Extensions of the Dual-Resource Constrained Flexible Job-Shop Scheduling Problem

by

Katherine B. Adams

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2019 by the
Graduate Supervisory Committee:

Jorge Sefair, Co-Chair
Ronald Askin, Co-Chair
Scott Webster

ARIZONA STATE UNIVERSITY

May 2019

ABSTRACT

The shift in focus of manufacturing systems to high-mix and low-volume production poses a challenge to both efficient scheduling of manufacturing operations and effective assessment of production capacity. This thesis considers the problem of scheduling a set of jobs that require machine and worker resources to complete their manufacturing operations. Although planners in manufacturing contexts typically focus solely on machines, schedules that only consider machining requirements may be problematic during implementation because machines need skilled workers and cannot run unsupervised. The model used in this research will be beneficial to these environments as planners would be able to determine more realistic assignments and operation sequences to minimize the total time required to complete all jobs. This thesis presents a mathematical formulation for concurrent scheduling of machines and workers that can optimally schedule a set of jobs while accounting for changeover times between operations. The mathematical formulation is based on disjunctive constraints that capture the conflict between operations when trying to schedule them to be performed by the same machine or worker. An additional formulation extends the previous one to consider how cross-training may impact the production capacity and, for a given budget, provide training recommendations for specific workers and operations to reduce the makespan. If training a worker is advantageous to increase production capacity, the model recommends the best time window to complete it such that overlaps with work assignments are avoided. It is assumed that workers can perform tasks involving the recently acquired skills as soon as training is complete. As an alternative to the mixed-integer programming formulations, this thesis provides a math-heuristic approach that fixes the order of some operations based on Largest Processing Time (LPT) and Shortest Processing Time (SPT) procedures, while allow-

ing the exact formulation to find the optimal schedule for the remaining operations. Computational experiments include the use of the solution for the no-training problem as a starting feasible solution to the training problem. Although the models provided are general, the manufacturing of Printed Circuit Boards are used as a case study.

DEDICATION

I dedicate this thesis to my family and friends for their continuing support during my studies.

ACKNOWLEDGMENTS

I would like to thank Dr. Jorge Sefair for his guidance since the junior year of my undergraduate studies. Through his course "Urban OR" I discovered an interested in research and mathematical modeling and now I am moving forward with my studies in pursuit of a PhD degree starting next Fall. I would not be able to stand where I stand today without his support during the last two years.

I would like to thank Dr. Ronald Askin for his perspective and experience, and for the "reality checks" which allowed the models developed to be applicable and relevant to real manufacturing systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Background

The advancement of technology in the last few decades has made several types of computer devices accessible to an increasing number of individuals. Simultaneously, the range of applications for computer devices has greatly expanded, encompassing consumer electronics, health devices, automotive applications, and industrial applications, among others. Being one of the fundamental components of virtually every electronic device, the demand for Printed Circuit Boards (PCBs) increased dramatically over the past few years, challenging industries to develop efficient practices to fulfill the demand. In addition, companies must quickly respond to shifts in consumer preferences and invest in product development to create competitive advantages. From the manufacturing perspective, this is a challenging task as new boards are usually manufactured in small batches, have short life cycles, and undergo multiple inspections and testing procedures before they begin to be produced in large volumes for customers. In particular, scheduling manufacturing operations of new PCBs requires consideration of custom product routes for each batch of boards, different processing times for the machines, sequence-dependent setup times, and different skill sets for each worker.

In this work, we use optimization to model this problem since it is a combinatorial problem with a great level of detail, all of which can be specified in mathematical modeling. However, there are computational challenges to this approach due to the

large number of binary decision variables which describe job and operation assignments. At the core of our model is the use of disjunctive constraints reflecting the sequence in which jobs are processed. That is, if two jobs share a resource, one needs to be processed before the other. Likewise, at most one operation of a job can be active at any time. We use the same structure to sequence the operations that each worker must perform. We develop a mathematical heuristic which allows medium-sized instances (with 10 jobs, 10 operations each, 21 machines, 15 workers) to be solved with training option.

This problem is motivated by the real operation of a research and development facility for semiconductor manufacturing.

1.2 Problem Description

The implementation of Surface-Mount Technology (SMT) for the manufacturing of PCBs became widely used in the mid-1980s. This technology has allowed part of the assembly process to be automated and organized into production lines. SMT lines allow individual boards to move sequentially from one machine to the next. This movement of individual boards differs from other typical manufacturing processes, where an entire batch of boards must complete an operation before proceeding to the following operation.

SMT operations are required for manufacturing of all batches/jobs, and include processes such as applying solder paste, placing surface-mounted components, inspecting positions of placed parts, and reflowing the solder by having the boards go through ovens, a process which fixes the components in place. Following the SMT operations, the boards may return to the beginning of the SMT lines if they are

double-sided; otherwise, they proceed to other operations which are job-specific (based on a preassigned sequence of operations depending on the type of board). Each job can be assigned to any available machine that can perform a required operation. Trained workers must oversee each operation, which can be interrupted at any moment if errors are found through automated or manual inspections and testing procedures.

Figure 1 illustrates the layout of a facility with two SMT lines and other operations which could include 3D X-ray inspections, selective solder, wave solder, microscope inspection, mechanical assembly, among others. Note that the job represented in gray goes through the SMT processes two times, while the job represented in black goes through SMT only once. There is no predetermined order for the custom (non-SMT) operations, otherwise we could classify this problem as a flexible flow-shop (Pinedo 2008, Kurz and Askin 2004) − a type of layout where there is a predetermined order of operations, but that allows for some jobs to skip operations.

