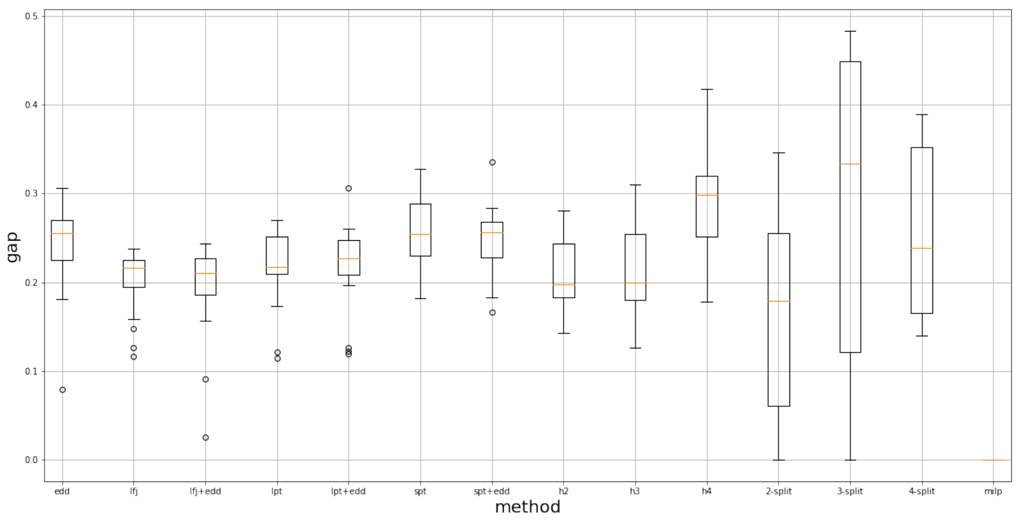

Figure A.5: Makespan small datasets box-plots for gap

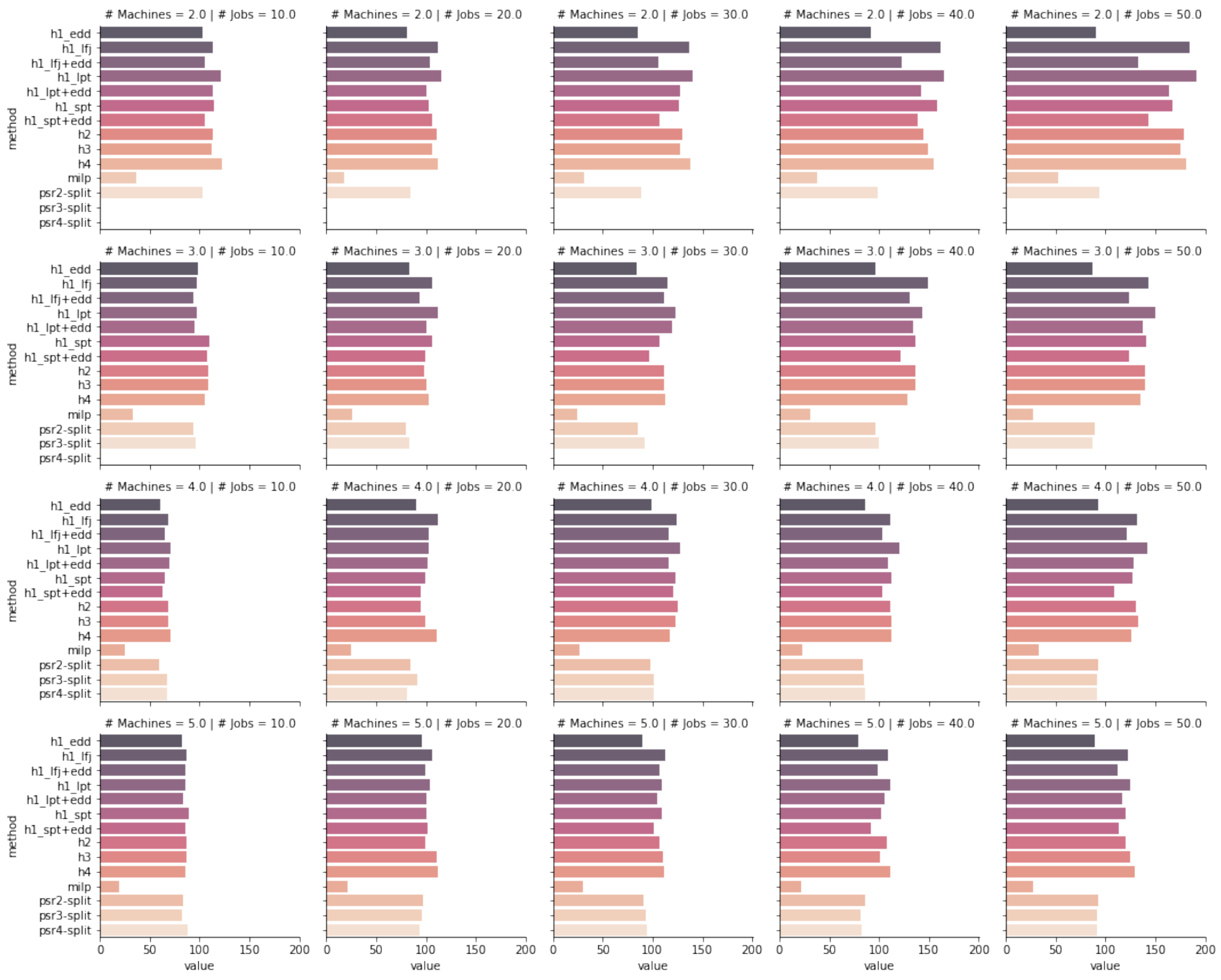

Figure A.6: Lateness small datasets values

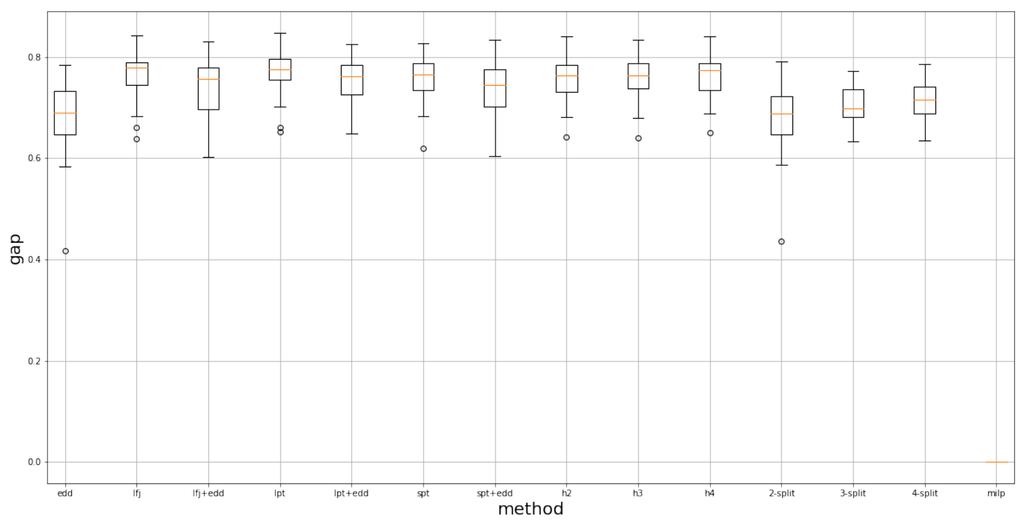

Figure A.7: Lateness small datasets box-plots for gap

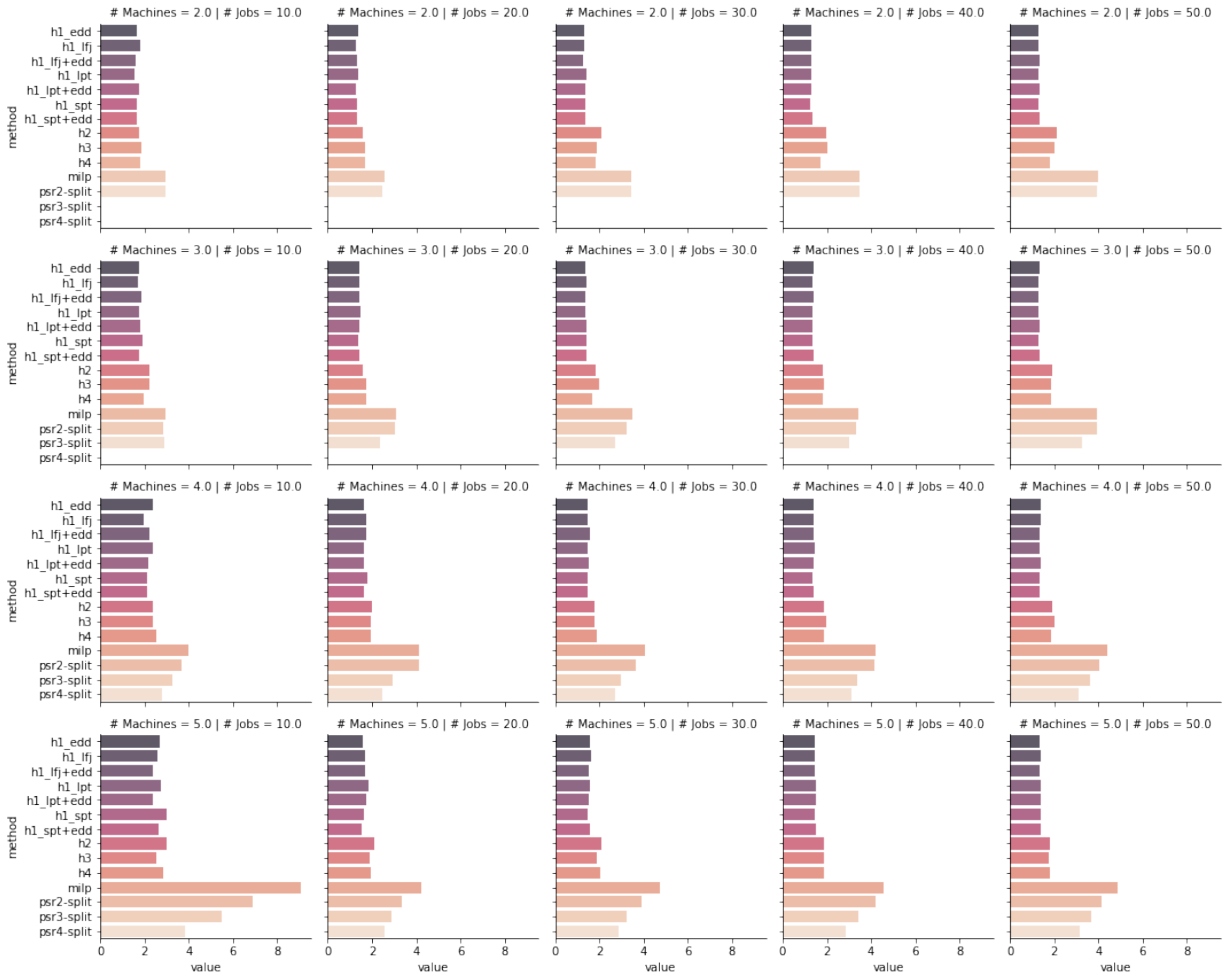

Figure A.8: Setup ratio small datasets values

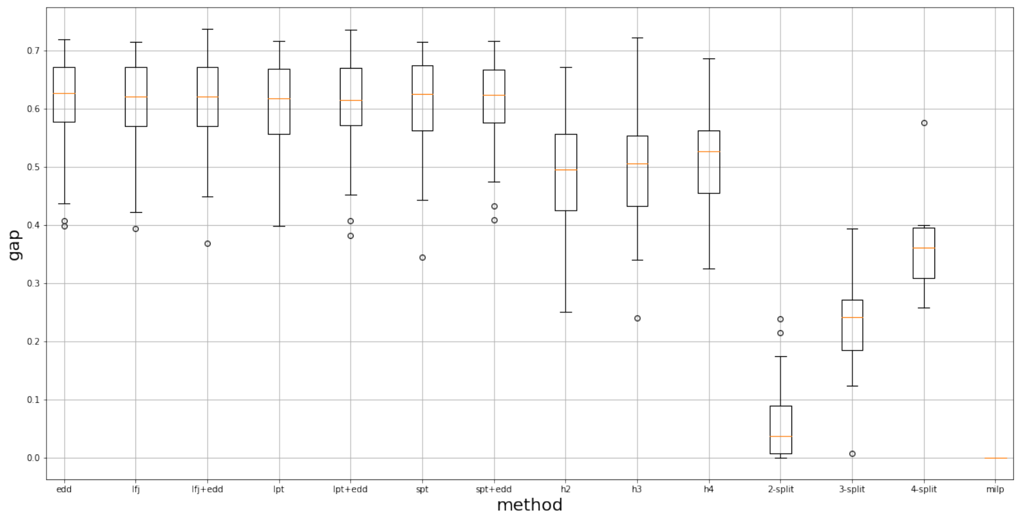

Figure A.9: Setup ratio small datasets box-plots for gap

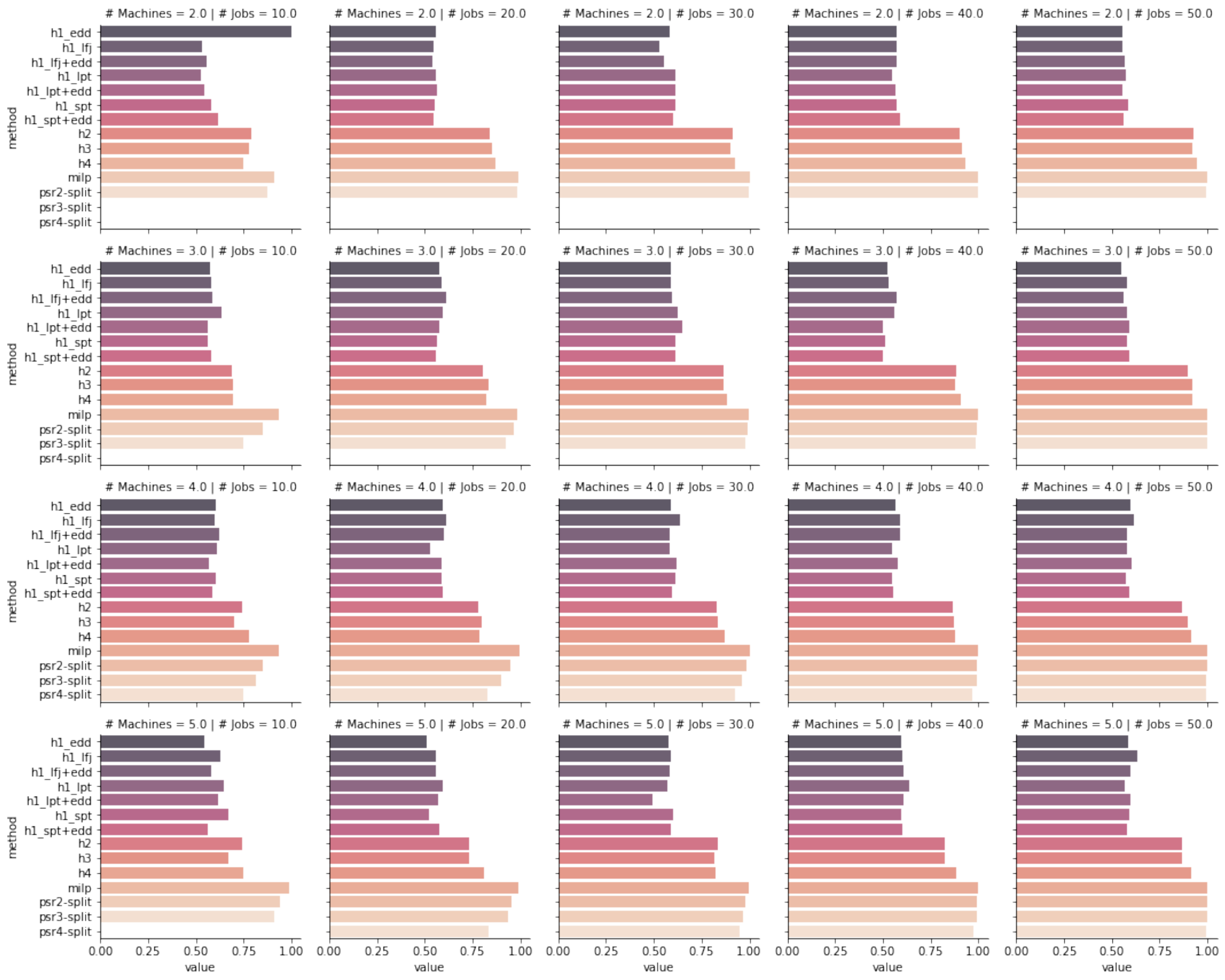

Figure A.10: Color preference small datasets values

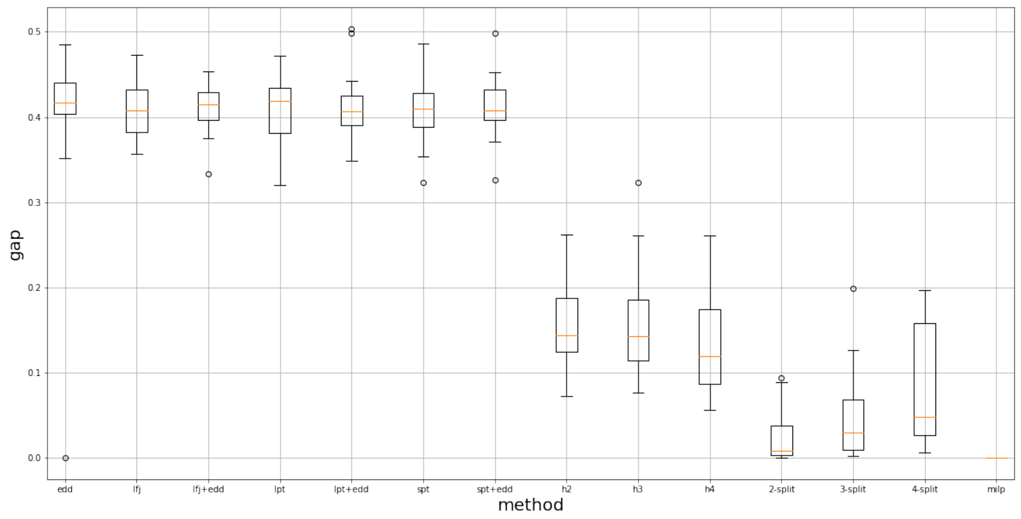

Figure A.11: Color preference small datasets box-plots for gap

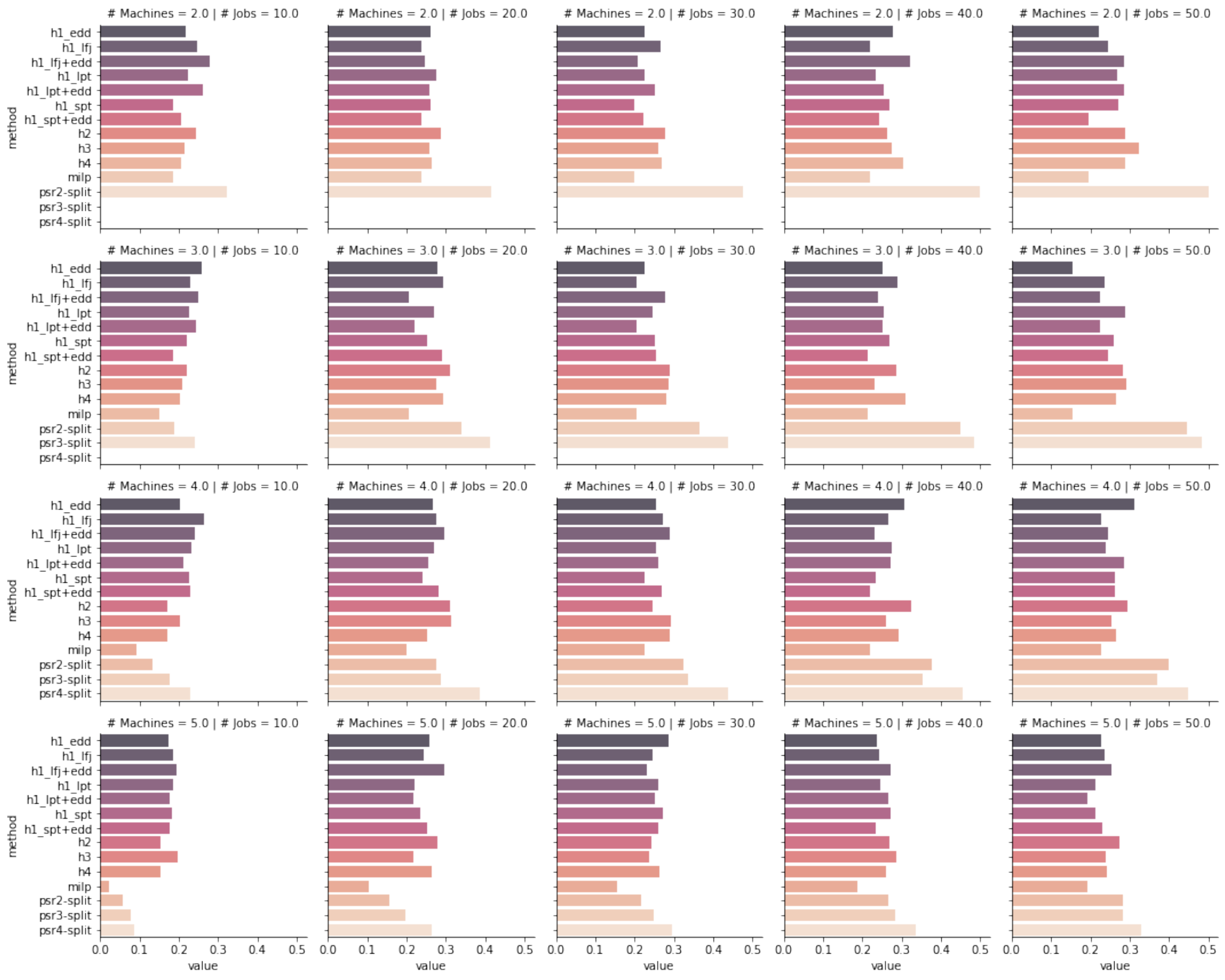

Figure A.12: Shade consistency small datasets values

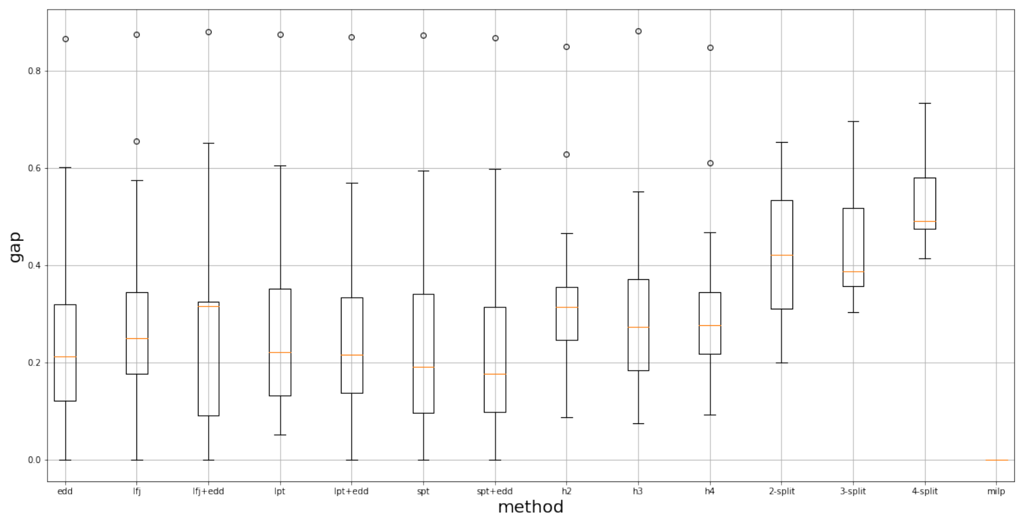

Figure A.13: Shade consistency small datasets box-plots for gap

