The Dissertation Committee for Zhufeng Gao Certifies that this is the approved version of the following dissertation:

Assembly and Test Operations with Multipass Requirements in Semiconductor Manufacturing

Committee:

Jonathan F. Bard, Supervisor

James E Bickel

Erhan Kutanoglu

David P Morton

Leon S Lasdon

Assembly and Test Operations with Multipass Requirement in Semiconductor Manufacturing

by

Zhufeng Gao, B.S.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2014

Assembly and Test Operations with Multipass Requirement in Semiconductor Manufacturing

Zhufeng Gao, Ph.D.

The University of Texas at Austin, 2014

Supervisor: Jonathan F. Bard

In semiconductor manufacturing, wafers are grouped into lots and sent to a separate facility for assembly and test (AT) before being shipped to the customer. Up to a dozen operations are required during AT. The facility in which these operations are performed is a reentrant flow shop consisting of several dozen to several hundred machines and up to a thousand specialized tools. Each lot follows a specific route through the facility, perhaps returning to the same machine multiple times. Each step in the route is referred to as a "pass." Lots in work in process (WIP) that have more than a single step remaining in their route are referred to as multi-pass lots. The multi-pass scheduling problem is to determine machine setups, lot assignments and lot sequences to achieve optimal output, as measured by four objectives related to key device shortages, throughput, machine utilization, and makespan, prioritized in this order. The two primary goals of this research are to develop a new formulation for the multipass problem and to design a variety of solution algorithms that can be used for both planning and real-time control. To begin, the basic AT model considering only single-pass scheduling and the previously developed greedy randomized adaptive search procedure (GRASP) along with its extensions are introduced. Then two alternative schemes are proposed to solve the multipass scheduling problem. In the final phase of this research, an efficient procedure is

presented for prioritizing machine changeovers in an AT facility on a periodic basis that provides real-time support. In daily planning, target machine-tooling combinations are derived based on work in process, due dates, and backlogs. As machines finish their current lots, they need to be reconfigured to match their targets. The proposed algorithm is designed to run in real time.

Table of Contents

List of Tables ix

List of Figures xi

Chapter 1: Introduction 1

Chapter 2: Literature Review 6

Chapter 3: Problem Description 14

3.1 Explanation of terms 14

3.2 Problem statement 17

Chapter 4: Basic AT Modeling 19

4.1 Introduction to basic AT modeling 19

4.2 Basic GRASP 24

4.3 Extended model and extended GRASP 25

Chapter 5: Introduction to the Multipass Model 26

5.1 Logpoint and operation number 26

5.2 Input files 26

5.3 AT scheduling for the multipass model 32

Chapter 6: Multipass Scheduling Scheme I 35

6.1 Mathematical model 35

6.2 Solution methodology 40

6.2.1 Phase I: single-pass algorithm 41

6.2.2 Phase II: multipass algorithm without changeovers 42

6.2.3 Phase III: changeover algorithm 51

6.3 Output files 54

6.4 Computational Result 59

Chapter 7: Multipass Scheduling Scheme II 67

7.1 Mathematical model 67

7.1.1 Assignment model 68

7.1.2 Sequencing model 75
7.2 Solution Methodology 80
7.2.1 Phase I: assignment model 81
7.2.2 Phase II: sequencing model 83
7.2.3 Phase III: changeover algorithm 84
7.3 Computational Results 87
7.3. 1 GRASP vs. ASC 88
7.3.2 Assignment vs. Sequencing Solutions 94
Chapter 8: Real-time Decision Support for AT Operations 100
8.1 Comparison of current and maximum capacity solutions 101
8.2 Comparison algorithm 104
8.3 Priority list construction 105
8.3.1 Rules for setting the priorities 107
8.3.2 Calculations 107
8.3.3 Greedy randomized procedure for lot assignments 115
8.4 Mixed-integer programming model for real-time control problem 117
8.5 Computational experiments 118
8.5.1 Results for `Compare_Algorithm` 118
8.5.2 Results for Priority_List_Algorithm 120
8.5.3 Comparison of heuristic and MIP results 122

Chapter 9: Future Work 128

Appendix A: Procedure to Account for Logpoint and Operation Number in Multipass Scheme I.....130

Appendix B: : Pseudocodes for Subroutines in Multipass Scheme I.....131

B.1 INITIALIZATION 131

B.2 UPDATING CANDIDATE LOT LIST 132

B.3 CHECKING WHETHER NEXT PASS EXISTS 133

B.4 GREEDY RANDOMIZED PROCEDURE FOR ASSIGNING LOTS 134

Appendix C: Pseudocodes for Changeover Algorithm Subroutines.....137

C.1 INITIALIZATION 137

C.2 UPDATING CANDIDATE_LOT_LIST 139

C.3 CHANGEOVER A MACHINE 140

Appendix D: Complexity of the Assignment Model and Sequencing Model 142

Appe ndix E – Mathematical Programming Models for Real-Time Decision Making 146

References.....152

List of Tables

Table 1: Name and brief description of primary input files. 27

Table 2: Example of a route 30

Table 3: Portion of WIP file 31

Table 4: Output data files 54

Table 5: An example of "solution.csv" for one machine instance 56

Table 6: An example of "multi_solution.csv" for one machine instance 57

Table 7: Comparison of single-pass with multipass results 61

Table 8: Comparison of total key device shortages 62

Table 9: Weighted sum of key device shortages 63

Table 10: Weighted sum of lots processed 64

Table 11: Comparison of average makespan and machine time 65

Table 12: Runtime comparison 66

Table 13: Comparison of GRASP with ASC results 89

Table 14: Comparison of weighted sum of key device shortages 91

Table 15: Comparison of weighted sum of lots processed 91

Table 16: Comparison of number of machines 92

Table 17: Comparison of average machine time 93

Table 18: Comparison of CPU time 94

Table 19: Input and output statistics for assignment model 95

Table 20: Input and output statistics for sequencing model 95

Table 21: Comparison of lots processed and machine time 97

Table 22: Comparison of objective function value and weighted sum of key device shortages 98

Table 23: Comparison of weighted sum of lots and number of machine used .99

Table 24: Example of output from comparison algorithm 103
Table 25: Example of priority computations† 106
Table 26: Machines for example of `Priority_List_Algorithm` 113
Table 27: Tooling for example of `Priority_List_Algorithm` 113
Table 28: Initial tooling setups for example of `Priority_List_Algorithm` 113
Table 29: Example results for `Priority_List_Algorithm` 114
Table 30: Compare results for group 1 119
Table 31: Compare results for group 2 119
Table 32: Compare results for group 3 120
Table 33: Priorities for resetting available machines 121
Table 34: Heuristic changeover results 123
Table 35: CPLEX changeover results for Model 1 124
Table 36: Percentage difference between CPLEX and heuristic solutions for Model 5† 125
Table 37: CPLEX changeover results for Model 6 126
Table 38: Percentage difference between CPLEX and heuristic solutions for Model 6† 127

List of Figures

Figure 1: A sample scheduling for multipass problem.....33

Figure 2: Sample output from single-pass algorithm 42

Figure 3: First time a machine finishes its assinged lots 50

Figure 4: Second time a machine finishes its assigned lots.....50

Figure 5 Schedule derived from `Multipass Algorithm` for example ...51

Figure 6: Flowchart for `Changeover_Algorithm`.....53

Figure 7: Sample results of Phase I 83

Figure 8: Sample results of Phase II.....84

Figure 9: Sample results of Phase III.....86

Chapter 1: Introduction

The technology used to fabricate, assemble and test semiconductor devices is the most complex and expensive in the world of manufacturing. Although the processes differ by product and company, the general approach can be divided into two major phases: front-end operations, referred to as wafer fabrication and wafer probe (Mönch et al. 2011), and the back-end operations, known as assembly and test (AT). Both phases consist of a series of intricate steps that utilize sophisticated equipment and technology. Front-end operations start with a raw disc-shaped wafer typically made out of silicon and end with electronic circuits in the form of chips on the wafer (details are provided by Mönch et al. 2011). In the second phase, the wafers are grouped into lots and delivered to an AT facility. Assembly consists of four steps: (i) "die preparation" where each wafer is sawed into individual integrated circuits (IC); (ii) "die attachment" where the ICs are attached to a support structure or package (e.g., lead frame); (iii) "IC bonding" where the ICs are connected to the electrical contacts of the package – this allows interaction with the outside world; and (iv) "IC encapsulation" where each device is encased in a plastic molding compound or ceramic material, giving physical and chemical protection to the circuits.

Next, the packaged devices are put through a series of tests to ensure that their circuits are working properly. The testing is performed on a variety of machines that are programmed to check different operating specifications including functionality, voltage, current and timing. If no shorts or faults are discovered, the devices are shipped to the customer or placed in finished goods inventory. For more detail on back-end operations, see, e.g., Ovacik and Uzsoy (1997).

In the semiconductor industry, on-time delivery is a critical component of customer satisfaction. Failing to meet promised due dates may incur steep financial penalties and result in an unrecoverable loss of business. Therefore, careful planning at each stage in the supply chain has become the norm, especially since wafer fabrication and AT operations are most often performed in different facilities in different countries. In this research, we investigate the latter, which is arguably the more critical of the two manufacturing phases being the last link in the supply chain. The goal of the research was to develop new mathematical models that could be used to provide production plans for up to a week at a time, given a backlog of work in process (WIP) in the form of lots, and a demand targets for a subset of devices. In formulating our models, the following four objectives were considered: minimizing the shortage of *key* devices, maximizing the weighted throughput of lots processed, minimizing the number of machine used, and minimizing the makespan. The work was done in conjunction with a leading semiconductor manufacturer who has several AT facilities in Asia.