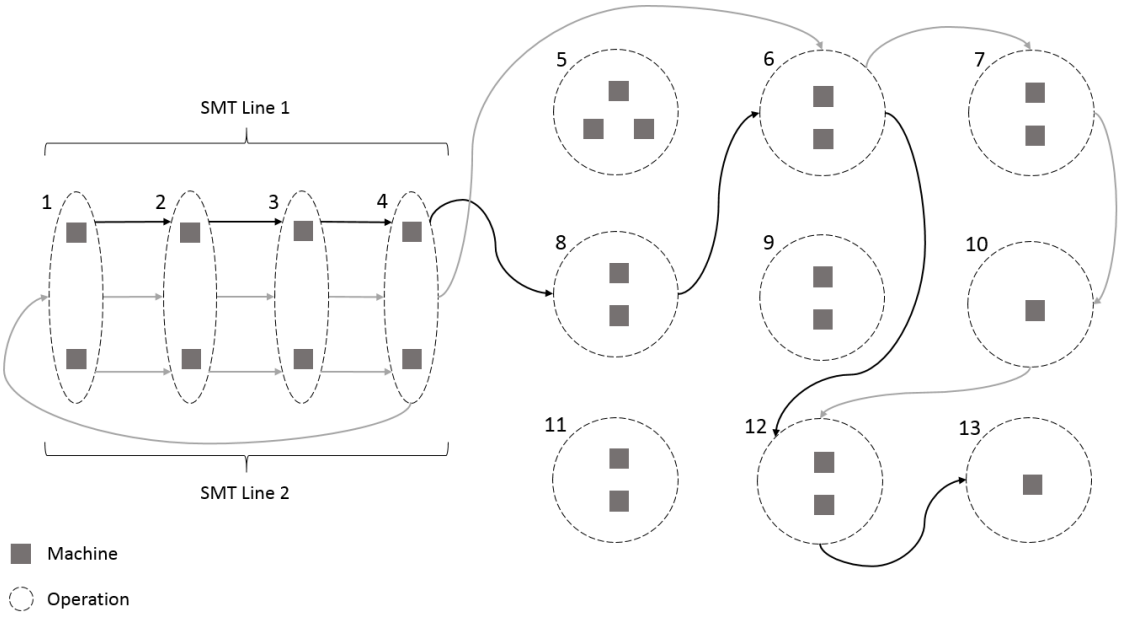

Figure 1: Manufacturing process of PCBs

Also note that some operations have more than one possible machine available to execute the operations. In these cases, machine assignments must also be decided by the model.

To automate the scheduling process for manufacturing of PCBs in high-mix and low-volume facilities, optimization methods can be advantageous. They allow not only to schedule all manufacturing operations, but also to determine the optimal job sequencing and machine/worker assignments. In this work, we minimize the makespan required to complete all jobs. For scheduling problems where each job requires processing in several machines, minimizing the makespan also maximizes machine utilization (Chakravarty and Balakrishnan 1997). In some cases, worker training is needed to improve the efficiency of manufacturing processes. As a result, optimization tools are also useful to produce a schedule of the duties for each worker, including the most convenient time for training, if needed.

The literature of mathematical models for the scheduling of manufacturing operations is vast. Similar problems have been formulated using extensions of the Job-shop Scheduling Problem (JSP) formulation, in which each job has to be completed according to a set of ordered operations. We have identified that the Dual-Resource Constrained Flexible Job-shop Problem (DRCFJSP) is the extension of the JSP that best fits our problem, as it can account for machining and staffing constraints simultaneously. This is relevant in our application since none of the machines can run unsupervised, and workers in specialized fields such as PCB manufacturing are highly skilled and present a significant cost to these companies, making them a valuable and often scarce resource. Job-shop implies that each job goes through a different predetermined route of operations. Flexible job-shop implies that there is more than one parallel machine to perform each operation. In our problem, parallel machines

performing the same operation are assumed to be identical. The dual constraints are the need for a machine and operator. In addition, our problem has significant sequence-dependent changeover times at machines.

Although optimization can be used to solve problems similar to this one, combinatorial problems can be extremely challenging computationally since there are many possible sequencing and resource assignment decisions to be considered. In this paper, we present two mathematical models to provide an optimal schedule to complete a set of jobs. The first one schedules a set of jobs for workers with fixed skill sets. The second one includes a training budget and allows each worker to be trained for at most one additional operation. This formulation allows newly trained workers to be assigned to operations as soon as the training is complete, including those requiring the new skills.

1.3 Literature review

Extensive research has been conducted in the field of scheduling due to the role it can play in maximizing product throughput and production efficiency in manufacturing systems (Pinedo 2008, Askin 1993, Baker 1974). Works of authors including Brucker and Schlie (1990) extended the classical Job-shop Scheduling Problem (JSP) formulation and allowed optimization methods to start being applied to more realistic problems. One class of such problems assumes that there is more than one available machine to perform each operation, and therefore the scheduling models must consider job-sequencing and machine assignments decisions together. This type of problem is known as a Flexible Job-shop Scheduling Problem (FJSP). Brucker and Schlie (1990) were among the first authors to propose a solution to

the FJSP, which they called a Job-shop Scheduling Problem with multi-purpose machines. They introduced a polynomial-time algorithm to solve the FJSP for two jobs using a shortest path problem formulation. Shen, Dauzère-Pérès, and Neufeld (2018) extended the FJSP by accounting for sequence-dependent setup times for the machines, an important feature in real-world settings because the preparation of a machine often depends on the job completed immediately before on the same machine. The problem was then solved with a tabu search algorithm approach. As an alternative solution method to the FJSP with sequence-dependent setup times, Rossi (2014) provided a swarm intelligence approach to this problem. In our situation, changing part feeders, replacing solder paste, adjusting reflow temperatures, can all create changeover dependence.

Models that use the FJSP formulation require the assumption that there is always an available worker to perform a job when it is assigned to a machine. As the complexity of manufacturing operations increases, this assumption no longer applies to settings where machines are abundant and workers are scarce. This led to research of the Dual-Resource Constrained Flexible Job-shop Scheduling Problem (DRCFJSP). Models in this area make different assumptions about worker availability, their skill levels, efficiency, and training (Lobo et al. 2013). Dhiflaoui, Nouri, and Driss (2018) provide a comprehensive review of Dual-Resource Constrained (DRC) scheduling problems, including problems with or without machine and worker flexibility, in which a machine or worker may or may not be able to process different jobs and operations. They also classified papers based on their solution method (exact method, heuristic or meta-heuristic) and optimization criteria.

Due to the need to solve three subproblems simultaneously, namely job sequencing, machine assignment, and worker assignment, DRCFJSP problems are NP-hard and,

therefore, it is likely that there is no known algorithm capable of solving them efficiently. Indeed the JSP alone is NP-hard (Sotskov and Shakhlevich 1995). Several authors have proposed heuristic methods (e.g., Lobo et al. 2013) and meta-heuristic approaches (e.g., Zhang, Wang, and Xu (2017), Zheng and Wang (2016), Lei and Guo (2015), Wu et al. (2018)) to obtain approximate solutions to the DRCFJSP. Heuristic methods are typically constructive algorithms (e.g., greedy algorithms) developed for specific problems, while meta-heuristics are population-based techniques which can be applied to different problem categories. Zhang, Wang, and Xu (2017) proposed a hybrid solution method to balance global search and local search by using particle swarm optimization combined with simulated annealing based on a Variable Neighborhood Search (VNS) structure. Zheng and Wang (2016) also developed a hybrid approach to balance global and local search through a knowledge-guided optimization algorithm. Lei and Guo (2015) developed a sequential approach with two VNS procedures to obtain solutions to two subproblems of the DRCFJSP − operation sequencing and resource assignment. Wu et al. (2018) used a hybrid approach based on a genetic algorithm with a variable neighborhood search to provide solutions to the DRCFJSP, while accounting for workers with heterogeneous abilities and learning capabilities. The intuition behind these hybrid approaches lies in the ability of rapidly converging to a local solution but also exploring other solutions to avoid being limited to local optima neighborhoods. However, neither heuristic nor meta-heuristic approaches can guarantee the optimality of the achieved solution.

Park (1991) was among the first authors to study how worker flexibility can impact production capacity, showing that the level of cross-training can impact production in manufacturing environments where the number of machines is larger than the number of workers. The author compared production levels for different levels of cross-training

and dispatching rules. His results indicated that cross-training has diminishing returns of scale, where initial worker cross-training has the most marginal improvement when workers initially had no cross-training.