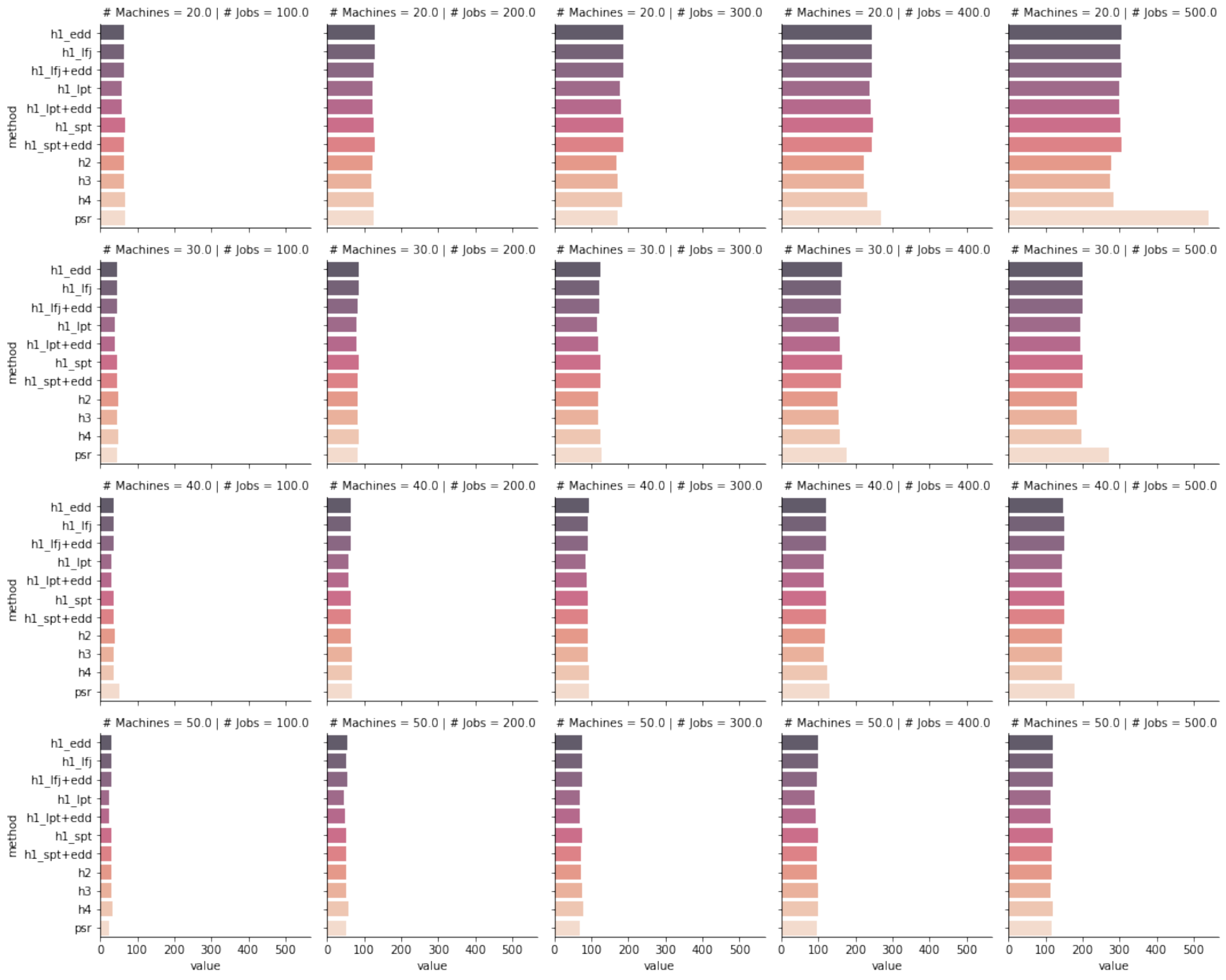

Figure A.14: Makespan large datasets values

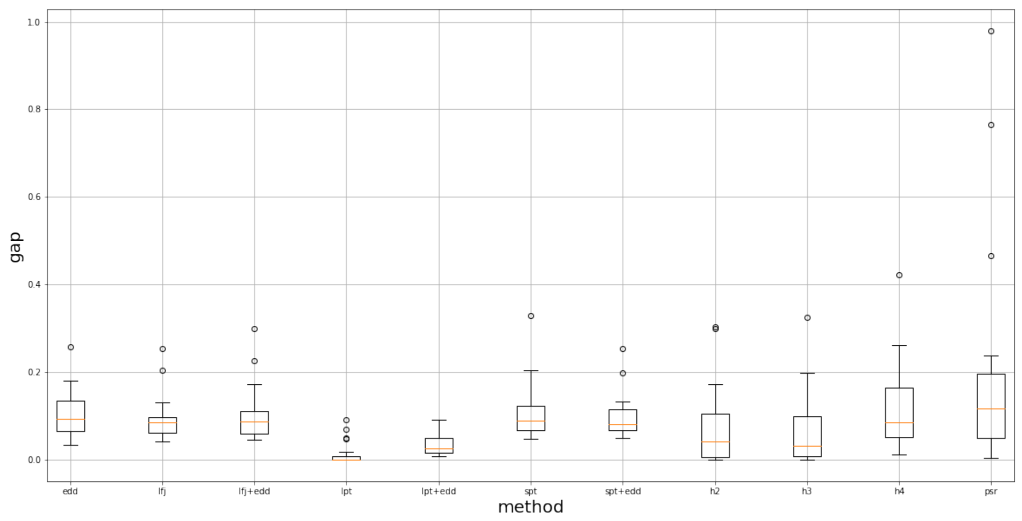

Figure A.15: Makespan large datasets box-plots for gap

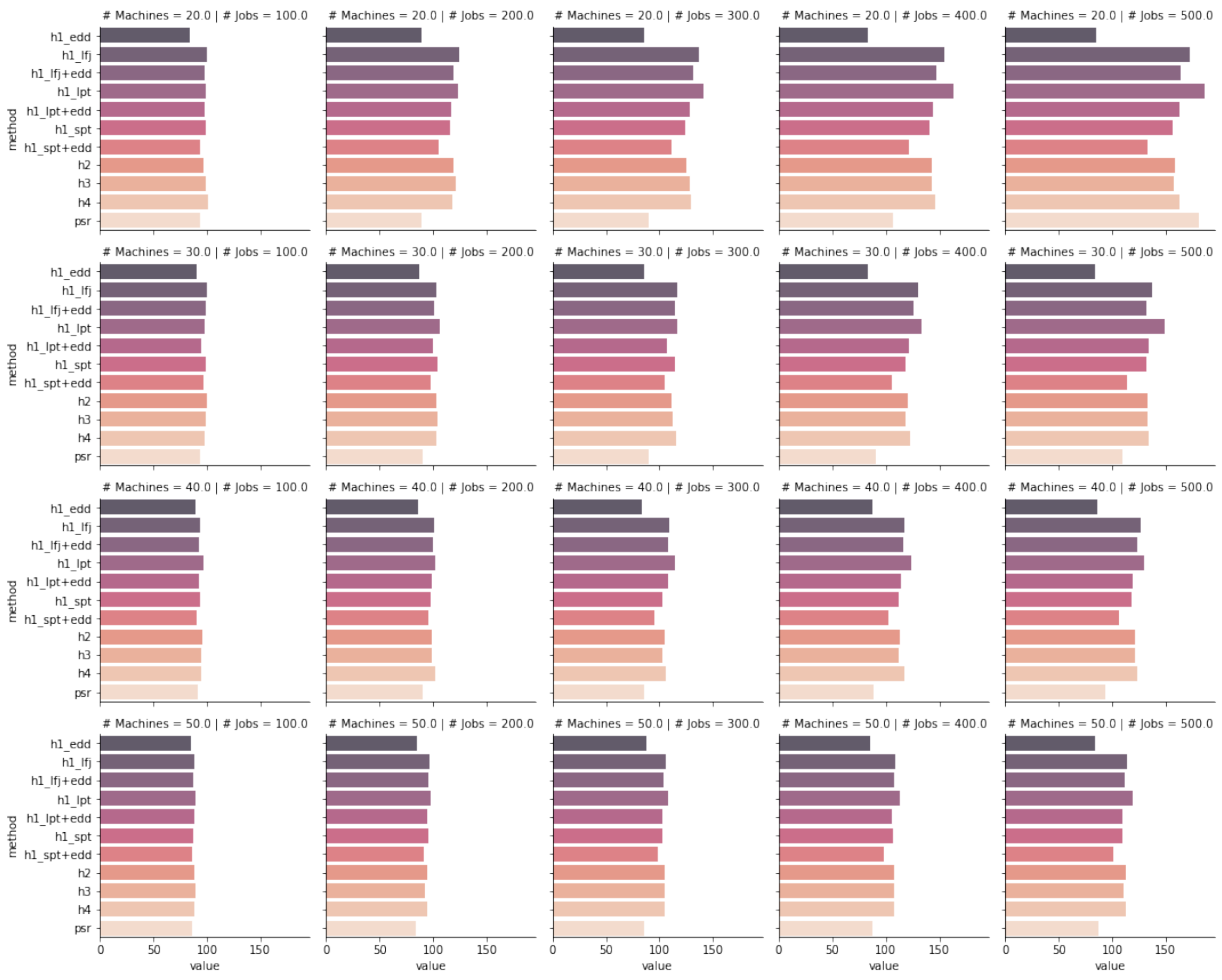

Figure A.16: Lateness large datasets values

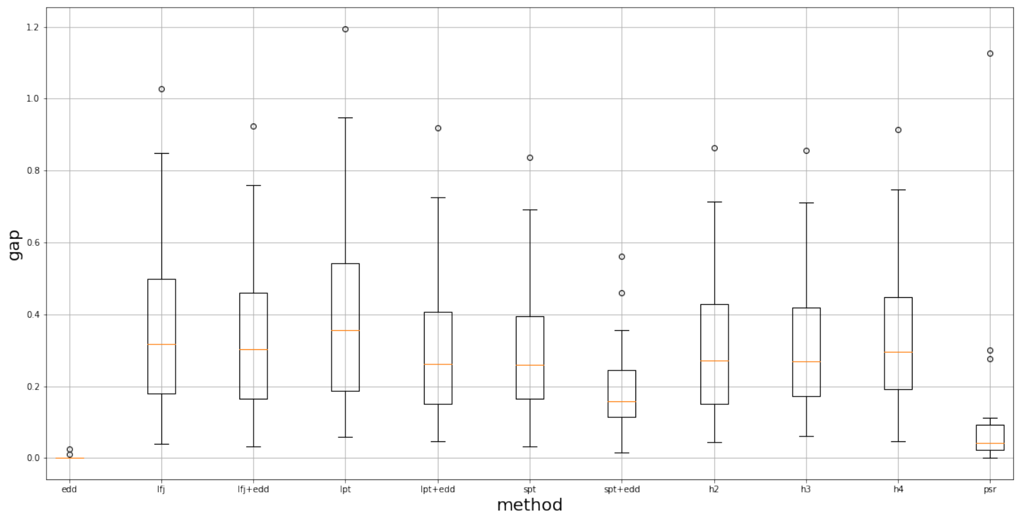

Figure A.17: Lateness large datasets box-plots for gap

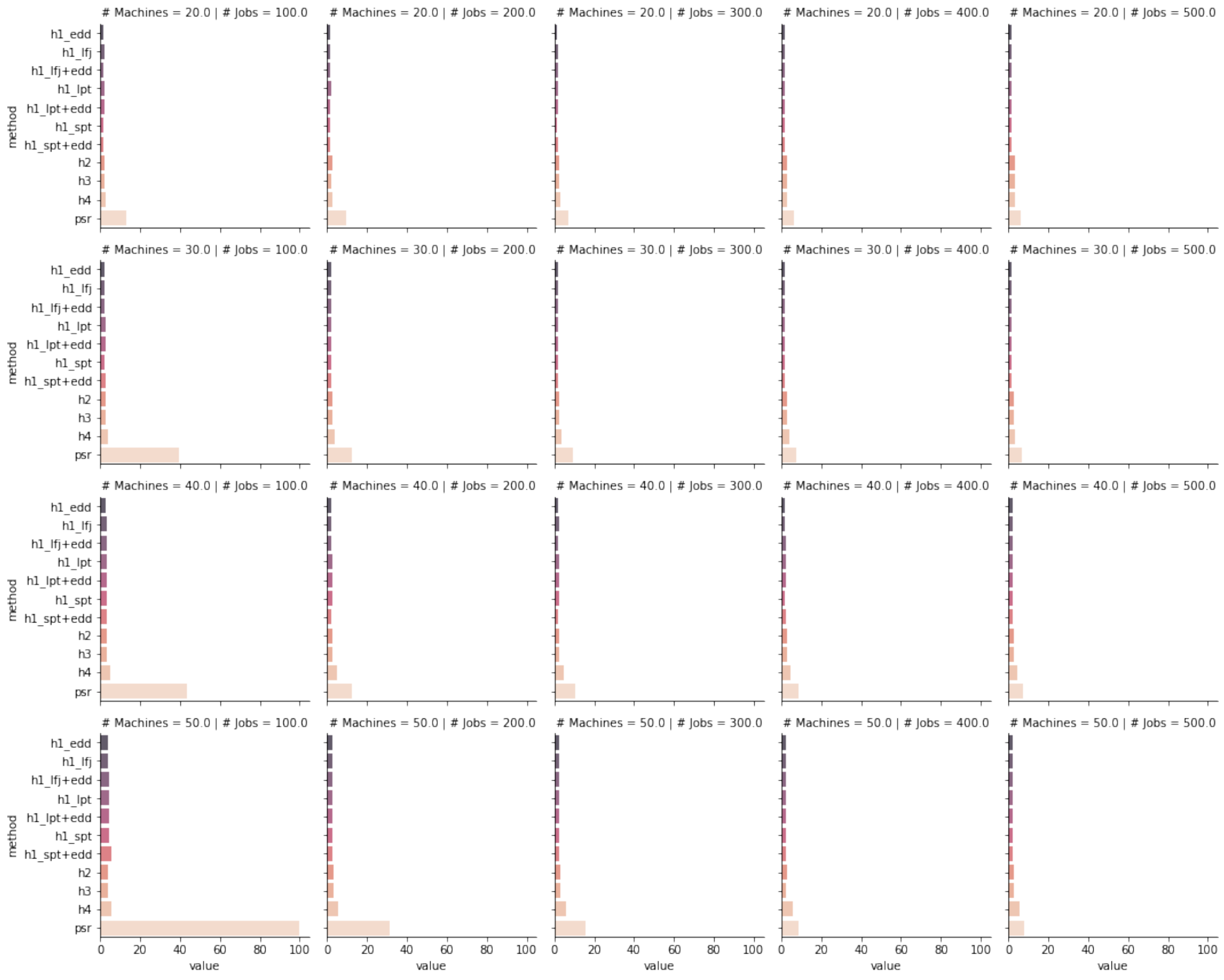

Figure A.18: Setup ratio large datasets values

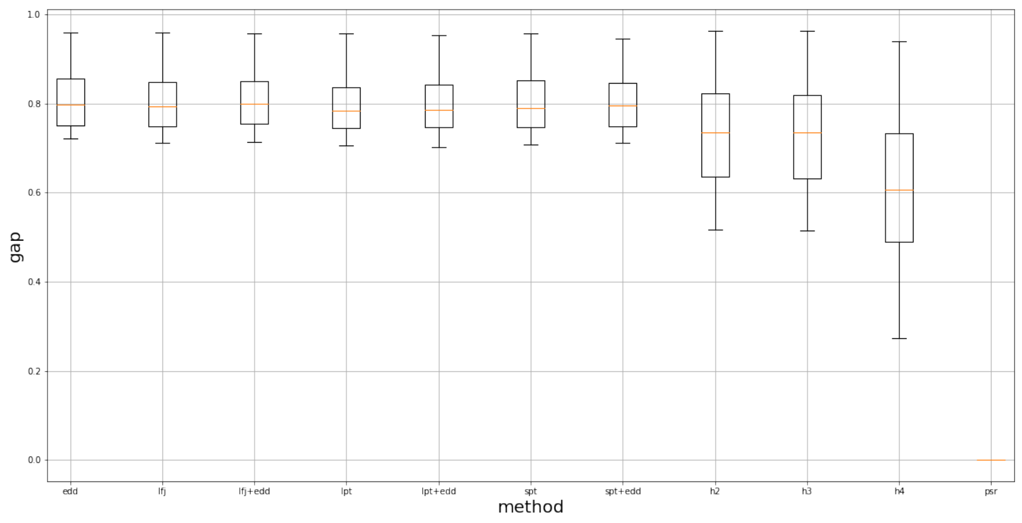

Figure A.19: Setup ratio large datasets box-plots for gap

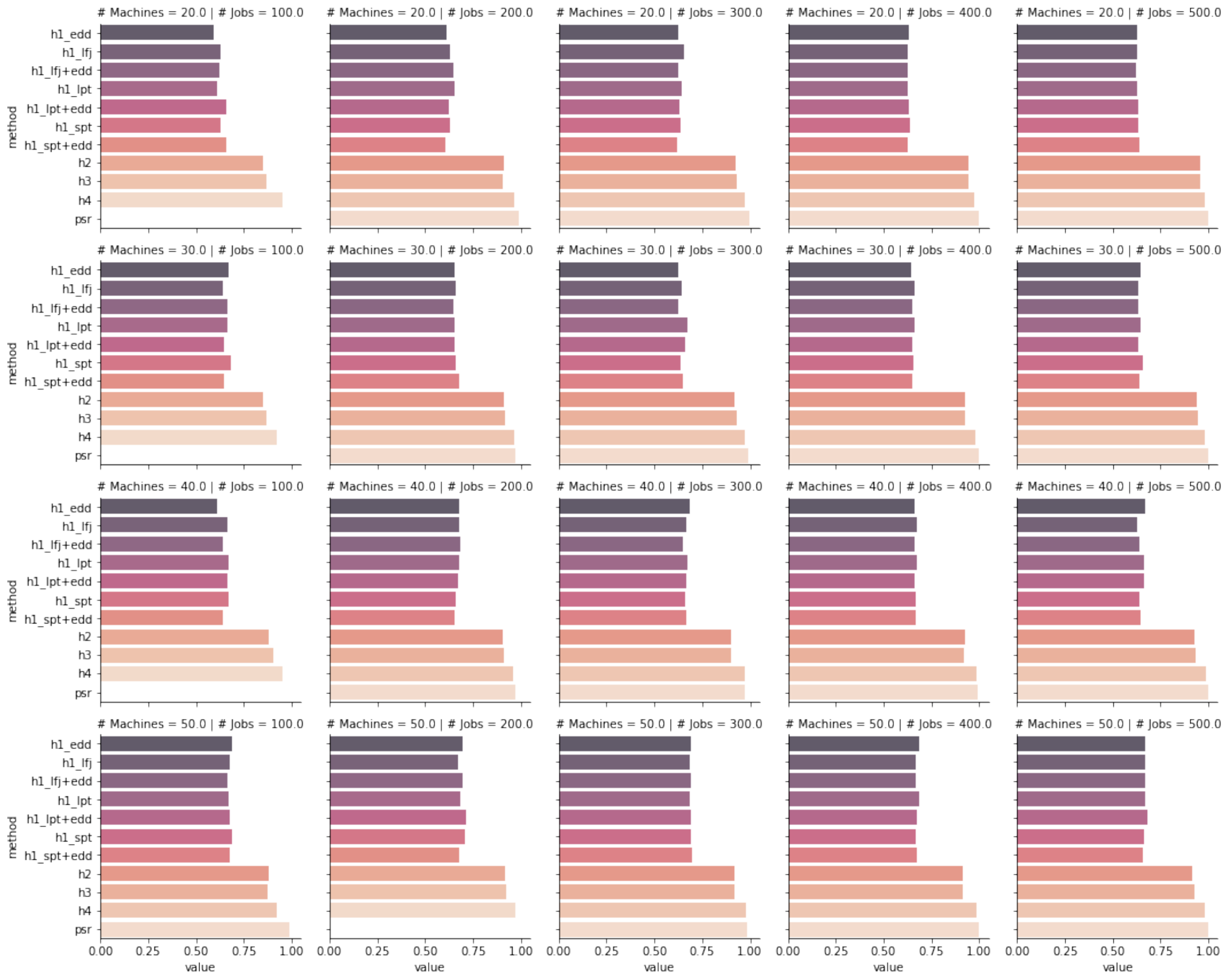

Figure A.20: Color preference large datasets values

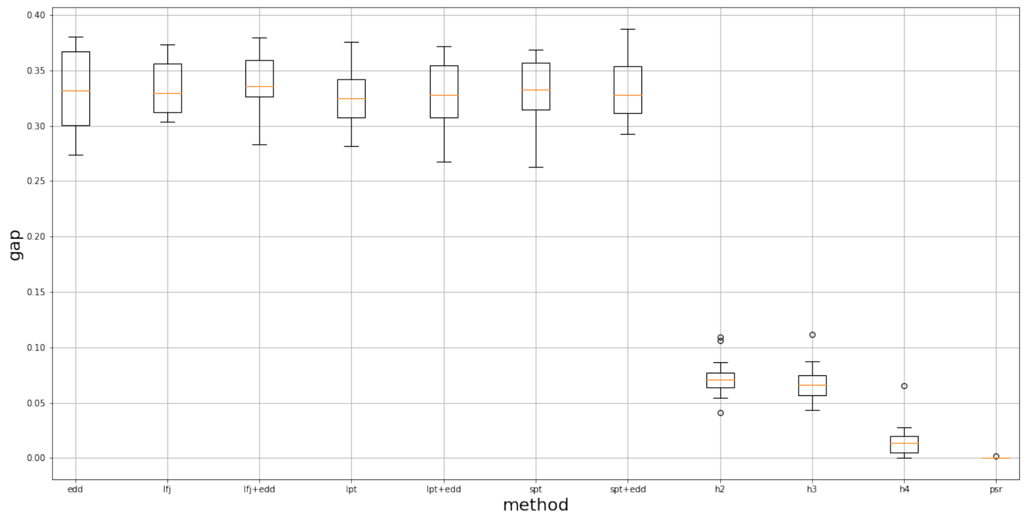

Figure A.21: Color preference large datasets box-plots for gap

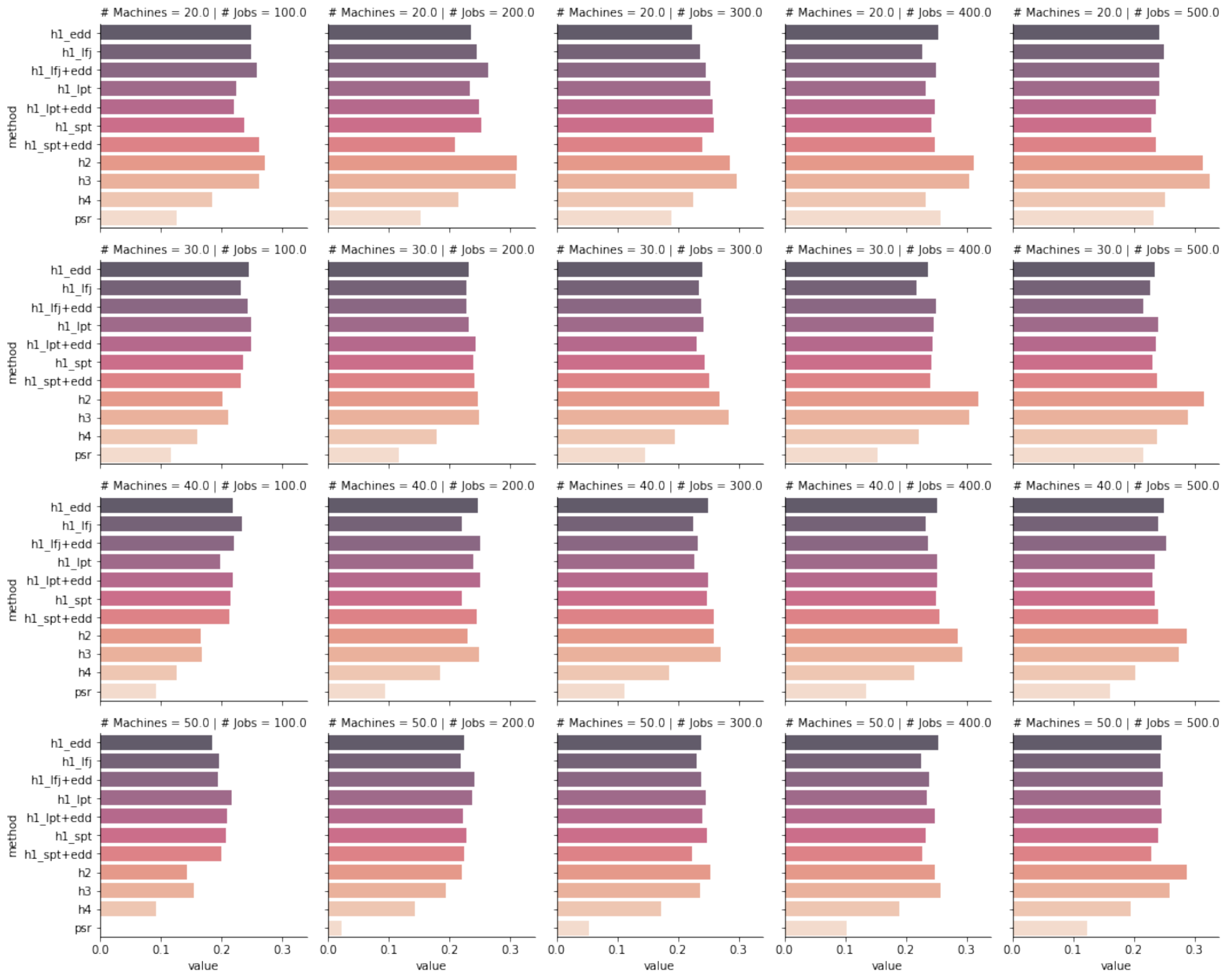

Figure A.22: Shade consistency large datasets values