Back-end operations are performed using families of parallel machines configured with various types of tooling. Each device follows a predetermined sequence of steps called a *route* and is processed in batches or lots without interruption. After finishing the current step, a lot may return to the same machine or to a different machine for the next step, giving rise to what is termed reentrant flow (Graves et al., 1983). What needs to be decided is how to set up each machine with tooling to run at a specific temperature, which lots to assign to each machine, and how to sequence the lots once they are assigned. The difficulty in making "optimal" decisions is a consequence of the large number of candidate machine-tooling-temperature-lot combinations, and the fact that it is necessary to plan for each step in the route and not just the next one in the sequence. This

is what we call the *multi-pass* aspect of the problem or, as mentioned, the reentrant aspect of the flow.

The first attempt to model this problem was undertaken by Deng et al. (2010) who developed a greedy randomized adaptive search procedure (GRASP) to solve the single-pass version of the problem that only took into account the upcoming step. The model developed by Deng et al. (2010) did not give the multipass requirements much consideration. In fact, each lot must undergo a series of operations, defined by its *route,* that are spaced no more than a predetermined number of minutes apart. When creating schedules, it is therefore necessary to look ahead and take into account machine and tooling requirements for all operations in a route, and not just the current one. An effective way to deal with this issue is to give higher priority to those lots with several passes remaining over those having only one pass to go. Nevertheless, this greatly increases the complexity of the problem and challenges our ability to develop efficient algorithms

The main contribution of this research has been to introduce two different schemes that take into account the multipass requirements omitted from the original model by Deng et al. (2010) and to design solution algorithms that can be implemented in an industrial setting. Related contributions include the design of real-time support procedures and test results using actual facility data to validate all algorithms.

The first scheme for the multi-pass version is a three-phase heuristic. In the first phase, an attempt is made to schedule as many lots as possible in accordance with the four objectives mentioned above. Only the upcoming step is considered for each lot. In the second phase, subsequent steps in the route are considered and the corresponding lots are assigned to machines if the tooling and temperature are compatible, and capacity still exists. In the third phase, machines are reconfigured in an effort to exploit their unused

capacity. Nevertheless, suboptimal solutions are almost always the result because schedules are constructed without regard to downstream steps in the routes. This was born out in our latest testing.

The second scheme is a three-phase optimization-based approach that is aimed at correcting the shortsightedness of the first scheme. The new methodology is centered on solving two mixed-integer programming (MIP) models; the first is used to assign tooling and lots to machines, and the second to sequence the lots. This constitutes Phases I and II. In Phase III, a changeover procedure is applied to make use of any remaining machine capacity. Compared to the first scheme, the second methodology considers all steps in a route when planning the machine setups and lot assignments, rather than just the upcoming step.

The models are now being tested at several facilities of the collaborating company in the Far East to examine its effectiveness for capacity planning, estimating daily completions, and giving direction to shop floor personnel for resetting machines. Under ideal conditions, models like this provide an indication of maximum throughput (e.g., see Freed et al. 2006). However, if the initial state of the system is noticeably different than the state recommended by the model, it is not clear how to make the transition as opportunities arise over the day. Equipment failures, insufficient tooling, the arrival of hot lots, and limited manpower invariably undermine the best of plans. As a consequence, guidance is needed to handle disruptions and changing circumstances.

In practice, one of the biggest obstacles to realizing the recommended state by the end of the planning horizon is crew availability. Changeovers take anywhere from 30 to 60 minutes, so only a handful can be performed each hour. To help shop floor personnel make the most productive decisions under these circumstances, we have developed a real-time dynamic procedure for prioritizing machine changeovers and lot

assignments. The methodology uses the maximum-capacity solution obtained from the planning model as a target, and constructs a list of recommendations based on available tooling, lot weights, processing requirements, and remaining machine capacities. The driving force is the perceived value of the lots in WIP, which is measured by several parameters including their age, size, planned cycle time, upcoming operation, and the number of similar lots. An aggregate benefit measure is computed for each lot and used to evaluate all feasible machine setups. This allows us to construct a priority list for resetting one or more machines. The procedure is intended to be run every few minutes or whenever machines and crew are idle.

The next section presents a literature review. Section 3 introduces the statement of the AT problem with multipass. Section 4 provides mathematical formulations of basic AT problems and a GRASP to the basic AT. Section 5 introduces the multipass AT problem. Sections 6 and 7 propose two different modeling and solution methodologies for multipass AT. Section 8 discusses real-time decision support. Section 9 presents opportunities for future work.

Chapter 2: Literature Review

AT facilities with multipass requirements can be viewed as reentrant shop flows where a job may return to a machine several times before its completion. The concept of reentrant flow was introduced by Graves et al. (1983) motivated by production scheduling for a plant manufacturing integrated circuits. Allahverdi et al. (1999) undertook a comprehensive review of research directed towards the solution of static scheduling problems involving setup decisions. Gupta and Sivakumar (2006) conducted a survey of job shop scheduling methods in semiconductor manufacturing. Lin and Lee (2011) updated their findings, summarizing models, solution methods, and applications appearing in the literature through 2009 that focused on reentrant flow in front-end operations. The reentrant shop investigated in this research is based on back-end operations but shares many characteristics of the job shop version of the problem described by Lin and Lee (2011). However, when multiple passes are considered, as they are here, the scheduling becomes more difficult since it is necessary to take into account tooling constraints and temperature requirements for each job or lot. During the past thirty years, a large number of solution methods have been proposed. To provide structure to our review, we use the following classification scheme: dispatching rules, mathematical programming, and heuristics.

Dispatching rules are prevalent in both the scheduling and industrial practice literature. Although such rules are computationally efficient and easy to implement, they are often myopic and give poor results. In an early survey, Panwalker and Iskandar (1977) presented over 100 dispatching rules, which they classified as either local (e.g., first-in-first-out) and global. Examples of the latter are the minimum inventory variability scheduler proposed by Li et al. (1996) and the next arrival control heuristic from Fowler et al. (1992). The key weakness of local dispatching rules is that they

cannot handle reentrant flows. Global rules overcome this difficulty to some extent, but they can be complicated to implement and rarely produce optimal solutions, especially when the problem is defined by multiple objectives. Taking an integrated approach, Dabbas and Fowler (2003) combined a variety of local dispatching rules to sidestep the weaknesses of global rules. Using simulation, they demonstrated that their algorithm performed better than any common single dispatching policy with respect to on-time delivery, variability of lateness, and mean cycle time metrics. In related work, Choi et al. (2011) suggested a real-time decision tree based dispatching rule selector. In the first step, a real-time scheduler determines when to choose a new dispatching rule; in the second step, a new dispatching rule is chosen using decision trees. The methodology was demonstrated using data from a thin film transistor-liquid crystal display manufacturing line, which is a typical reentrant flow shop.

With respect to mathematical programming methods, the use of scheduling models and decomposition techniques have figured prominently among researchers. Considering scheduling methods first, Graves et al. (1983) modeled a wafer fab as a reentrant flow shop with the objective of minimizing the average throughput time subject to meeting a given production rate. They developed a cyclic technique under high volume requirements for scheduling jobs with similar or identical routings which included multiple tasks at one or more facilities. The idea was to reduce the problem size by creating cycles over the planning horizon; however, computational feasibility still proved to be a challenge. Kubiak et al. (1996) designed a two-step approach to minimize total completion time in a reentrant shop with one hub machine that jobs enter multiple times. They proved that the shortest processing time rule produced optimal schedules under the assumptions that the hub machine is the bottleneck and that the processing times of jobs on that machine are at least as great as on any other machine

(this is referred to as the *hereditary order*). Relaxing either assumption did not allow for efficient algorithms. Zhang et al. (2007) proposed a two-level hierarchical capacity planning framework to reconfigure kit components in AT operations. The first level focused on midterm planning while the second level created executable plans for individual facilities. The authors also proposed a MIP for the first level problem. The methodology was successfully applied at one of Intel's AT sites resulting in an annual $10 million saving in the purchase of kit components.

Although the literature on parallel machine scheduling is also vast, there has been little published research that looks at machine-tooling combinations, and virtually none that considers temperature requirements – a unique aspect of our problem. Quadt and Kuhn (2009) investigated at a simplified version of the AT planning problem and developed a MIP formulation based on capacitated lot-sizing models with backorders and setups. They assumed that the machines could be grouped by family so a separate scheduling problem could be solved for each. This eliminated the need to treat the machines as nonhomogeneous and greatly reduced the size of the original problem. Nevertheless, even without tooling and temperature considerations the resultant MIPs could not be solved exactly, so heuristics were used. Similar work was undertaken by Chen and Chen (2008), Chung et al. (2009), Jia and Mason (2009), Kang and Shin (2010), Pfund et al. (2008). The largest parallel scheduling instances that can be solved to optimality contain up to a half-dozen machines and 30 jobs (e.g., see Bard and Rojanasoonthon 2006).