Our work seeks to provide new mathematical formulations that incorporate details relevant in real manufacturing contexts. One of the main contributions of our work is the development of a DRCFJSP formulation that includes both sequence-dependent setup times and worker travel time from one machine to another. These setup times depend on the sequence of jobs running through a machine or executed by a worker. In our second model, we consider the value of cross-training workers to increase production capacity. If it is beneficial for the production plan, our model recommends the best skills to train the workers on, along with the appropriate time-windows that avoid overlaps with work assignments. Moreover, we present two algorithmic procedures which allow our models to be solved to optimality in small-sized problem instances with our training model including setup times, and in small to medium-sized instances with our no-training model including setup times. One of our algorithmic procedures solves the model with no cross-training and with cross-training sequentially to warm-start the training model. Our second procedure is a math-heuristic that uses the Largest Processing Time (LPT) heuristic for machine assignment and the Shortest Processing Time (SPT) heuristic to create a sequence of jobs in the SMT lines. The LPT heuristic has been shown to give solutions that are within a bound from the minimum makespan for identical parallel machines (Graham 1969), and the SPT heuristic is known to minimize total weighted flow-time (Webster and Baker 1995).

To be able to visualize the output of our models, we created an algorithm that generates Gantt charts with a schedule overview and with a worker schedule view.

The worker schedule shows the order of job-operation pairs that each worker must complete, which allows workers to plan out their work day according to their work assignments and to move efficiently from one machine to the next. To obtain these schedules, we used a two-stage procedure that finds an alternative solution to the general makespan minimization problem that also minimizes the makespan of each job. It is preferable that workers and machines are idle as soon as possible to take breaks or carry on with other tasks.

Chapter 2

A CONCURRENT SCHEDULING MODEL FOR MACHINES AND WORKERS

In this section, we present a mixed-integer programming model for concurrent scheduling of machines and workers. Our model can account for the skill sets of each worker and schedule a set of jobs to optimality.

2.1 Scheduling model without training option

This model can optimally schedule a set of jobs and all their operations while simultaneously considering machine and worker assignments. In this section, we explain the modeling assumptions, present the notation used in our model, and describe the mathematical constraints in detail.

We assume that all operations are non-preemptive, therefore, once an operation starts, it cannot be interrupted until it has been completed. In addition, an operation requires a machine and a worker with the skills to operate it to be processed. This means that both resources will be "busy" and unable to be scheduled for other operations during the time-window given by the start and finish time (start time plus processing time) of an operation. In this model, the processing times are assumed to be deterministic and dependent on the job and operation type. We assume that the processing time of any operation is identical regardless of the machine or worker performing the operation.

Within a job, we assume that the order of operations is not flexible, and therefore an operation of a job cannot begin until its preceding operations have been completed.

Moreover, we assume that a worker cannot perform an operation that is not part of their skill set. Similarly, a machine cannot be used for an operation that it is not capable of performing.

We assume that a changeover time is required when a job-operation is performed immediately after another one by the same worker or machine. This additional changeover time is assumed to be sequence-dependent. To mathematically describe our problem, we define B as the set of jobs to be performed, and O_i as the set of operations required by job $i \in B$. We denote the set of available machines by M and the set of machines eligible to perform operation j by M_j. Similarly, we denote the set of workers by W and those able to perform operation j by W_j.

We use decision variables $z_{iji'j'}$ to determine if job (i, j) precedes (i', j'). This variable does not establish if (i, j) is an *immediate* predecessor of (i', j'), which implies that there might be other jobs between (i, j) and (i', j') for a machine or worker sequence of assignments. We use parameter $c_{iji'j'}$ to denote the changeover time between job-operation pairs (i, j) and (i', j'). Moreover, we denote the travel time of a worker from machine k to machine k' by $\theta_{kk'}$. We assume that the following properties hold true for the changeover times, as in Shen, Dauzère-Pérès, and Neufeld (2018):

$$c_{iji'j'} + c_{i'j'i''j''} \geq c_{iji''j''} \qquad \forall i, i', i'' \in B, \forall j \in O_i, \forall j' \in O_{i'}, \forall j'' \in O_{i''}$$

$$\theta_{kk'} + \theta_{k'k''} \geq \theta_{kk''} \qquad \forall k, k', k'' \in M$$

In addition to sequencing decisions, our model makes decisions including operation assignments to machines and workers. This is done through decision variable x_{ijkl}, that is equal to one if job-operation pair $(i, j), i \in B, j \in O_i$, is executed by worker

$l \in W_j$ on machine $k \in M_j$. We denote the start time of each job-operation pair (i, j) by s_{ij}. These are all intricate decisions, because changing the order of job-operation pairs or resource assignments can impact the start time of each job-operation pair and potentially increase or decrease the overall makespan.

Next we show a summary of all sets, parameters and decision variables used in this model.

Sets and parameters

B:	Set of jobs (index i)
O:	Set of operations (index j)
M:	Set of machines (index k)
W:	Set of workers (index l)
M_j:	Set of machines eligible for operation j
M_n:	Set of machines in SMT line n
W_j :	Set of workers that can perform operation j
S:	Set of SMT lines (index n)
O_i:	Set of ordered operations of job i (set of consecutive integers)
t_{ij}:	Processing time of operation j of job i
α_j:	1 if operation j is part of SMT; 0 otherwise
L^m:	A large number (for machine Big-M constraints)
L^w:	A large number (for worker Big-M constraints)
$c_{iji'j'}$:	Time required to changeover from job-order pair (i, j) to (i', j')
$\theta_{kk'}$:	Time required for a worker to move from machine k to machine k'

Decision variables

x_{ijkl}:	1 if operation j of job i is assigned to worker l and machine k; 0 otherwise
s_{ij}:	Start time of operation j for job i
$y^m_{iji'j'}$:	Auxiliary variable that equals 0 if job-operation pairs (i, j) and (i', j') are assigned to the same machine, for $i < i'$
$y^w_{iji'j'}$:	Auxiliary variable that equals 0 if job-operation pairs (i, j) and (i', j') are assigned to the same worker, for $i < i'$
$z_{iji'j'}$:	1 if (i, j) precedes (i', j'), for $i < i'$; 0 otherwise.
$\tilde{c}_{iji'j'}$:	Time required for worker to changeover from (i, j) to (i', j')
w_{in}:	Auxiliary variable that equals 1 if job i uses SMT line n
$C_{\max}$:	Maximum completion time of all jobs (makespan)

Using the notation above, we describe all mathematical constraints used in our model. The objective function in (1) seeks to minimize the makespan, which is the total time required to complete all jobs. Constraints in (2) define the makespan based on the completion time of the last operation of each job.