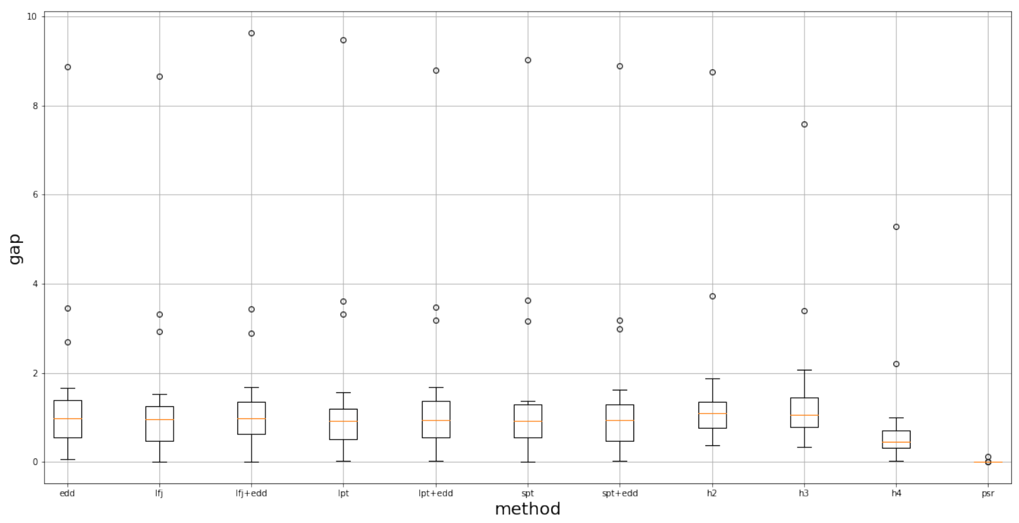

Figure A.23: Shade consistency large datasets box-plots for gap

References

[1] Hamid Abedinnia et al. “Machine scheduling problems in production: A tertiary study”. In: *Computers & Industrial Engineering* 111 (2017), pp. 403–416.

[2] Mojtaba Afzalirad and Javad Rezaeian. “Resource-constrained unrelated parallel machine scheduling problem with sequence dependent setup times, precedence constraints and machine eligibility restrictions”. In: *Computers & Industrial Engineering* 98 (2016), pp. 40–52.

[3] Derya Eren Akyol and G Mirac Bayhan. “Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach”. In: *The International Journal of Advanced Manufacturing Technology* 37.5-6 (2008), pp. 576–588.