Decomposition techniques are usually applied with the scheduling model together. Based on the work of Ovacik and Uzsoy (1994) and Ovacik and Uzsoy (1997), Demirkol and Uzsoy (1997) examined the performance of schedules obtained by decomposition aimed at minimizing the maximum lateness. One observation from their

experiments was that minimizing the maximum lateness leads to good solutions for other objectives like minimizing the makespan. In the facility they modeled, the final set of operations included test, brand and burn-in, each performed at different workcenters, but always using the same family of machines at each. A complicating feature of the problem was sequence-dependent setup times due to the fact that lots could be tested at various temperatures. Branding took place at a common workcenter, after which some lots required an additional operation that was performed at the test workcenter. These flow restrictions contrast with ours where a job may undergo many operations as part of the testing regimen, sometimes returning to the same machine multiple times, or perhaps visiting a different machine at each step in its route. Demirkol et al. (1995) proposed a procedure that decomposes job shop scheduling problems into workcenters consisting of groups of identical machines. They schedule the workcenters one by one in decreasing order of importance. However, their analysis was limited to the post-burn-in segment of the final testing phase, which is just one of the steps in AT operations. Knutson et al. (1999) investigated a problem in which lots in an AT facility were formed to match the size of customer orders. They assumed that all lots consisted of the same type of chip and that yield losses were zero. The planning horizon was set to one day and any delivery tardiness or over supply was treated as a penalty. The problem was formulated as a nonlinear integer program with three objectives: maximize the satisfaction of customer demand, minimize the number of die (chips) sent to the warehouse, and minimize delivery tardiness. To find solutions, a two-stage decomposition approach was used. Demirkol and Uzsoy (2000) proposed a decomposition method for minimizing the maximum lateness in reentrant flow shops with sequence-dependent setup times. Using data from a wafer fab they were not able to improve upon results obtained with simple dispatching rules, calling into question the computational burden accompanying their

scheme. The problem proved harder than expected. In a similar vein, Bard et al. (2010) presented a decomposition algorithm for production planning in a high volume fab that uses quarterly commitments to define daily target outputs. The objective was to minimize the sum of the deviations between the target outputs and finished goods inventory. The planning horizon was broken into weekly subproblems that could be solved to optimality within a few minutes. A post-processor was then applied to smooth production and to increase machine utilization. Extensive testing on realistic size instances spanning 4–13 weeks showed that the proposed scheme could find solutions quickly, and was much more effective than Lagrangian relaxation or Benders decomposition

Regarding heuristics, Pearn et al. (2004) proposed a three network-based heuristics and report on a case study for the scheduling problem associated with the final testing of integrated-circuits, which is a generalization of the classical reentrant flow batching problem as well as the identical parallel machine problem. In their model, jobs were clustered by product type and processed on groups of parallel machines at each step in their route. Processing times were a function of the product type, and the machine setup times were sequentially dependent. The objective was to minimize the total machine workload without violating due dates.

With respect to metaheuristics methods, Chen et al. (2008) proposed a genetic algorithm and Chen et al. (2008) developed a hybrid tabu search procedure to minimize makespan. In the reentrant flow shop they investigated, all jobs have the same routing through the machines, and the same sequence is traversed several times to complete the jobs. Song et al. (2007) applied ant colony optimization to reduce the conversion time of a bottleneck machine during AT operations. Three objectives were also investigated: minimize unfilled customer demand, minimize total number of machine changeovers, and

minimize total changeover time. Their algorithm was successfully applied at an Intel AT facility and achieved changeover time reductions of up to 20% compared to the manual approach then being used.

The work proposed in the first scheme of this research derives from the model and solution procedure in Deng et al. (2010). Their GRASP was designed to examine a diversity of machine-tooling combinations and lot assignments over many iterations. Tests were conducted using data from the collaborating semiconductor manufacturer with the results showing that the GRASP achieves high quality solutions comparable to those obtained with CPLEX in often half the time. However, what was missing from their work was the inclusion of constraints and logic for scheduling more than a single operation for a lot at a time

The methodology that is described in the second scheme of this research decomposes the original multipass scheduling problem into assignment and sequencing problems. Pinto and Grossmann (1998) provided an overview of various assignment and sequencing models used for chemical process scheduling. They mainly focused on single-machine assignment models in which the assignment of jobs to machines is known, and multiple-machine assignment models based on time slots and event times. The approach that was common to most of the studies they reviewed required an initial specification of the number of time slots for each machine in the facility; however, for the multiple machine assignment problem, they observed that there is no efficient way to calculate the exact number of time slots required to accommodate all the jobs within the given time horizon

Regarding the assignment problem, Mazzola and Neebe (1986) developed a branch-and-bound algorithm and a heuristic for finding solutions when side constraints are present. They provided test results for both procedures for over 400 randomly

generated instances. Bard and Wan (2006) constructed a multi-commodity network model for assigning tasks to postal service workers during their daily shifts. They developed a delayed idle period assignment algorithm in which idle periods were treated implicitly and idle time was scheduled in a post-processing phase. In addition, they designed a decomposition algorithm that divided a week into 7 daily problems, and applied tabu search to each to find solutions. In their problem, the workforce was homogenous, the number of time periods was predetermined, and there was no sequencing requirement among tasks.

In the context of wafer fabrication, Kim et al. (2008) investigated a process by which lots are assigned to orders with the objective of meeting due-dates. They proposed three soft pegging strategies under which the assignment of lots to orders could be changed during the production period. Discrete event simulation was used to evaluate the performance of the three strategies.

With respect to sequencing models, Lee and Lee (2006) designed a Petri net based- method for single-armed cluster tools with various reentrant wafer flow patterns, which drove a MIP model that was used to find an optimal sequence for a given wafer flow pattern. Denton et al. (2007) presented a two-stage stochastic recourse model and some practical heuristics for computing operating room schedules that hedge against the uncertainty of surgery durations. They focused on the simultaneous effects of sequencing surgeries and assigning start times. Hwang and Sun (1997) investigated a problem of finding a production sequence of the jobs to minimize makespan. They formulated the problem as a general two-machine flow shop with a set of job precedence constraints, and developed an exact solution procedure based on a modified dynamic programming approach. A small numerical example highlighting their methodology was presented but no numerical tests were conducted.

Other than our prior work, little if any published research exists on the reentrant flow, machine-tooling-lot assignment problem that goes beyond traditional job-shop scheduling. In undertaking this study, we viewed the computational challenge as one of obtaining high-quality solutions quickly. As a practical matter, shop floor planners at the collaborating company were not willing to wait more than a few hours for results

Chapter 3: Problem Description

3.1 EXPLANATION OF TERMS

AT operations are performed on a variety of machines that must be set up with the appropriate tooling to run under a designed temperature, sometimes referred to as *certification*. Each machine belongs to a *machine family*, which contains a multiple number of identical instances. The same is true for the tooling, which is categorized by family type. During machine setup, tooling is placed on the machine and the temperature is adjusted accordingly. This takes a certain amount of time and requires at least one person to perform the basic operations. Therefore, it is desirable to maintain the same setup for as long as possible, only considering changeovers when the WIP is exhausted. To clarify the terminology, we say that a *tooling setup* is a specific number of tooling pieces from one or more tooling families to be run under a designated temperature. Note that a machine may only be compatible with a subset of tooling families and a subset of temperatures. Also, tooling may only be compatible with a subset of temperatures. The set of temperatures considered in this paper is {1 = low, 2 = medium, 3 = high}, which is sufficient for most situations.

Each machine can not only be set up once during the planning horizon to operate at one temperature, but also be re-setup after the machine finishing all the lots assigned to it. That is, if machine m is set up with tooling configuration λ_1 under temperature τ_1, and assigned a set of lots I_1, then after finishing lots I_1, it can run with another tooling setup λ_2 under another temperature τ_2 to process another lot set I_2 later in the planning horizon, when τ_2 is feasible for configuration λ_2.

An individual unit undergoing AT operations is referred to as a device. Homogeneous wafers containing the same device are grouped into lots and go through the AT facility as a batch. Some lots contain critical devices that are given the highest

priority to ensure that promised delivery dates are met. These devices are defined as key devices, and lots containing them are called key lots. For each key device there is a minimum production target for the planning horizon. Failure to meet the targets results in large penalties. However, once the minimum target is achieved, lots with key devices are redefined as regular lots and only prioritized by their weight, an input parameter that depends on how long the lot has been in process and its relative importance. Note that different lots may contain the same device but vary in size. In our data set, for example, both lots 4000654 and 4000655 contain device XPS54386PWPR, but the quantity in lot 4000654 is 8640 and that in lot 4000655 is 3564. As mentioned, each lot has its own weight which is specified in the input file ”wip.csv.”

The age of a lot is the current time minus the time it entered the facility. Each lot has a planned cycle time (CT) that is constantly compared to its age, as measured by the time it enters the facility. Age, and planned and cumulative CT are used in part to determine a weight that reflects the urgency with which a lot should be included in the schedule. Two lots may consist of the same device but differ in chip count, age, and upcoming step, and so will have different weights. Lots are assigned a value that depends on their age and remaining cycle time in the facility. Regardless of the weight or designation, though, all devices in a lot must be fully processed at each step without preemption, but can be buffered between steps.