$$\min\ C_{\max} \tag{1}$$

$$s.t.\ \ C_{\max} \geq s_{i|O_i|} + t_{i|O_i|} \qquad \forall i \in B \tag{2}$$

Because each job has a specific set of ordered operations (route) to be completed, Constraints in (3) establish the precedence relations among the operations of each job, and determine that an operation that does not belong to SMT cannot start until all of its preceding operations have been completed. Constraints in (4) determine that all SMT operations of a job must start together as these require several machines working simultaneously.

$$s_{i,j+1} \geq s_{ij} + t_{ij} \qquad \begin{array}{l} \forall i \in B, \forall j = 1, \ldots, \\ |O_i| - 1 : \alpha_{j+1} = 0 \end{array} \tag{3}$$

$$s_{ij} = s_{i,j+1} \qquad \begin{array}{l} \forall i \in B, \forall j \in O_i : j = 1, \ldots, \\ |O_i| - 1 : \alpha_j = \alpha_{j+1} = 1 \end{array} \tag{4}$$

The assignment of an operation from a job to an eligible worker and an eligible machine is enforced by Constraints in (5).

$$\sum_{k \in M_j} \sum_{l \in W_j} x_{ijkl} = 1 \qquad \forall i \in B, \forall j \in O_i \tag{5}$$

Disjunctive Constraints (6)-(9) are only active when there is a possible resource conflict due to (i, j) and (i', j') sharing a machine or worker resource. If this occurs, an operation may not start until the previous operation has been completed and both the machine has been setup for the next operation and the assigned worker has arrived to the machine location. Note that if $y^m_{iji'j'}$ is equal to zero, constraints (6) and (7) become "active", in which case variable $z_{iji'j'}$ will determine if (i, j) will precede (i', j') or vice versa. If $z_{iji'j'} = 1$, then the corresponding Constraint (7) is relaxed and Constraint (6) enforces that operation j' of job i' should start after the completion time of operation j of job i, including changeover time. A similar analysis can be done when $z_{iji'j'} = 0$. Constraints (6) and (7) are relaxed when $y^m_{iji'j'} = 1$ regardless of the value of $z_{iji'j'}$. Similarly, Constraints (8) are enforced when $y^w_{iji'j'} = 0$, leading variable $z_{iji'j'}$ to determine if (i, j) will precede (i', j') or vice versa. Note that the changeover time in Constraints (8) and (9), $\tilde{c}_{iji'j'}$, is determined by the selection of machines, as it is not known where a worker will be performing (i, j) and (i', j').

$$\begin{aligned} & s_{ij} + t_{ij} + c_{iji'j'} \leq && \forall i \in B, \forall j \in O_i, \forall i' \in B : \\ & \quad s_{i'j'} + L^m(1 - z_{iji'j'} + y^m_{iji'j'}) && i' < i, \forall j' \in O_{i'} \end{aligned} \tag{6}$$

$$\begin{aligned} & s_{i'j'} + t_{i'j'} + c_{i'j'ij} \leq && \forall i \in B, \forall j \in O_i, \forall i' \in B : \\ & \quad s_{ij} + L^m(z_{iji'j'} + y^m_{iji'j'}) && i' < i, \forall j' \in O_{i'} \end{aligned} \tag{7}$$

$$\begin{aligned} & s_{ij} + t_{ij} + \tilde{c}_{iji'j'} \leq && \forall i \in B, \forall j \in O_i, \forall i' \in B : \\ & \quad s_{i'j'} + L^w(1 - z_{iji'j'} + y^w_{iji'j'}) && i' < i, \forall j' \in O_{i'} \end{aligned} \tag{8}$$

$$\begin{aligned} & s_{i'j'} + t_{i'j'} + \tilde{c}_{i'j'ij} \leq && \forall i \in B, \forall j \in O_i, \forall i' \in B : \\ & \quad s_{ij} + L^w(z_{iji'j'} + y^w_{iji'j'}) && i' < i, \forall j' \in O_{i'} \end{aligned} \tag{9}$$

Constraints in (10) enforce the condition that $y^m_{iji'j'}$ takes the value of 0 if job-

operation pairs (i,j) and (i',j') are scheduled to use the same machine. The expression in brackets in Constraint (10) determines whether job-operation pairs (i,j) and (i',j') use the same machine. If so, the right hand side of (10) is equal to 0, which forces $y^m_{iji'j'} = 0$, and (6) and (7) become active. Similarly, Constraints in (11) force $y^w_{iji'j'} = 0$ if job-operation pairs (i,j) and (i',j') are assigned to the same worker, which makes (8) and (9) become active. Constraints in (12) determine the changeover time for a worker moving from machine k, where job-operation pair (i,j) is performed, to machine k', where job-operation pair (i',j') is performed. Constraints (13) and (14) tighten the feasible region for this problem by eliminating solutions in which y-variables are equal to one and z-variables are equal to zero. This is valid since having a y-variable equal to one implies that the disjunctive constraints are relaxed, so any value of the z-variables is feasible. Constraints in (15) determine that a job-operation pair cannot be assigned to a machine in an SMT line unless the corresponding job has been assigned to the same line. Constraints in (16) determine that only one SMT line can be selected for each job.

$$y^m_{iji'j'} \leq 2 - \left[\sum_{l \in W_j} x_{ijkl} + \sum_{l \in W_{j'}} x_{i'j'kl} \right] \quad \begin{array}{l} \forall i \in B, \forall j \in O_i, \forall i' \in B : i' > \\ i, \forall j' \in O_{i'}, \forall k \in M_j \cap M_{j'} \end{array} \tag{10}$$

$$y^w_{iji'j'} \leq 2 - \left[\sum_{k \in M_j} x_{ijkl} + \sum_{k \in M_{j'}} x_{i'j'kl} \right] \quad \begin{array}{l} \forall i \in B, \forall j \in O_i, \forall i' \in B : i' > \\ i, \forall j' \in O_{i'}, \forall l \in W_j \cap W_{j'} \end{array} \tag{11}$$

$$\tilde{c}_{iji'j'} \geq \theta_{kk'} \Big(x_{ijkl} + x_{i'j'k'l} - 1 \Big) \quad \begin{array}{l} \forall i \in B, \forall j \in O_i, \forall i' \in B, \\ \forall j' \in O_{i'}, \forall l \in W_j \cap W_{j'}, \\ \forall k \in M_j, \forall k' \in M_{j'} \end{array} \tag{12}$$

$$y^m_{iji'j'} \leq z_{iji'j'} \quad \begin{array}{l} \forall i \in B, \forall j \in O_i, \forall i' \in B : \\ i' > i, \forall j' \in O_{i'} \end{array} \tag{13}$$

$$y^w_{iji'j'} \leq z_{iji'j'} \qquad \forall i \in B, \forall j \in O_i, \forall i' \in B : i' > i, \forall j' \in O_{i'} \tag{14}$$

$$x_{ijkl} \leq w_{in} \qquad \forall i \in B, \forall j \in O_i, \forall k \in M_j \cap M_n, \forall l \in W, \forall n \in S \tag{15}$$

$$\sum_{n \in S} w_{in} = 1 \qquad \forall i \in B \tag{16}$$

Constraints (17) and (18) determine the nature of the binary decision variables. Note that Constraints (19)-(21) define w_{in}, $y^m_{iji'j'}$ and $y^w_{iji'j'}$ as continuous variables. This is because Constraint (5) forces exactly one x_{ijkl} variable to be equal to one for each job-operation pair, which together with (15) and (16) imply that only one w_{in} variable would be equal to one. The same analysis applies to $y^m_{iji'j'}$ and $y^w_{iji'j'}$ variables which only need to be equal to zero when resources are shared, which is enforced in Constraints (10) and (11). Constraints (22)-(24) indicate that all continuous variables representing time should be non-negative.