[4] Ali Allahverdi. “The third comprehensive survey on scheduling problems with setup times/costs”. In: *European Journal of Operational Research* 246.2 (2015), pp. 345–378.

[5] Bradley E Anderson et al. “An efficient network-based formulation for sequence dependent setup scheduling on parallel identical machines”. In: *Mathematical and Computer Modelling* 57.3-4 (2013), pp. 483–493.

[6] Jaime A Arango, Jaime A Giraldo, and Omar D Castrillon. “Scheduling of non-related parallel machines with sequence dependent setup times and dynamic entry using genetic algorithms”. In: *Información Tecnológica* 24 (2013), pp. 73–84. ISSN: 0718-0764.

[7] Oliver Avalos-Rosales, Francisco Angel-Bello, and Ada Alvarez. "Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times". In: *The International Journal of Advanced Manufacturing Technology* 76.9-12 (2015), pp. 1705–1718.

[8] M Aramon Bajestani and Reza Tavakkoli-Moghaddam. "A new branch-and-bound algorithm for the unrelated parallel machine scheduling problem with sequence-dependent setup times". In: *IFAC Proceedings Volumes* 42.4 (2009), pp. 792–797.

[9] Kenneth R Baker and JWM Bertrand. "A dynamic priority rule for scheduling against due-dates". In: *Journal of Operations Management* 3.1 (1982), pp. 37–42.

[10] Natalia P Basán et al. "Scheduling of flexible manufacturing plants with redesign options: A MILP-based decomposition algorithm and case studies". In: *Computers & Chemical Engineering* 136 (2020), p. 106777.

[11] Ann Melissa Campbell and Martin Savelsbergh. "Efficient insertion heuristics for vehicle routing and scheduling problems". In: *Transportation science* 38.3 (2004), pp. 369–378.

[12] Donald C Carroll. "Heuristic sequencing of single and multiple component jobs." PhD thesis. Massachusetts Institute of Technology, 1965.