Regarding the flow, a device needs to undergo a predetermined sequence of operations that are regarded as a *route*. There is a one-to-one relationship between a route and a device. Each operation in a route is referred as a step or a *pass*, specifying which combinations of machine and tooling setups can perform this operation on this device. Each pass has a unique internal id referred to as a *logpoint*. For example, the respective logpoints for first and second passes of device SN0806054PWPR are 7100 and 7121. As

seen, the logpoints of two subsequent passes may not be consecutive. Although each device has a unique route, each step in the route may be performed by different machines and setups. We refer to these alternatives as *subroute.* For example, for the first pass of device SN0806054PWPR, there is preferred setup and four alternatives. The main setup uses machine family ETS-0-64 and the alternatives use machine families ETS-1-64, ETS-1-128, ETS-1M-64, and ETS-2-64, respectively. In this example, all subroutes share the same tooling family, number of required tooling pieces, and temperature.

For the devices manufactured by the sponsoring company, a route may contain anywhere from 1 to 5 steps. We refer to any lot with more than a single step remaining in its route as a multipass lot. Because AT facilities are typically arranged as a job shop and a route may return a lot to the same machine several times, the majority of AT operations result in reentrant flow. At time zero, lots may either be in process or queued up on the shop floor waiting for a machine to become free. All such lots are referred to as first-pass lots regardless of the current step in their route. Once the current step is completed, the lot becomes a second-pass lot, with the same logic applying to subsequent steps. As a consequence, when a lot is being scheduled, all future steps in its route must be taken into considerations. This leads to the concept of a virtual lot. To illustrate, assume that lot 4000654 containing device XPS54386PWPR is in its first step at logpoint 7100. Also assume that its route contains a total of two steps. Thus the second pass of lot 7100 with logpoint 7121 is a virtual lot that must be taken into account when developing a production plan. To reiterate, at the start of the planning horizon, the second step in the route of device XPS54386PWPR is viewed as a virtual lot and only becomes a “real” lot when its first step is finished. Again, each pass or step may call for different machines and tooling setups, adding greatly to the complexity of the problem.

In the basic AT problem, each operation is treated as independent of the others, thus allowing the corresponding problems to be solved separately. As such, the discussion in the basic AT problem relates to an individual operation rather than the AT facility as a whole. For an incoming lot, a particular subroute must be selected when there is more than one option. Each subroute specifies the eligible machine family, the tooling requirements, the processing rate, and the operating temperature. Once a subroute is selected for the upcoming operation, the lot is assigned to one of the machines in the specified family and the required tooling pieces are installed. Each assigned lot is processed completely without preemption.

At the start of the planning horizon, a machine may already be set up with tooling and processing some lot that is defined as an *initial lot*. In such cases, we also use the terms *initial machine* and *initial tooling*. The machine, tooling, lot, route and initial machine information are recorded in the input files "machines.csv," "tooling.csv," "wip.csv," "route.csv" and "initialsetup.csv," respectively. A detailed description of these files can be found in Chapter 5.

3.2 PROBLEM STATEMENT

Using the above concepts and terms, we now define the multipass scheduling problem: For a given set of machines, tooling, lots, and route table, we wish to decide which machines to set up, which tooling and temperature levels to assign to each machine, which lots to assign to which machines, and how to sequence the real (and virtual) lots to optimize four prioritized objective function components. When no additional lots can be processed with the derived machine-tooling assignments but time still remains in the planning horizon, we wish to determine how best to change over the machines to exploit the unused capacity.

The full objective function is the sum of the following four terms: the weighted shortage of key devices, the weighted sum of lots processed, the number of machines used, and the makespan. The objective is to minimize the first, third and fourth terms and to maximize the second. The constraints can be divided into four categories. The first category is associated with resource availability and limits the different machine-tooling assignments to the number of machines and tooling available. The second category deals with routing issues. At each step in a route a feasible machine-tooling-temperature combination must be selected. Within the model, alternative subroutes are permitted but penalized to encourage the selection of the main subroute. The third category limits machine time plus changeover time to the length of the planning horizon, while the fourth set of constraints imposes precedent relations on the order the steps in a route follow. The sequencing constraints derive from the multipass nature of AT operations. A lot cannot start its subsequent pass until it finishes its current pass. It will be seen that the virtual lots not only increase the scale of the problem but create modeling difficulties that make good solutions difficult to find.

The real time decision support presents an efficient procedure for prioritizing machine changeovers in a semiconductor assembly and test facility on a periodic basis. A production plan provides guidelines for running a manufacturing facility over the mid-term and is often derived with sophisticated models that may require hours of computation time. When implemented, though, emergency orders, new requests and other disruptions can quickly throw the plan out of alignment. To get back on track, this means constant re-planning and updating. In addition, even if there is no disruption in daily planning, as machines finish their current lots, they need to be reconfigured to match their target setups, derived based on work in process, due dates, and backlogs. The proposed algorithm in Section 9 is designed to achieve this objective and run in real time.

Chapter 4: Basic AT Modeling

In this section, we present the mathematical formulations for the basic AT problem in which it is assumed that all lots require a single operation only. To find solutions, we use an enhanced version of our GRASP (Deng et al. 2010) to handle initial lots and multiple setups.

4.1 INTRODUCTION TO BASIC AT MODELING

The basic model for the AT machine scheduling problem with resource constraints considers at most one setup for each machine. A critical assumption is that all machines are idle at the beginning of the planning horizon, all tooling pieces are detached, and that setup and unloading times are negligible. Nevertheless, even with these simplifications, the corresponding MIP requires a large amount of notation to correctly represent all the machine-tooling-temperature combinations, and from a practical point of view, is intractable. Current technology limits the size of instances that can be solved optimally to less than a dozen machines and several hundred lots.

Indices and sets

D	set of all devices; $j \in D$
K	set of key devices; $k \in K \subseteq D$
L	set of lots in WIP; $l \in L$
Λ	set of feasible tooling setups; $\lambda \in \Lambda$
M	set of machines (each machine is a member of a machine family); $i \in M$
N	set of feasible temperature combinations for machines and tooling; $n, m \in N$
$N(n)$	set of temperature combinations that intersect combination n
R	set of routes (each route is a collection of subroutes that represent a specific machine–tooling–temperature combination); $r \in R$
T	set of tooling families; $t \in T$

TP set of operating temperatures; $\tau \in TP$

$TP(n)$ set of operating temperatures that are elements of temperature combination n

Parameters and data

$b_{\lambda t}$ number of tooling pieces from family t required by tooling setup λ

$n_{mt}^{tooling}$ number of tooling pieces from family t available under temperature combination m

$n_l^{devices}$ number of devices (chips) in lot l

$n_k^{min\_key}$ minimum number of chips associated with key device k required to be processed over the planning horizon

ρ_{ilr} processing rate of lot l on machine i using subroute r (devices per hour)

w_l weight (benefit) associated with processing lot l (function of lot age and the remaining planned cycle time)

ε_k^{short} weight (penalty) associated with shortage of key device k

ε_r penalty for choosing subroute r

ε_M penalty on the number of machines used

ε_T penalty on the makespan

C normalizing constant associated with the various key device shortages

H_i (capacity) number of hours available on machine i over the planning horizon

Decision variables

x_{ilr} 1 if lot l is processed by machine i with subroute r, 0 otherwise

$y_{i\lambda}$ 1 if machine i uses tooling setup λ, 0 otherwise

Δ_k^{short} shortage of key device k

t^{max} latest completion time among all machines processing lots (makespan)

$t_{i\lambda}$ total time used by machine i with tooling setup λ to process lots

$$\text{Minimize} \quad \sum_{k\in K} \varepsilon_k^{short}\Delta_k^{short} - \sum_{i\in M}\sum_{l\in L(i)}\sum_{r\in R(i,l)} \left(w_l - \varepsilon_r\right) x_{ilr} + \varepsilon_M \sum_{i\in M}\sum_{\lambda\in\Lambda(i)} y_{i\lambda} + \varepsilon_T t^{max} \tag{1a}$$

$$\text{subject to} \quad \sum_{i\in M(l)}\sum_{r\in R(i,l)} x_{ilr} \le 1, \quad \forall\, l \in L \tag{1b}$$

$$\sum_{\lambda\in\Lambda(i)} y_{i\lambda} \le 1, \; \forall\, i \in M \tag{1c}$$

$$\sum_{i\in M}\sum_{\tau\in TP(n)}\sum_{\lambda\in\Lambda(i,t,\tau)} b_{\lambda t} y_{i\lambda} \le \sum_{m\in N(n)} n_{mt}^{tooling}, \quad \forall\, t \in T, n \in N \tag{1d}$$

$$t_{i\lambda} = \sum_{l\in L(i,\lambda)}\sum_{r\in R(i,l,\lambda)} \left(\frac{n_l^{devices}}{\rho_{ilr}}\right) x_{ilr}, \; \forall\, i \in M, \lambda \in \Lambda(i) \tag{1e}$$

$$t_{i\lambda} \le H_i y_{i\lambda}, \; \forall\, i \in M, \lambda \in \Lambda(i) \tag{1f}$$

$$t^{max} \ge t_{i\lambda}, \; \forall\, i \in M, \lambda \in \Lambda(i) \tag{1g}$$

$$\sum_{i\in M}\sum_{l\in L(i,k)}\sum_{r\in R(i,l)} n_l^{devices} x_{ilr} + C\Delta_k^{short} \ge n_k^{min\_key}, \quad \forall\, k \in K \tag{1h}$$

$$x_{ilr} \in \{0,1\}, \forall\, i \in M, l \in L(i), r \in R(i,l),$$

$$y_{i\lambda} \in \{0,1\}, \; t_{i\lambda} \ge 0, \; \forall\, i \in M, \lambda \in \Lambda(i)$$

$$\Delta_k^{short} \ge 0, \; \forall\, k \in K, \; t^{max} \ge 0 \tag{1i}$$

Note that indices enclosed in parentheses are used to qualify a set; for example, $L(i,\lambda)$ is the set of lots that can be processed on machine i with tooling setup λ.