$$x_{ijkl} \in \{0, 1\} \qquad \forall i \in B, \forall j \in O_i, \forall k \in M_j, \forall l \in W \tag{17}$$

$$z_{iji'j'} \in \{0, 1\} \qquad \forall i \in B, \forall j \in O_i, \forall i' \in B : i' > i, \forall j' \in O_{i'} \tag{18}$$

$$0 \leq w_{in} \leq 1 \qquad \forall i \in B, \forall n \in S \tag{19}$$

$$0 \leq y^m_{iji'j'} \leq 1 \qquad \forall i \in B, \forall j \in O_i, \forall i' \in B : i' > i, \forall j' \in O_{i'} \tag{20}$$

$$0 \leq y^w_{iji'j'} \leq 1 \qquad \forall i \in B, \forall j \in O_i, \forall i' \in B : i' > i, \forall j' \in O_{i'} \tag{21}$$

$$s_{ij} \geq 0 \qquad \forall i \in B, \forall j \in O_i \tag{22}$$

$$\tilde{c}_{iji'j'} \geq 0 \qquad \forall i \in B, \forall j \in O_i, \forall i' \in B, \forall j' \in O_{i'} : i' \neq i \cup j' \neq j \tag{23}$$

$$C_{\max} \geq 0 \tag{24}$$

2.2 Scheduling model with training option

In this model, we allow some workers to receive training to perform one operation that is not initially in their skill set, which can impact the total time required to finish all jobs. The model determines the best timing for each worker to receive training, if any, guaranteeing that there is no overlap with any operation assigned to the same worker.

For this model, the same assumptions from Section 2.1 apply. In addition, we assume that a worker is allowed to perform an operation that is not part of their skill set only if training is finished beforehand. Each worker can receive training on at most one operation, and there is a cardinality constraint which limits the number of workers that can be trained.

Below we detail the additional sets, parameters, and decision variables used in this model.

Sets and parameters

W_j: Set of workers that can perform operation j (with no prior training required)
Q_l: Set of operations that can be performed by worker l
δ_j: Time required to train a worker to perform operation j
T: Maximum number of workers that can be trained

Decision variables

p_{jl}:	1 if worker l is trained on operation j; 0 otherwise
$\gamma_{ijj'l}$:	Auxiliary variable that equals 0 if worker l is assigned to job-operation pair (i, j) and scheduled for training in j'
h_{ij}:	Auxiliary variable that equals 1 if job-operation pair (i, j) is completed before the worker assigned to it attends training
τ_{jl}:	Start time of training in operation j for worker l

This model is built upon model (1)-(24) except that Constraints (5) are slightly modified to allow the assignment of an operation to any worker and to an eligible machine.

$$\sum_{k \in M_j} \sum_{l \in W} x_{ijkl} = 1 \qquad \forall i \in B, \forall j \in O_i \tag{25}$$

Constraints in (17) are substituted by the following set of constraints, which allow the creation of assignment variables for workers that are initially unable to perform an operation. This change allows for workers to be assigned to newly trained operations after training is complete.

$$x_{ijkl} \in \{0, 1\} \qquad \forall i \in B, \forall j \in O_i, \forall k \in M_j, \forall l \in W \tag{26}$$

Constraints in (27) determine that a worker cannot perform an operation that is not included in their skill set unless training is received. Constraints (28) and (29) limit the training provided to each worker and the number of workers trained, respectively.

$$x_{ijkl} \leq p_{jl} \qquad \forall i \in B, \forall j \in O_i, \forall k \in M_j, \forall l \in W \setminus W_j \tag{27}$$

$$\sum_{j \in O \setminus Q_l} p_{jl} \leq 1 \qquad \forall l \in W \tag{28}$$

$$\sum_{l \in W} \sum_{j \in O \setminus Q_l} p_{jl} \leq T \tag{29}$$

Disjunctive Constraints (30) and (31) avoid overlap between training activities and work assignments and are only active if the same worker is assigned to an operation and to training. If this occurs, one task cannot begin before the previous one is completed. Note that Constraints (30) and (31) become active when $\gamma_{ijj'l} = 0$, in which case variable h_{ij} will determine if job-operation pair (i, j) will precede training for worker l or vice versa. In this case, if $h_{ij} = 1$, then Constraint (31) is relaxed and Constraint (30) enforces that job-operation pair (i, j) cannot start until training has been completed on operation j'. For cases when a worker is assigned to the same operation they have been trained on (i.e. $j = j'$), the work assignment may only start after the training has been completed, as enforced by Constraint (32).

$$s_{ij} \geq \tau_{j'l} + \delta_{j'} - L(1 - h_{ij} + \gamma_{ijj'l}) \qquad \begin{array}{l} \forall i \in B, \forall j \in O_i, \\ \forall j' \in O, \forall l \in W \setminus W_j \end{array} \tag{30}$$

$$s_{ij} + t_{ij} \leq \tau_{j'l} + L(h_{ij} + \gamma_{ijj'l}) \qquad \begin{array}{l} \forall i \in B, \forall j \in O_i, \\ \forall j' \in O, \forall l \in W \setminus W_j \end{array} \tag{31}$$

$$s_{ij} \geq \tau_{jl} + \delta_j - L\left[1 - \sum_{k \in M_j} x_{ijkl}\right] \qquad \forall i \in B, \forall j \in O_i, \forall l \in W \setminus W_j \tag{32}$$

Constraints in (33) enforce the condition that $\gamma_{ijj'l}$ takes the value of 0 if job-operation pair (i, j) and training on operation j' are both assigned to worker l.