[13] Jeng-Fung Chen. "Scheduling on unrelated parallel machines with sequence-and machine-dependent setup times and due-date constraints". In: *The International Journal of Advanced Manufacturing Technology* 44.11-12 (2009), pp. 1204–1212.

[14] Edward G Coffman Jr, Michael R Garey, and David S Johnson. "An application of bin-packing to multiprocessor scheduling". In: *SIAM Journal on Computing* 7.1 (1978), pp. 1–17.

[15] Mohammad I Daoud and Nawwaf Kharma. "Efficient compile-time task scheduling for heterogeneous distributed computing systems". In: *12th International Conference on Parallel and Distributed Systems-(ICPADS'06)*. Vol. 1. IEEE. 2006, 9–pp.

[16] Thanh-Cong Dinh and Hyerim Bae. "Parallel servers scheduling with dynamic sequence-dependent setup time". In: *Intelligent Decision Technologies*. Springer, 2012, pp. 79–87.

[17] Baoqiang Fan and Guochun Tang. "A column generation for a parallel machine scheduling with sequence-dependent setup times". In: *Tongji Daxue Xuebao/ Journal of Tongji University(Natural Science)* 34.5 (2006), pp. 680–683.

[18] Bernat Gacias, Christian Artigues, and Pierre Lopez. "Parallel machine scheduling with precedence constraints and setup times". In: *Computers & Operations Research* 37.12 (2010), pp. 2141–2151.

[19] Sara Gestrelius, Martin Aronsson, and Anders Peterson. “A MILP-based heuristic for a commercial train timetabling problem”. In: *Transportation Research Procedia* 27 (2017), pp. 569–576.

[20] AH Gharehgozli, R Tavakkoli-Moghaddam, and N Zaerpour. “A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates”. In: *Robotics and Computer-Integrated Manufacturing* 25.4-5 (2009), pp. 853–859.