As in goal programming, the subscripted weights (w and ε) in (1a) are designed to prioritize the order in which each objective function term is optimized. The first term corresponds to the objective of minimizing the shortage of the key devices and is given the largest weights such that $\varepsilon_k^{short} >> \max\{w_l : l \in L\}$. The second term is aimed at maximizing the total weighted number of lots processed over the planning horizon, which is the second objective. For lot l, w_l = lot age + total planned cycle time – cumulative cycle time. The parameter ε_r in the second term of (1a) is the penalty incurred when (sub)route r is chosen. Both primary and alternate routes exist for most lots. To encourage the selection of primary routes when at all possible, we use the following settings: $\varepsilon_r = 0$ for r a primary route, $\varepsilon_r \in (0, \min\{w_l : l \in L\})$ for r an alternate route.

The third term in (1a) is intended to minimize the number of machines that are set up over the planning horizon before changeovers are considered, and the last term is designed to minimize the workspan. The corresponding weights must be specified to satisfy the following relationships: $\min\{w_l : l \in L\} >> \varepsilon_M >> \varepsilon_T$. When all the weights w_l have the same value and $\varepsilon_k^{short} = \varepsilon_M = \varepsilon_T = 0$, the problem is equivalent to maximizing the throughput. How the weights are calculated in the numerical test is explained in detail in Section 7.3.1.

Before describing the constraints, we would like to clarify the difference between a (sub)route indexed by r, and a tooling setup indexed by λ. A device has a route, which specifies the machine family, the tooling families and number of pieces from each, and the operating temperature. Setups are associated with machines and indicate the actual tooling pieces and operating temperature specified for each. Clearly, there is significant overlap between these two terms but not a one-to-one relationship; several routes can have the same setup because λ is machine independent.

Accordingly, constraints (1b) require that if lot l is assigned to machine $i \in M(l)$, then the tooling associated with one of the routes $r \in R(i,l)$ must be installed on that machine. Lot l cannot be assigned to more than one machine or be given more than one route. These constraints do not require that each lot be processed but the objective function ensures that the as many lots as possible are selected for processing when there are a sufficient number of machines, tooling pieces, and time available.

Constraints (1c) limit each machine i to at most one tooling configuration λ from the set $\Lambda(i)$. When the number of lots $|L|$ is small, or when the available tooling is limited, it may not be desirable or feasible to set up all machines. Because changeovers are not considered at this point, once the tooling-temperature combination λ is selected

for a particular machine, only lots compatible with that combination can be processed on that machine.

Constraints (1d) restrict the total number of tooling pieces assigned to machines from family t to the number of pieces available under temperature combination n. The left-hand side of these constraints counts the number of tooling pieces from family t associated with the choice of $y_{i\lambda}$ over all machines, temperatures in $TP(n)$, and corresponding tooling setups. The right-hand side counts the total available number of tooling pieces in family t under temperature combination n by summing $n_{mt}^{tooling}$ over all combinations $m \in N(n)$. For each $t \in T$, there are $n_{mt}^{tooling}$ tooling pieces that can be used under the n^{th} combination if m shares some temperatures with n. As an example, assume that there are three discrete temperatures, that is, $TP = \{1,2,3\}$, and $n_{mt}^{tooling} = 1$, for all $t \in T$, $m \in N$, and let the set of possible temperature combinations $N = \{\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$. For $n = 4$, for example, the temperature set $TP(4) = \{1,2\}$ and $N(4) = NC(\{1,2\}) = \{\{1\}, \{2\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\} = N \setminus \{3\}$. The right-hand side of (1d) under combination n is then $|N(4)| = 6$ for all tooling families $t \in T$.

Constraints (1e) compute the amount of processing time consumed by machine $i \in M$ under tooling configuration $\lambda \in \Lambda(i)$ when lot $l \in L(i,\lambda)$ is assigned to it. The complementary constraints (1f) ensure that no machines exceed their capacity. Although we don't specify the length of the planning horizon explicitly, it is bounded by $\max\{H_i : i \in M\}$. The next set of constraints (1g) is used to determine the makespan, $t^{\max}$. The hierarchical nature of the objective function, though, does not necessarily lead to the minimum makespan, even when an exact optimum is obtained for the problem. The makespan will be minimal only for the given number of machines required to meet the first three objectives.

Constraints (1h) ensure that as many lots as possible containing key device k are processed, at least until demand $n_k^{min\_key}$ is satisfied. The shortage Δ_k^{short} will be positive if some of the demand cannot be met due to limited resources. In that case, a penalty equal to $\varepsilon_k^{short}\Delta_k^{short}$ is incurred, where $C = \max\{w_l : l \in L\} + 0.1\Sigma_{l\in L}w_l$ is a normalizing constant used to ensure that the left-hand-side coefficients in (1h) are all the same order of magnitude. In (1i), binary restrictions are placed on the x and y variables, and nonnegative restrictions are placed on the remaining Δ and t variables. The solution to (1a) – (1i) provides the target values for configuring the facility over the planning horizon.

4.2 BASIC GRASP

In the original work by Deng et al. (2010), model (1) was solved with a two-level decomposition strategy embedded in a reactive GRASP. Initial lots, setups and the multipass requirements of some lots were not taken into account.

The algorithm was based on the observation that model (1) becomes much easier to solve when the machines setups are given, that is, when the $y_{i\lambda}$ variables are fixed, leaving what we call the *lower level problem* (LLP) in the x_{ils} variables. Although practical instances of LLP can often be solved as an integer program with a commercial code, we took a heuristic approach to avoid dependence on third party software. For the *upper level problem* (ULP), a strategic decision must be made concerning machine-tooling assignments. Rather than sequentially selecting the most beneficial combinations, as gauged by the weighted sum of lots that each combination can process, the $y_{i\lambda}$ variables are randomly chosen in accordance with an adaptive greedy function that self-adjusts to reflect the quality of the feasible solutions uncovered at each iteration. The integer program associated with LLP is then solved as a linear program and the x_{ilr} variables that are 1 in the solution are fixed. The remaining x_{ilr} variables are chosen with

a randomization scheme based again on a function that measures the immediate benefit of assigning lot l to machine i provided its tooling setup λ is compatible with route r. This process is repeated many times allowing for a full exploration of the feasible region. In phase II, a novel linear programming-Monte Carlo-based neighborhood search scheme that makes use of local branching ideas (Fischetti and Lodi 2003) is called to improve the results. The details along with the various pseudocodes are provided by Deng et al. (2010).

4.3 Extended model and extended GRASP

The basic AT model assumes that all machines are idle at the beginning of the planning horizon, all tooling pieces are detached, and that setup and unloading times are negligible. However, these assumptions are too restrictive in practice. In the extended version of the model, initial setups and lot processing are taken into account. As a consequence the available time for machine i, denoted by H_i, is not the entire planning horizon H, but the difference between H and the time required to finish the current lot being processed. The logic in the GRASP is adjusted accordingly. To account for the setup and unloading times, we let τ_l^{load} be the aggregate time to perform these functions for lot l and replace constraint (1e) with

$$t_{i\lambda} = \sum_{l \in L(i,\lambda)} \sum_{r \in R(i,l,\lambda)} \left(\frac{n_l^{devices}}{\rho_{ilr}} + \tau_l^{load} \right) x_{ilr}, \ \forall\, i \in M, \lambda \in \Lambda(i) \qquad (1e')$$

in the extended model. The calculations in the extended GRASP are modified to reflect this change

Chapter 5: Introduction to the Multipass Model

5.1 LOGPOINT AND OPERATION NUMBER

The routing table identifies all the operations that must be performed on each device during assembly and test. The basic unit is called a "step" and corresponds to a combination of a *logpoint* and an operation number. The logpoint is an internal accounting reference in TI's data base system and is typically a four-digit number such as 7100, which corresponds to "final test 1." There may be several operations at each logpoint but in the vast majority of cases, there is only one. Consequently, we use the words "step" and "operation" interchangeably unless there is a need to distinguish them.

Logpoint-operation data are contained in the input files *route.csv* and *wip.csv* which are illustrated in Tables 2 and 3. It is assumed that if a lot is available for processing, it has an entry in wip.csv, which gives its upcoming step, its weight, the associated number of devices, its cycle time (CT), and related information. From the route.csv file we can determine the remaining steps for the device that constitutes the lot. To establish a frame of reference, we say that first-pass lots are those lots whose upcoming step is shown in wip.csv, regardless of specific logpoint and operation number. Higher-pass lots don't yet exist and so can be considered "virtual." That is, a second-pass lot is created only after the upcoming step of the corresponding first-pass lot is completed. This naming convention applies to all subsequent steps

5.2 INPUT FILES

A series of input files are required to run the program that encodes model (1) as well as the procedures described in subsequent sections. Table 1 lists the major files and gives a brief description of each.

Table 1: Name and brief description of primary input files.