Constraints (34) and (35) enforce the condition that $\gamma_{ijj'l}$ takes the value of 1 if worker l is not assigned to training on j' or to job-operation pair (i, j).

$$\gamma_{ijj'l} \leq 2 - \left[\sum_{k \in M_j} x_{ijkl} + p_{j'l} \right] \qquad \forall i \in B, \forall j \in O_i, \forall j' \in O, \forall l \in W_j \setminus W_{j'} \tag{33}$$

$$\gamma_{ijj'l} \geq 1 - p_{j'l} \qquad \forall i \in B, \forall j \in O_i, \forall j' \in O, \forall l \in W_j \setminus W_{j'} \tag{34}$$

$$\gamma_{ijj'l} \geq 1 - x_{ijkl} \qquad \forall i \in B, \forall j \in O_i, \forall j' \in O, \forall k \in M_j, \forall l \in W_j \setminus W_{j'} \tag{35}$$

Constraints (36) and (37) determine which variables are binary, Constraint (38) provides the upper and lower bounds for continuous variable $\gamma_{ijj'l}$, and Constraint (39) indicates that all continuous variables representing the beginning of training time should be non-negative.

$$p_{jl} \in \{0, 1\} \qquad \forall j \in O, \forall l \in W \setminus W_j \tag{36}$$

$$h_{ij} \in \{0, 1\} \qquad \forall i \in B, \forall j \in O_i \tag{37}$$

$$0 \leq \gamma_{ijj'l} \leq 1 \qquad \forall i \in B, \forall j \in O_i, \forall j' \in O, \forall l \in W \setminus W_j \tag{38}$$

$$\tau_{jl} \geq 0 \qquad \forall j \in O, \forall l \in W \setminus W_j \tag{39}$$

Chapter 3

ALGORITHMIC ENHANCEMENTS

In this section, we discuss two strategies to deal with the great computational complexity of our model, which is NP-hard and therefore not expected to be able to handle large-scale instances.

3.1 Warm-start with no-training model

This strategy involves solving the model without training option and using the obtained solution to warm-start the training model. The intuition behind it is that offering training can only keep the makespan the same or improve it. This is because any feasible solution the the no-training model is feasible to the training model. In other words, increasing the level of cross-training cannot deteriorate the previously obtained solution.

3.2 Warm-start with heuristic techniques

Another strategy used is based on two heuristic approaches which are well known in manufacturing environments. The first one is the Largest Processing Time (LPT) heuristic (Askin 1993), typically used to assign jobs to identical parallel machines with the goal of balancing the workloads on each machine. Because our problem has SMT lines which every job must go through, we used this heuristic to decide which SMT line each job should be assigned to. After assigning jobs to the SMT lines, we

sequence them based on the Shortest Processing Time (SPT) heuristic, a dispatching rule which performs well with respect to mean job flow time.

The pseudo-code for the heuristic techniques described is shown in Algorithm 1. The goal of the heuristic procedure is to create an ordered list of jobs that will be assigned to the SMT lines. The algorithm takes as input a matrix with all jobs numbered from 1 to $|B|$ in the first column and their corresponding SMT processing time in the second column. For the algorithm below, we are assuming there are two SMT lines available.

Algorithm 1 LPT + SPT

```
1:  job := 0
2:  index := 0
3:  l1 := ∅                          → Initialize array of jobs for SMT line 1
4:  l2 := ∅                          → Initialize array of jobs for SMT line 2
5:  while a ≠ ∅ do
6:      begin
7:      index := max(a.col(1))       → Get maximum processing time in col. 1
8:      job := a[index, 0]           → Get corresponding job number in col. 0
9:      if sum(l1.col(1)) < sum(l2.col(1)) then
10:         l1.add(job, a[index])    → Add to l1 if it has lowest total proc. time
11:     else
12:         l2.add(job, a[index])                    → Add to l2 otherwise
13:     a.delete(a[index])
14:     end
15: l1.reverse                       → Reorder jobs in l1 by increasing processing time
16: l2.reverse                       → Reorder jobs in l2 by increasing processing time
```

As shown in Algorithm 1, the jobs are successively assigned to the SMT line that has lowest total processing time until all jobs have been assigned. The goal is to balance the workload on each SMT line. After all jobs have been assigned, they are sequenced in decreasing order, so we reverse the job order to apply the SPT heuristic.

The motivation of using these heuristics is to significantly reduce the number of possible sequences (permutations) and machine assignments for the SMT lines, so

the models can be accelerated. We still allow the model to decide what is the best worker assignment for each job and operation, as well as all machine assignments and sequencing decisions for the remaining non-SMT operations. This heuristic solution is used to in both the no-training and the training model.

In the following section, we compare the results obtained with our models and the two enhancement strategies with respect to performance, relaxation gap, and solution quality.

Chapter 4

RESULTS

In this chapter, we describe an illustrative example to explain the model's features and show how the results can be visualized and interpreted. In the following section we present the results after running the model with randomly generated instances of different sizes. We also show how our two algorithmic techniques − warm-start training model with no-training model and warm-start with LPT-SPT heuristic − can be used to reduce the computational time of our models.

4.1 Illustrative example

Consider a case with 3 jobs, 3 operations per job, and 2 workers, for which we want to find the solution that minimizes the makespan. The required operations for each job are described in Table 1.

Job i	Operation j	Eligible Machines M_j	Eligible Workers W_j
1	1	{1}	{1}
	2	{2}	{2}
	3	{3, 4}	{1}
2	3	{3, 4}	{1}
	4	{5}	{2}
	5	{6, 7}	{1}
3	5	{6, 7}	{1}
	6	{8}	{2}
	7	{9}	{1}

Table 1: Data for illustrative example

The results for the illustrative examples can be visualized using Gantt charts. Below are two different views of the same schedule with the results from the no-training model. In Figures 2 and 3, the minimum makespan to complete all jobs is 19.4 time units.

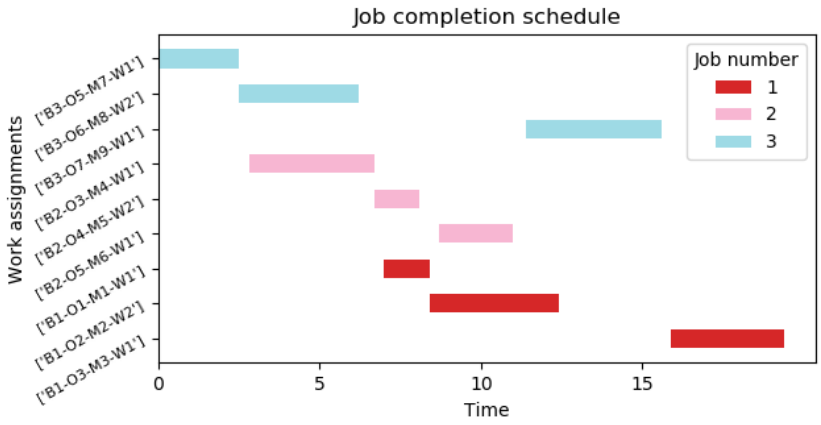

Figure 2: Gantt chart for no-training model with job categories

Figure 3: Gantt chart for no-training model with worker categories

Next we show the results of the training model, which allows the two workers to be trained in one operation each. In Figures 4 and 5, the minimum makespan to complete all jobs with training option is 16.8 time units.