[21] Ravindra Gokhale and M Mathirajan. “Scheduling identical parallel machines with machine eligibility restrictions to minimize total weighted flowtime in automobile gear manufacturing”. In: *The International Journal of Advanced Manufacturing Technology* 60.9-12 (2012), pp. 1099–1110.

[22] Alper Hamzadayi and Gokalp Yildiz. “Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times”. In: *Computers & Industrial Engineering* 106 (2017), pp. 287–298.

[23] Zheng-liang Hou and Xiu-ping Guo. “Parallel machine scheduling with resources constraint and sequence dependent setup times”. In: *Proceedings of 2012 3rd International Asia Conference on Industrial Engineering and Management Innovation (IEMI2012)*. Springer. 2013, pp. 801–811.

[24] Dayong Hu and Zhenqiang Yao. “Identical parallel machines scheduling with sequence-dependent setup times”. In: *Jixie Gongcheng Xuebao(Chinese Journal of Mechanical Engineering)* 47.16 (2011), pp. 160–165.

[25] Dayong Hu and Zhenqiang Yao. “Parallel machines scheduling with sequence-dependent setup times constraints”. In: *Advanced Science Letters* 4.6-7 (2011), pp. 2528–2531.

[26] Xiaoyun Ji. *Graph partition problems with minimum size constraints*. Rensselaer Polytechnic Institute, 2004.

[27] Cheol Min Joo and Byung Soo Kim. “Parallel machine scheduling problem with ready times, due times and sequence-dependent setup times using meta-heuristic algorithms”. In: *Engineering Optimization* 44.9 (2012), pp. 1021–1034.

[28] Edmar Hell Kampke, José Elias Claudio Arroyo, and André Gustavo Santos. “Iterated local search with path relinking for solving parallel machines scheduling problem with resource-assignable sequence dependent setup times”. In: *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer. 2010, pp. 107–118.

[29] Yong Ha Kang and Hyun Joon Shin. “An adaptive scheduling algorithm for a parallel machine problem with rework processes”. In: *International Journal of Production Research* 48.1 (2010), pp. 95–115.

[30] Jae-Gon Kim, Seokwoo Song, and BongJoo Jeong. "Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times". In: *International Journal of Production Research* 0.0 (2019), pp. 1–16. DOI: 10.1080/00207543.2019.1672900. eprint: https://doi.org/10.1080/00207543.2019.1672900. URL: https://doi.org/10.1080/00207543.2019.1672900.

[31] Ji-Su Kim, Jung-Hyeon Park, and Dong-Ho Lee. "Iterated greedy algorithms to minimize the total family flow time for job-shop scheduling with job families and sequence-dependent set-ups". In: *Engineering Optimization* 49.10 (2017), pp. 1719–1732.

[32] ME Kurz and RG Askin. "Heuristic scheduling of parallel machines with sequence-dependent set-up times". In: *International journal of production research* 39.16 (2001), pp. 3747–3769.

[33] Young Hoon Lee, Kumar Bhaskaran, and Michael Pinedo. "A heuristic to minimize the total weighted tardiness with sequence-dependent setups". In: *IIE transactions* 29.1 (1997), pp. 45–52.

[34] Young Hoon Lee and Michael Pinedo. "Scheduling jobs on parallel machines with sequence-dependent setup times". In: *European Journal of Operational Research* 100.3 (1997), pp. 464–474.

[35] Xiaohui Li et al. "Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem". In: *Journal of Intelligent Manufacturing* 23.4 (2012), pp. 1179–1194.

[36] Ching-Jong Liao, Cheng-Hsiung Lee, and Hsing-Tzu Tsai. "Scheduling with multi-attribute set-up times on unrelated parallel machines". In: *International Journal of Production Research* 54.16 (2016), pp. 4839–4853.

[37] Ching-Jong Liao, Yu-Lun Tsai, and Chien-Wen Chao. "An ant colony optimization algorithm for setup coordination in a two-stage production system". In: *Applied Soft Computing* 11.8 (2011), pp. 4521–4529.

[38] Shih-Wei Lin, Chung-Cheng Lu, and Kuo-Ching Ying. "Minimization of total tardiness on unrelated parallel machines with sequence-and machine-dependent setup times under due date constraints". In: *The International Journal of Advanced Manufacturing Technology* 53.1-4 (2011), pp. 353–361.

[39] Yang-Kuei Lin and Feng-Yu Hsieh. "Unrelated parallel machine scheduling with setup times and ready times". In: *International Journal of Production Research* 52.4 (2014), pp. 1200–1214.

[40] Rasaratnam Logendran, Brent McDonell, and Byran Smucker. "Scheduling unrelated parallel machines with sequence-dependent setups". In: *Computers & Operations Research* 34.11 (2007), pp. 3420–3438.

[41] Quan Lu and Maged M Dessouky. “A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows”. In: *European Journal of Operational Research* 175.2 (2006), pp. 672–687.

[42] Mahdi Naderi-Beni et al. “Fuzzy bi-objective formulation for a parallel machine scheduling problem with machine eligibility restrictions and sequence-dependent setup times”. In: *International Journal of Production Research* 52.19 (2014), pp. 5799–5822.

[43] Michael Pinedo. *Scheduling*. Vol. 5. Springer, 2012.

[44] Pedro Leite Rocha et al. “Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times”. In: *Computers & Operations Research* 35.4 (2008), pp. 1250–1264.

[45] Rubén Ruiz and Carlos Andrés-Romano. “Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times”. In: *The International Journal of Advanced Manufacturing Technology* 57.5-8 (2011), pp. 777–794.

[46] Marcos Wagner Jesus Servare Junior et al. “A Linear Relaxation-Based Heuristic for Iron Ore Stockyard Energy Planning”. In: *Energies* 13.19 (2020), p. 5232.

[47] Pankaj Sharma and Ajai Jain. “A review on job shop scheduling with setup times”. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 230.3 (2016), pp. 517–533.

[48] Reza Tavakkoli-Moghaddam et al. “Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints”. In: *Computers & Operations Research* 36.12 (2009), pp. 3224–3230.

[49] M Duran Toksarı, Daniel Oron, and Ertan Güner. “Some scheduling problems with past sequence dependent setup times under the effects of nonlinear deterioration and time-dependent learning”. In: *RAIRO-Operations Research* 44.2 (2010), pp. 107–118.

[50] C Tsai and C Tseng. “Unrelated parallel-machines scheduling with constrained resources and sequence-dependent setup time”. In: *Proceedings of the 37th International Conference on Computers and Industrial Engineering, Alexandria, Egypt*. 2007, pp. 20–23.

[51] Shunji Umetani, Yuta Fukushima, and Hiroshi Morita. “A linear programming based heuristic algorithm for charge and discharge scheduling of electric vehicles in a building energy management system”. In: *Omega* 67 (2017), pp. 115–122.

[52] Ari PJ Vepsalainen and Thomas E Morton. “Priority rules for job shops with weighted tardiness costs”. In: *Management science* 33.8 (1987), pp. 1035–1047.

[53] I-Lin Wang, Yi-Chi Wang, and Chih-Wei Chen. “Scheduling unrelated parallel machines in semiconductor manufacturing by problem reduction and local search heuristics”. In: *Flexible Services and Manufacturing Journal* 25.3 (2013), pp. 343–366.

[54] Yue Xi and Jaejin Jang. “Scheduling jobs on identical parallel machines with unequal future ready time and sequence dependent setup: An experimental study”. In: *International Journal of Production Economics* 137.1 (2012), pp. 1–10.

[55] Kuo-Ching Ying and Hui-Miao Cheng. “Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic”. In: *Expert Systems with Applications* 37.4 (2010), pp. 2848–2852.

[56] YiZeng Zeng, Ada Che, and Xueqi Wu. “Bi-objective scheduling on uniform parallel machines considering electricity cost”. In: *Engineering Optimization* 50.1 (2018), pp. 19–36.

[57] Xiaoyan Zhu and Wilbert E Wilhelm. “Scheduling and lot sizing with sequence-dependent setup: A literature review”. In: *IIE Transactions* 38.11 (2006), pp. 987–1007.