File name	Description
input.txt	A configuration file to specify values for the algorithm parameters
key_package.csv	Specify the key packages with the corresponding target outputs
key_pin_package.csv	Specify the key pin packages with the corresponding target outputs
keydevices.csv	Specify the key devices with the corresponding target outputs
machines.csv	Indicates which family each machine instances belong to, specifies permissible temperatures for each operation
machine_hours.csv	Specifies available running time for each machine instance
tooling.csv	Indicates which family each tooling instances belong to, specifies temperatures permissible for each operation
toolingfamily_setuptime.csv	Specify the setup time for each tooling family
route.csv	Specify routes for the devices to be processed
wip.csv	Indicates the number of chips, weights, device category and other related info for the incoming lots
initialsetup.csv	Lists each machine instance, its corresponding family, the tooling installed on it, and the operating temperature in the beginning of the time horizon.

The route.csv and wip.csv files are illustrated in Tables 2 and 3, respectively. Looking at Table 2 we can see that the logpoints for device QPWPRG4 are "7100," "7101," "7102," "7110," and "7112"; a description of each is given in column 4. In Table 3, the first two columns give the lot identifier (id) and corresponding device name. Looking at the first record, observe that lot 263 contains 4806 items each being device QPWPRG4, and its upcoming logpoint is “7100” and hence has four more steps to go. Thus the first pass of lot 263 is "7100," its second-pass lot is "7101," its third-pass lot is "7102," its fourth-pass lot is "7110," and its fifth and final pass lot is "7112." Note that in the fifth row of data in Table 3, lot 329 also contains device QPWPRG4 but its upcoming step is "7102." Accordingly, the first pass of lot 329 is "7102," its second

pass is "7110," and its third and final pass 329 is "7112." Knowing the logpoint information for each device is a prerequisite for multipass modeling and analysis.

Tables 2 and 3 also contain the basic data required to set up and solve our optimization model that will be present in Chapter 6 and 7. Each row in Table 2 corresponds to a subroute-pass combination for a particular route (LTR-T3 in the tables), and contains the step information (logpoint, operation number, description), the processing rate in parts per hour (PPH), the machine family, the tooling family and number of tooling pieces required, and the temperature (temp). The "Subroute" column lists the preferred option (blank) and alternatives that are available for each step and pass. For example, for logpoint 7100, operation 1, there are four options. Each requires the same tooling and temperature, but offers the possibility of four different machines: ETS-0-64, ETS-1-64, ETS-1-128, ETS-1M-64, with the first being preferred. The corresponding four rows define $R(i, l, p)$, the set of subroutes that use machine i to process the p^{th} pass of lot l. Here, $l = 263$ and $p = 1$.

Table 3 defines the WIP at the beginning of the planning horizon. As mentioned, each row corresponds to a particular lot and gives the device name, the quantity of devices in the lot, the value of the objective function weight parameter (w_{lp}), the upcoming step, and cycle time information. Those lots that are running at time zero (current time) can be identified by examining the "Start time" column. A non-blank entry specifies when the lot started processing. The next column gives the machine instance on which it is running, and the last column gives the current time. The difference between the current time and start time indicates how long the lot has been in process. To determine when it will finish, we need to first calculate the total time that the lot requires on the current machine i. Dividing the "Quantity" in Table 3 by the "PPH" in Table 2 gives us the desired result. Now, subtracting the in-process time from the total

time tells us how many hours the lot still needs (call this value ΔH_i), and indirectly, when machine *i* and its tooling will become free. This information is used to update the capacity of machine *i*.

Table 2: Example of a route

Route name	Step name	Step description	Device	Subroute	PPH	Machine Family	Tooling family	Tooling quantity	Temp
LTR-T3	7100	FinalTest1	QPWPRG4		1988	ETS-0-64	Master648	1	2
LTR-T3	7100	FinalTest1	QPWPRG4	alt	1988	ETS-1-64	Master648	1	2
LTR-T3	7100	FinalTest1	QPWPRG4	alt	1988	ETS-1-128	Master648	1	2
LTR-T3	7100	FinalTest1	QPWPRG4	alt	1988	ETS-1M-64	Master648	1	2
LTR-T3	7101	FinalTest2	QPWPRG4		1988	ETS-0-64	Master648	2	1
LTR-T3	7101	FinalTest2	QPWPRG4	alt	1988	ETS-1-64	Master648	2	1
LTR-T3	7101	FinalTest2	QPWPRG4	alt	1988	ETS-1-128	Master648	2	1
LTR-T3	7101	FinalTest2	QPWPRG4	alt	1988	ETS-1M-64	Master648	2	1
LTR-T3	7102	FinalTest3	QPWPRG4		1988	ETS-0-64	Master648	1	3
LTR-T3	7102	FinalTest3	QPWPRG4	alt	1988	ETS-1-64	Master648	1	3
LTR-T3	7102	FinalTest3	QPWPRG4	alt	1988	ETS-1-128	Master648	1	3
LTR-T3	7102	FinalTest3	QPWPRG4	alt	1988	ETS-1M-64	Master648	1	3
LTR-T3	7110	QASample1	QPWPRG4		1988	ETS-0-64	Master648	1	2
LTR-T3	7110	QASample1	QPWPRG4	alt	1988	ETS-1-64	Master648	1	2
LTR-T3	7112	QASample3	QPWPRG4		1988	ETS-0-64	Master648	1	3
LTR-T3	7112	QASample3	QPWPRG4	alt	1988	ETS-1-64	Master648	1	3

Table 3: Portion of WIP file

Lot name	Device	Quantity	Weight	Step name	Planned CT	Cum CT	Lot age (hrs)	Start time	Machine instance	Current time
263	QPWPRG4	4806	1000	7100	15.3	77	83.9			5/24/2010 11:49
275	SWR5111W	5760	500000	7100	18.8	62.6	81.4	5/24/2010 11:36	AMAT19-1	5/24/2010 11:49
275	51116PWPR	5744	500000	7100	0	62.6	81.4			5/24/2010 11:49
299	C5696PNR	523	5000	7100	16.9	60.7	77	5/24/2010 11:31	AMAT505-1	5/24/2010 11:49
329	QPWPRG4	7676	8300	7102	9.3	12.2	21.5	5/24/2010 8:46	AMAT25-1	5/24/2010 11:49
342	TPS65161	8640	4000	7100	15	53.3	68.6			5/24/2010 11:49
347	TPS65161	5759	1	7100	0	0	0			5/24/2010 11:49
378	Q1SO7420Q	271	500000	7141	38.1	56.9	61.8	5/24/2010 9:27	AMAT15-1	5/24/2010 11:49
395	2U54616Q	1960	328800	7124	42.4	5.4	58.8			5/24/2010 11:49
419	160APWPR	4320	1	7100	0	0	0			5/24/2010 11:49
446	TPA0172	6238	3000	7100	16.3	33	49	5/24/2010 11:33	AMAT02-1	5/24/2010 11:49

5.3 AT SCHEDULING FOR THE MULTIPASS MODEL

With these concepts defined in Section 5.1 in mind, we can now define the AT scheduling problem for multipass lots. Given a finite planning horizon (H), a set of machines (M), a set of tooling families (T), a set of temperatures (TP), a set of routes (R), and a set of lots with upcoming steps $\{(l, logpoint(l) : \text{for all } l \in L\}$, we wish to determine an optimal sequence of machine-tooling setups and assignments of lots to machines at each step in their route so that a hierarchical series of objectives are met. In order of priority, the first objective is to minimize the shortage of key devices; the second is to maximize the weighted sum of lots processed; the third is to minimize the number of machine used; and the fourth is to minimize the makespan. At the beginning of the planning horizon, some lots will invariably be in process and hence some machines may already be set up. Depending on the scenario, it may be necessary to take these initial conditions into account.

Figure 1 depicts a typical schedule for three machines. Lots 0 and 8 are on machines 1 and 3 respectively at time zero. Their current logpoints are identified by the 4-digit number in the bar chart. During the planning process lots 1 – 7 are assigned to machines but only lot 3 can begin at time zero; the other six lots must wait until their assigned machines become free. The second pass of lots 1 and 3 are assigned to machine 1. The second pass of lot 6 is assigned to machine 3.