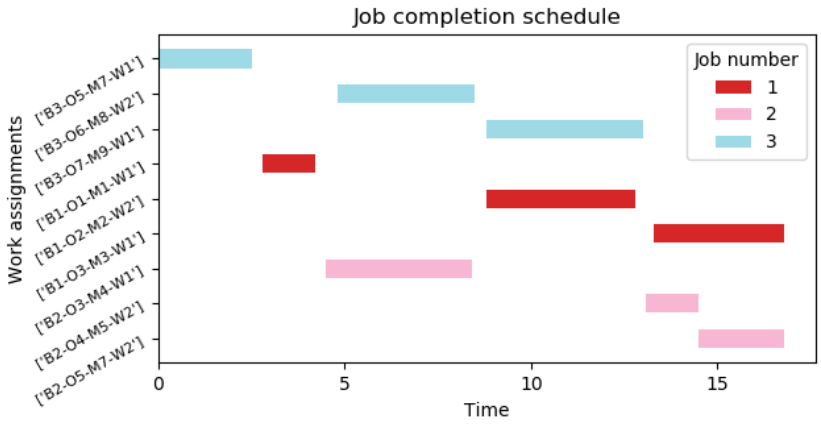

Figure 4: Gantt chart for training model with job categories

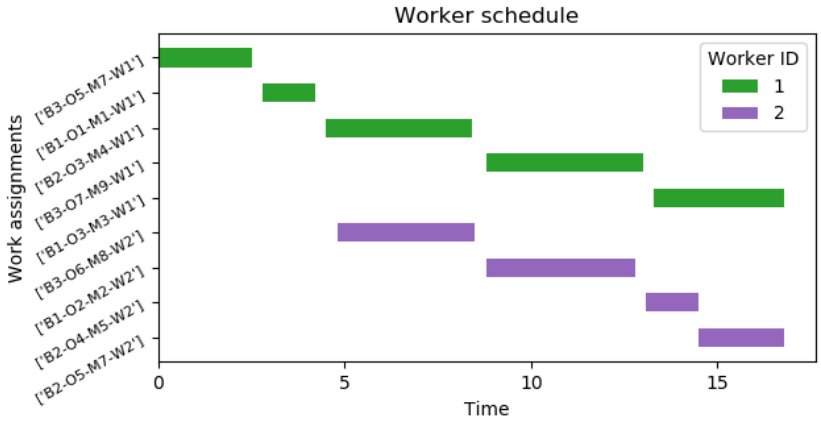

Figure 5: Gantt chart for training model with worker categories

We developed an algorithm to create the Gantt charts and visualize the results. This algorithm requires an input file with the following information:

- Job number (i)
- Operation number (j)
- Machine number (k)
- Worker ID (l)
- Start time (s)
- Processing time (t)

The completion time can be easily calculated in an additional column by adding the start time and duration of each operation. To generate a Gantt chart, the data must be sorted and grouped by the category of interest. Different views of the same schedule can be created by grouping assignments by job, machine, or worker. If we wish to view the schedule from the job completion perspective, we must create a column with the minimum start time of each job (start time of the first operation within a job). Then we can sort the data by minimum start time, job number, and start time.

We have found Gantt charts to be an effective way of visualizing results and inspecting the output of our models for accuracy.

4.2 Experimental results

We ran our no-training model and training model with instances of 5 jobs and 10 jobs. All workers can initially perform only one operation. Each operation has one or two machines on which it can be performed. For the training model, we assume that a maximum of 5 workers can receive training. All instances include 21 machines, 15

workers, and 10 operations per job. Each job has a set of 5 standard operations to go through plus 5 custom operations. The custom operations are sampled from a set of 10 operations using the following procedure:

- 2 out of 3 soldering techniques are chosen without replacement
- 2 out of 3 inspection methods are chosen without replacement
- 1 out of 4 other custom operations is chosen

The processing times, setup times and training times were generated using uniform distributions with different parameters. To implement sequence-dependent setup times, we assume that each job has a "family", and that the setup time is shorter on average when a job immediately follows another job from the same family. Below is a summary of how the parameters for the uniform distributions were generated.

- Processing time: $U(0.5, 4.5)$
- Changeover time for jobs within the same family: $U(0.5, 1.0)$
- Changeover time for jobs from different families: $U(1.0, 4.0)$
- Changeover time for worker moving between different machines (assumed to be zero for same machine): $U(0.25, 0.50)$
- Training time: $U(4.0, 8.0)$

We used Gurobi as a solver with a two-hour run-time limit on a computer with an Intel i5 processor and 4 GB of RAM. In Tables 2 and 4, we denote the no-training optimization model as **MIP**, the MIP including the constraints for the math-heuristic (LPT-SPT) as **MH-MIP**, and the procedure of warm-starting the MIP model with the math-heuristic MIP as **MIP WS with MH-MIP**. Tables 2 and 3 show the results for the instances with 5 jobs, 10 operations, 21 machines and 15 workers.

Model without training									
Instance	MIP			MH-MIP			MIP WS with MH-MIP		
	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap
1	1	26.5	0.00%	1	29.9	0.00%	1	26.5	0.00%
2	1	34.0	0.00%	1	36.3	0.00%	1	34.0	0.00%
3	1	34.4	0.00%	1	36.3	0.00%	1	34.4	0.00%
4	1	32.5	0.00%	1	33.6	0.00%	1	32.5	0.00%
5	1	32.3	0.00%	1	35.7	0.00%	1	32.3	0.00%
Average	**1**	**31.9**	**0.00%**	**1**	**34.4**	**0.00%**	**1**	**31.9**	**0.00%**

Table 2: Results for no-training model with 5 jobs, 10 operations, 21 machines and 15 workers

Note that the solutions from the math-heuristic MIP are not significantly larger than the MIP solutions, but the computational time is the same for both procedures for the instances with 5 jobs for the no-training and training model, respectively.

In Tables 3 and 5, we denote as the training optimization model as **MIP**, the MIP warm-started with the no-training model as **MIP WS with NT**, the training MIP including the constraints for the math-heuristic as **MH-MIP**, and the procedure of warm-starting the MIP model with the math-heuristic MIP as **MIP WS with MH-MIP**.

Model with training												
Instance	MIP			MIP WS with NT			MH-MIP			MIP WS with MH-MIP		
	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap
1	511	24.6	0.00%	182	24.6	0.00%	10	29.9	0.00%	235	24.6	0.00%
2	633	33.6	0.00%	289	33.6	0.00%	16	35.9	0.00%	660	33.6	0.00%
3	324	33.2	0.00%	29	33.2	0.00%	25	36.1	0.00%	39	33.2	0.00%
4	276	30.5	0.00%	33	30.5	0.00%	9	31.9	0.00%	24	30.5	0.00%
5	386	28.7	0.00%	365	28.7	0.00%	26	33.5	0.00%	474	28.7	0.00%
Average	**426**	**30.1**	**0.00%**	**180**	**30.1**	**0.00%**	**17**	**33.5**	**0.00%**	**286**	**30.1**	**0.00%**

Table 3: Results for training model with 5 jobs, 10 operations, 21 machines and 15 workers

Note that all instances could be solved with the MIP in 426 seconds on average. The remaining solution methods reached optimal solutions faster than the MIP, with

the MH-MIP being the fastest method. The math-heuristic provided a solution which is 11.30% larger than the best makespan value, on average. The benefit of allowing cross-training is a makespan reduction of 5.64% on average.