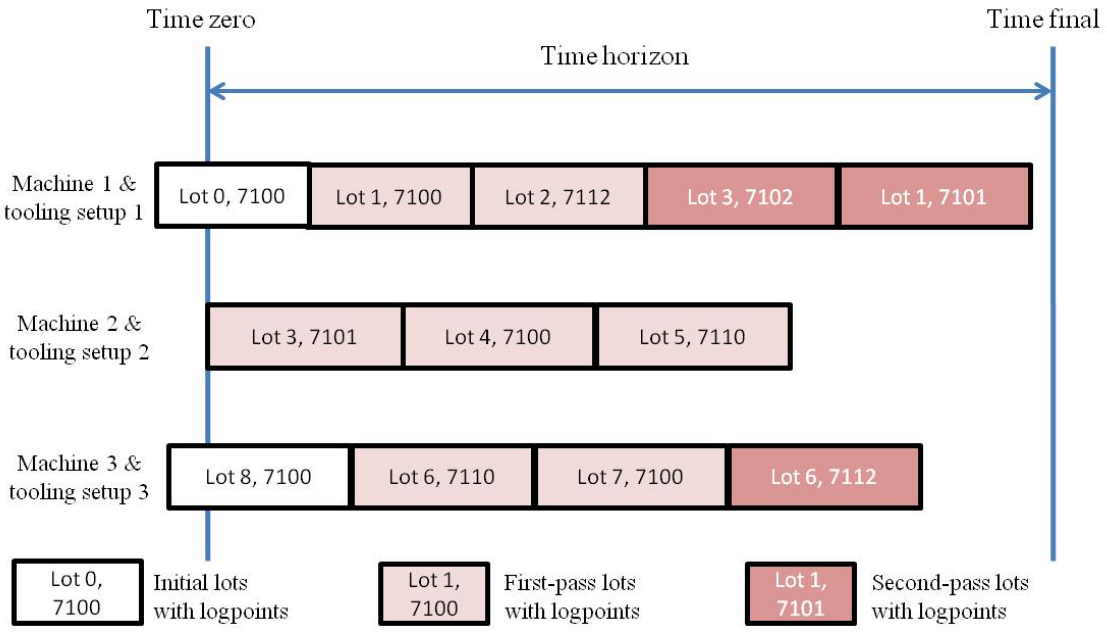

Figure 1: A sample scheduling for multipass problem

To summarize, the scheduling of multipass lots requires the choice of machine-tooling setups, lot assignments, and lot sequences to hierarchically optimize four objectives subject to the following constraints

1. All lots *l* with steps (*l*, *logpoint*(*l*)) must be processed in accordance with their subroutes. At most one machine and one subroute can be chosen for a lot at a particular step.
2. At most one tooling configuration can be installed on a machine at a time although changeovers are permitted.
3. For a given tooling family, the number of tooling pieces in use at any time cannot exceed the number available.
4. The amount of work assigned to each machine cannot extend beyond the planning horizon.

5. Unloading a finished lot and loading the upcoming lot requires a certain amount of time, which is assumed to be constant. In the analysis, 10 minutes is used for the total.

6. The prescribed sequencing of steps for a lot must be maintained; that is, pass p+1 of lot l cannot be started until pass p is finished.

Chapter 6: Multipass Scheduling Scheme I

6.1 MATHEMATICAL MODEL

The full machine setup and scheduling problem cannot be modeled efficiently as a MIP when machine changeovers and lot sequencing considerations are included. An exponential number of logic variables and constraints would be required to keep track of lot sequences and starting times on each machine. Instead, we present a partial model that includes the major components of the problem, and then describe how solutions are obtained that satisfy all the constraints. In the developments, we make use of the following notation.

Indices and sets

D	set of all devices; $j \in D$
K	set of key devices; $k \in K \subseteq D$
L	set of lots in WIP; $l \in L$
$L(j)$	set of lots in WIP containing device j; $l \in L(j)$
$L(i,k,s)$	set of lots containing key device k whose step s can be processed on machine i.
Λ	set of feasible tooling setups; $\lambda \in \Lambda$
M	set of machines (each machine is a member of a machine family); $i \in M$
N	set of feasible temperature combinations for machines and tooling; $n, m \in N$
$N(n)$	set of temperature combinations that intersect combination n
P	set of all possible passes; $p \in P$
$P(l)$	set of passes considered in the planning horizon for each lot l (if a lot is now in its second pass, for example, and a total of four passes are required to finish its processing, then passes 2, 3 and 4 will be considered); $p \in P(l)$

$p(j,l,s)$ pass number corresponding to step s of device j in lot l; $p(j,l,s) \in P(l)$, $j \in D$, $l \in L(j)$, $s \in S(j)$

R set of routes (each route is a collection of subroutes that represent a specific machine–tooling–temperature combination); $r \in R$

$R(i,l,p)$ set of subroutes that use machine i to process the p^{th} pass of lot l

$R(i,l,\lambda,p)$ set of subroutes that use machine i to process the p^{th} pass of lot l with tooling setup λ

$S(j)$ set of all steps in the route for the device j; $s \in S(j)$

T set of tooling families; $t \in T$

TP set of operating temperatures; $\tau \in TP$

$TP(n)$ set of operating temperatures that are elements of temperature combination n

Parameters and data

$b_{\lambda t}$ number of tooling pieces from family t required by tooling setup λ

H_i (capacity) number of hours available on machine i over the planning horizon

$n_{mt}^{tooling}$ number of tooling pieces from family t available under temperature combination m

$n_l^{devices}$ number of devices (chips) in lot l

$n_{ks}^{min\_key}$ minimum number of chips associated with key device k that are required to be processed over the planning horizon at step s

ρ_{ils} processing rate of lot l on machine i using subroute r (devices per hour)

w_{lp} weight (benefit) associated with processing lot l during pass p (function of lot age and the remaining planned cycle time)

ε_k^{short} weight (penalty) associated with shortage of key device k

ε_r penalty for choosing subroute r

ε_M penalty on the number of machines used

ε_T penalty on the makespan

τ_l^{load} unload plus load time for each lot l

Decision variables

x_{ilr}^p 1 if pass p of lot l is performed on machine i using subroute r, 0 otherwise

$y_{i\lambda}$ 1 if machine i uses tooling setup λ, 0 otherwise

Δ_{ks}^{short} shortage of key device k for lots undergoing step s

t^{max} latest completion time among all machines processing lots (makespan)

$t_{i\lambda}$ total time used by machine i with tooling setup λ to process lots

Model

$$\text{Min} \sum_{k\in K}\sum_{s\in S(k)} \varepsilon_k^{short}\Delta_{ks}^{short} - \sum_{i\in M}\sum_{l\in L(i)}\sum_{p\in P(l)}\sum_{r\in R(i,l,p)} \left(w_{lp}-\varepsilon_r\right)x_{ilr}^p + \varepsilon_M \sum_{i\in M}\sum_{\lambda\in\Lambda(i)} y_{i\lambda} + \varepsilon_T t^{max} \quad (2a)$$

subject to

$$\sum_{i\in M(l)}\sum_{r\in R(i,l)} x_{ilr}^p \le 1, \quad \forall\, l\in L, p\in P(l) \quad (2b)$$

$$\sum_{\lambda\in\Lambda(i)} y_{i\lambda} \le 1, \forall\, i\in M \quad (2c)$$

$$\sum_{i\in M}\sum_{\tau\in TP(n)}\sum_{\lambda\in\Lambda(i,t,\tau)} b_{\lambda t} y_{i\lambda} \le \sum_{m\in N(n)} n_{mt}^{tooling}, \quad \forall\, t\in T, n\in N \quad (2d)$$

$$t_{i\lambda} = \sum_{l\in L(i,\lambda)}\sum_{p\in P(l)}\sum_{r\in R(i,l,\lambda,p)} \left(\frac{n_l^{devices}}{\rho_{ilr}} + \tau_l^{load}\right) x_{ilr}^p, \forall\, i\in M, \lambda\in\Lambda(i) \quad (2e)$$

$$t_{i\lambda} \le H_i y_{i\lambda}, \forall\, i\in M, \lambda\in\Lambda(i) \quad (2f)$$

$$t^{max} \ge t_{i\lambda}, \forall\, i\in M, \lambda\in\Lambda(i) \quad (2g)$$

$$\sum_{i\in M}\sum_{l\in L(i,k,s)}\sum_{r\in R(i,l,p)} n_l^{devices} x_{ilr}^{p(k,l,s)} + C\Delta_{ks}^{short} \ge n_{ks}^{min\_key}, \quad \forall\, k\in K, s\in S(k) \quad (2h)$$

$$\sum_{i\in M(l)}\sum_{r\in R(i,l,p)} x_{ilr}^p \ge \sum_{i\in M(l)}\sum_{r\in R(i,l,p)} x_{ilr}^{p+1}, \forall\, l\in L, p\in P \quad (2i)$$

$$x_{ilr}^p \in \{0,1\}, \quad \forall\, i\in M, l\in L(i), p\in P(l), r\in R(i,l);$$

$$y_{i\lambda}\in\{0,1\}, t_{i\lambda}\ge 0, \quad \forall\, i\in M, \quad \lambda\in\Lambda(i);$$

$$\Delta_{ks}^{short}\ge 0, \forall\, k\in K, s\in S(k), \quad t^{max}\ge 0 \quad (2j)$$

Model (2a) – (2j) is an extension of the single-pass model (1) that now takes into account the reentrant flow. The original decision variables x_{ilr} in the basic model have been changed to x_{ilr}^{p} to reflect the pass number p. The first term in the objective function in (2a) is aimed at minimizing the shortages of key devices for each step. The second term represents the weighted sum of all lots processed over all possible passes. Depending on the lot weights, w_{lp}, this term tries to strike a balance between lots undergoing their first pass and those with several passes to come. The last two terms penalize the number of machines used and the makespan, respectively, which are viewed as secondary objectives.

Constraints (2b) ensure that during each pass, at most one machine and one subroute are chosen for a lot, while constraints (2c) limit the number of setups on a machine to 1. Constraints (2d) ensure that no more than the given number of tooling pieces in each family are assigned to machines. The next three constraints (2e) – (2g) track machine usage and make sure that the maximum time available on each machine is not exceeded. When the p^{th} pass of lot l is assigned to machine i (that is, $x_{ilr}^{p} = 1$ for some r) the corresponding time is summed on the right-hand side of (2e). There is no requirement, however, that all passes of a lot be performed on the same machine; in fact, each pass may require a different machine and setup.