Tables 4 and 5 show the results for the instances with 10 jobs for the no-training and training model, respectively.

Model without training									
Instance	MIP			MH-MIP			MIP WS with MH-MIP		
	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap
1	421	45.9	0.00%	2	46.6	0.00%	648	45.9	0.00%
2	354	49.7	0.00%	3	55.5	0.00%	634	49.7	0.00%
3	514	45.4	0.00%	2	52.9	0.00%	256	45.4	0.00%
4	197	47.5	0.00%	2	52.7	0.00%	134	47.5	0.00%
5	78	45.9	0.00%	1	48.6	0.00%	89	45.9	0.00%
Average	**313**	**46.9**	**0.00%**	**2**	**51.3**	**0.00%**	**352**	**46.9**	**0.00%**

Table 4: Results for no-training model with 10 jobs, 10 operations, 21 machines and 15 workers

The results in Table 4 indicate that the math-heuristic MIP could be solved in 1-2 seconds with a makespan which is 9.38% larger than the optimal value without the LPT and SPT constraints (on average) for the 5 instances used.

Model with training												
Instance	MIP			MIP WS with NT			MH-MIP			MIP WS with MH-MIP		
	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap	Time (s)	Obj.	Rel. gap
1	7200	47.0	35.96%	7200	45.9	33.55%	342	45.9	0.00%	7200	45.9	35.59%
2	7200	53.0	41.13%	7200	49.0	32.04%	510	52.3	0.00%	7200	52.3	43.21%
3	7200	60.9	55.99%	7200	45.1	38.36%	978	51.9	0.00%	7200	47.6	41.68%
4	7200	–	–	7200	46.7	32.33%	583	50.6	0.00%	7200	50.5	38.22%
5	7200	52.8	43.88%	7200	45.9	32.46%	887	47.9	0.00%	7200	47.5	36.78%
Average	**7200**	**53.4**	**44.24%**	**7200**	**46.5**	**33.75%**	**660**	**49.7**	**0.00%**	**7200**	**48.8**	**39.10%**

Table 5: Results for training model with 10 jobs, 10 operations, 21 machines and 15 workers

For the training model, which involves additional decisions such as which workers

should be trained and for which operations, the instances with 10 jobs could not be solved within two hours. Note that Instance 4 in Table 5 did not reach a feasible solution with the MIP. Although the instances with 10 jobs and training option could be solved with the math-heuristic, the other solution methods did not allow an optimal solution to be reached within two hours. The benefit of cross-training is not clear with 10 jobs since the training model could not reach the solutions from the no-training model, except for the MIP WS with NT, which improved the NT solution by 0.85%.

Chapter 5

CONCLUSION AND FUTURE RESEARCH

This work proposes a new mathematical formulation for the DRCFJSP with sequence-dependent setup times and training option, which can concurrently schedule machines and workers to complete all assigned jobs while minimizing the makespan. We included additional features in our model which are important in actual factories, such as sequence-dependent setup/changeover times for machines and workers, and the possibility of cross-training workers to increase production capacity. Due to the complexity of the DRCFJSP, we provided two algorithmic procedures which allow our models to be used to solve small to medium-sized instances for the model without training option, and small-sized instances for the model with training option.

Despite the computational challenge of solving these problems for larger instances, we showed how small to medium-sized instances can be solved to optimality in certain cases. Our math-heuristic allowed good quality solutions to be reached even for the instances with 10 jobs, 10 operations, 21 machines and 15 workers. Exact solutions methods allow planners to obtain better schedules than those provided by heuristics or meta-heuristics in which the relative optimality gap is unknown. We believe that further research must take place in this field to improve the performance of exact solution methods.

REFERENCES

Askin, Ronald G. 1993. *Modeling and analysis of manufacturing systems.* New York: Wiley.

Baker, Kenneth R. 1974. *Introduction to sequencing and scheduling.* New York: Wiley.

Brucker, P., and R. Schlie. 1990. "Job-shop scheduling with multi-purpose machines." *Computing* (Vienna) 45 (4): 369–375.

Chakravarty, Amiya K., and Nagraj Balakrishnan. 1997. "Job sequencing rules for minimizing the expected makespan in flexible machines." *European Journal of Operational Research* 96 (2): 274–288.

Dhiflaoui, Mondher, Houssem Eddine Nouri, and Olfa Belkahla Driss. 2018. "Dual-Resource Constraints in Classical and Flexible Job Shop Problems: A State-of-the-Art Review" [in eng]. *Procedia Computer Science* 126:1507–1515.

Graham, R. L. 1969. "Bounds on Multiprocessing Timing Anomalies." 17, no. 2 (March): 416–429.

Kurz, Mary E., and Ronald G. Askin. 2004. "Scheduling flexible flow lines with sequence-dependent setup times." *European Journal of Operational Research* 159 (1): 66–82.

Lei, Deming, and Xiuping Guo. 2015. "An effective neighborhood search for scheduling in dual-resource constrained interval job shop with environmental objective." *International Journal of Production Economics* 159:296–303.

Lobo, Benjamin J., Thom J. Hodgson, Russell E. King, Kristin A. Thoney, and James R. Wilson. 2013. "An effective lower bound on [formula omitted] in a worker-constrained job shop." *Computers and Operations Research* 40 (1): 328–343.

Park, Paul Sungchil. 1991. "The examination of worker cross-training in a dual resource constrained job shop." *European Journal of Operational Research* 52 (3): 291–299.

Pinedo, Michael L. 2008. *Scheduling: Theory, Algorithms, and Systems.* Third Edition. New York, NY: Springer New York.

Rossi, Andrea. 2014. "Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships." *International Journal of Production Economics* 153.

Shen, Liji, Stéphane Dauzère-Pérès, and Janis S. Neufeld. 2018. "Solving the flexible job shop scheduling problem with sequence-dependent setup times." *European Journal of Operational Research* 265 (2): 503–516.

Sotskov, Yu.N., and N.V. Shakhlevich. 1995. "NP-hardness of shop-scheduling problems with three jobs." *Discrete Applied Mathematics* 59 (3): 237–266.

Webster, Scott, and Kenneth R. Baker. 1995. "Scheduling Groups of Jobs on a Single Machine." *Operations Research* 43 (4): 692–703.

Wu, Rui, Yibing Li, Shunsheng Guo, and Wenxiang Xu. 2018. "Solving the dual-resource constrained flexible job shop scheduling problem with learning effect by a hybrid genetic algorithm." *Advances in Mechanical Engineering* (London, England) 10 (10).

Zhang, Jing, Wanliang Wang, and Xinli Xu. 2017. "A hybrid discrete particle swarm optimization for dual-resource constrained job shop scheduling with resource flexibility." *Journal of Intelligent Manufacturing* (New York) 28 (8): 1961–1972.

Zheng, Xiao-Long, and Ling Wang. 2016. "A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem." *International Journal of Production Research* 54 (18): 5554–5566.