Constraints (2h) keep track of the number of key devices processed and along with the first term in (2a) minimize shortages. These are the only demand constraints in the model. Note that the index function $p(k,l,s)$ included in the decision variable x_{ilr}^{p} is needed to convert the step id to the pass number. For the most part, the objective function drives output. It should be pointed out that the minimum requirement for key device k in the original model was given as $n_k^{min\_key}$, without the subscript s. Because each lot typically undergoes many steps, only the requirements of those at their final step

can be specified directly. Output requirements for key devices whose lots are at intermediate steps are a function of the final requirements. By examining all lots that consist of key device *k*, we can derive the appropriate values for their requirements at each step *s* in the computation of Δ_{ks}^{short}. In our implementation, we assume that for a given *k*, $n_{ks}^{min\_key}$ is constant for all *s*.

Constraints (2i) are introduced to partially account for the precedence relations between two consecutive passes of a lot. It ensures that if pass $p + 1$ is scheduled for lot *l* on some machine, then pass *p* has to be scheduled as well; however, it does not guarantee that pass *p* precedes pass $p + 1$ unless they are both assigned to the same machine, and then only implicitly because there are no sequencing constraints. To enforce the timing restriction, it would be necessary to keep track of the sequence on each machine and the time at which each lot finished. As mentioned this would require the addition of an unmanageably large number of new variables and constraints. Finally, variable definitions are given in (2i). For each device *j*, the set of remaining steps in its is *S*(*j*). If $|S|$ denotes the maximum number of steps for all devices, i.e., $|S| = \max_{j \in D} \{|S(j)|\}$, then in total there are $O(|M| \times |L| \times |P| \times |R|)$ variables and $O(2|L| \times |P| + |K| \times |S| + 3|M| \times |\Lambda| + |T| \times |N|)$ constraints.

Initial conditions. Once the updated machine capacities are determined by putting $H_i \leftarrow H_i - \Delta H_i$, we need to decide whether or not machines running initial lots can be reset when they become free. In our original work, this was the assumption we made so the only consequence of initial lots was reduced machine capacity. In an extended version of the model we relaxed this assumption and began with $y_{i\lambda} = 1$ for all machines *i* running lots at time zero, where λ is determined from Table 1 by picking the setup with the largest PPH; however, because the logpoint was ignored we didn't always select the

correct λ. This oversight is addressed in the current work as well as the option for changing over machines after their first setup.

6.2 SOLUTION METHODOLOGY

Upon solving model (2), we have an "optimal" assignment of lots to machines and the setup information for each machine, but not the lot sequences. Nevertheless, there still may be ample capacity remaining on a subset of the machines to accommodate higher-pass lots, even without changeovers. To ensure the attainment of solutions that make the most effective use of the available resources, we have developed a three-phase heuristic. In phase I, the input files are read and a solution is generated for the first-pass lots only by solving the equivalent of model (2) for $p = 1$ using an augmented version of the GRASP in Section 4.3. In phase II, the phase I solution is parsed to identify available second- and higher-pass of lots (a second-pass lot becomes available after its corresponding first-pass lot is finished and so on), and an attempt is made to insert them into existing sequences on the active machines. In phase III, the active machines are reset with different tooling at the time at which they would have finished all their assigned lots, and then assigned additional lots whenever possible. The latter could include unassigned first-pass lots. The goal of phase III is to maximize the utilization of each machine's remaining capacity.

To diversify the search for solutions, randomness is introduced at several points in the execution of the phases II and III to make lot assignments. In the implementation, these two phases are sequentially repeated many times and the schedule associated with the best objective function value is reported as the solution.

An outline of each phase is given below. Those interested in the algorithmic details are referred to the corresponding appendix, which contains most of the

pseudocodes and their description. Structurally speaking, phase I embodies the single-pass algorithm, phase II embodies the multipass algorithm, and phase III embodies the changeover algorithm.

6.2.1 Phase I: single-pass algorithm

The single-pass algorithm was developed from the basic versions of our GRASP that omitted initial machine-tooling setups and simply estimated the time required to finish lots running at time zero. The updated version reads the initial machine-tooling setups as input, calculates the exact time required to finish initial lots (setup times aren't considered because the initial tooling is already on the machine) and then applies the GRASP to get machine-tooling-lot assignments without fixing the initial tooling setups. Not all the machines have tooling on them at time zero. For discussion purposes, machines that do, as indicated in the "initialsetup.csv" file are called *initial machines* while the remainder are called *regular machines.* If an initial machine finishes its initial lot before the start of the time horizon, it will be taken as a regular machine by the single-pass algorithm. In addition, because the pass number of a lot was not considered, modifications to the code were made to take the logpoint and operation number into account. The procedure used for this purpose is given in Appendix A. Output is written to the file "solutions.csv."

A sample output of the single-pass algorithm is depicted in Figure 2. The pair (9,1) in the bar chart means lot 9 is undergoing pass number 1. As can be seen, the first pass of lot 9 is already assigned to machine 1 before the start of the planning horizon. When the schedule is developed lots 1 – 8 are assigned to one of the four machines. Lots 3, 6 and 7 start at time zero, but lot 1 as well as lots 2, 4, 5 and 8 must wait until their assigned machines are available.

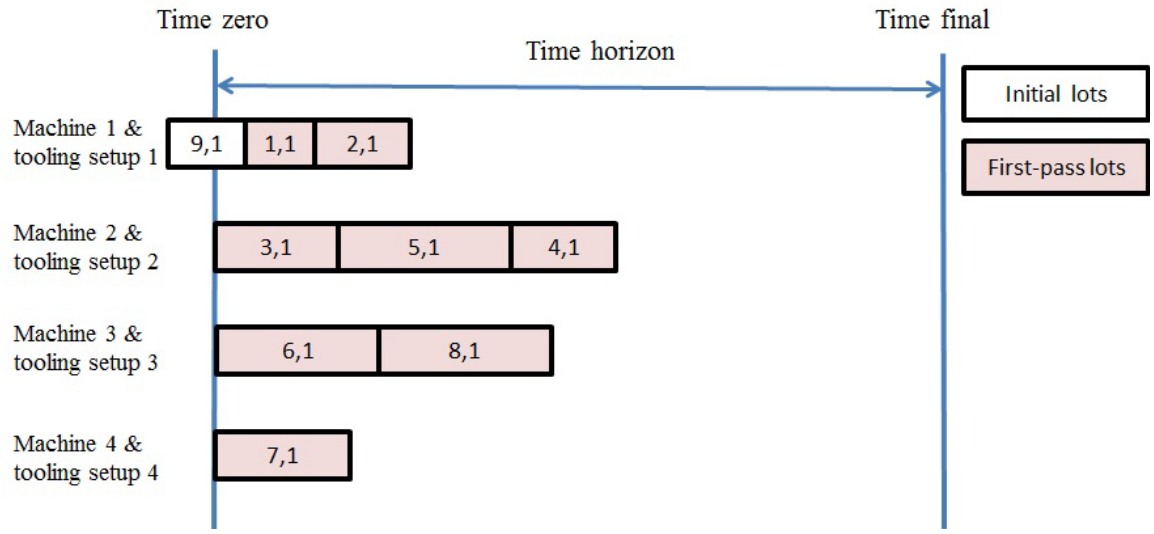

Figure 2: Sample output from single-pass algorithm

6.2.2 Phase II: multipass algorithm without changeovers

The multipass algorithm maintains the machine-tooling setups provided by the single-pass algorithm and tries to insert second- and higher-pass of lots to the existing schedule. Assume that the production target of each key device is the same for each pass and that the weight of a lot for each pass is the same. Also, the hierarchical nature of the objective function remains in force as do the constraints in model (2). A new set of constraints are needed, though, to account for the precedence relations among passes; that is, a higher pass lot cannot be processed until its preceding passes are all finished. In other words, the starting time of higher pass lots cannot be earlier than the completion time of their preceding passes.

The approach we take is to suppress the availability of higher-pass lots until their immediate predecessors are finished. This requires that the list of candidate lots be updated dynamically whenever a machine finishes its assigned lots. Moreover, it is necessary to keep track of the order in which machines become free.

The first step is to initialize all parameters and sets used in the algorithm and to create a candidate list for lots (*CL*), a candidate list for machines in use (*C_M_A_T*), and a candidate list for machines that can process higher-pass lots (*C_M_A_T_2*) from the phase I solution. Based on a ranking of when the active machines in *C_M_A_T* become free, we select the first machine and denote the time when it finishes its assigned lots as the *current_time*. All first-pass lots that are finished at the *current_time*, regardless of which machine they were processed on, are added to the candidate lot list, *CL*, provided that they are not at the last step in their route. An attempt is then made to sequence the lots in *CL* on the free machine. After all possible second- or higher-pass lot assignments are made, the *current_time* becomes the *last_time* and a new *current_time* is determined by identifying when the next machine on the list becomes free. Initially, *last_time* is set to 0.

Lots that finish in the interval [*last_time*, *current_time*] and are at an intermediate step in their route are added to *CL*. After updating *CL*, the candidate lots are ranked based on their benefit value as measured by their contribution to (2a). The next step is to identify which machines are eligible to be assigned the lots $l \in CL$ at the *current_time*. All machines finishing their assigned lots earlier than or at *curren_time* satisfy this condition.

Now, rather than assigning lots with greatest benefit first, a randomized procedure is used to allow us to explore a larger neighborhood. The computations are done hierarchically. In an outer set of iterations, we cycle through each available machine. In an inner loop, we implement a two-step randomized lot-assignment scheme. In the first step a ranked candidate list of lots $CL(i,\lambda(i)) \subseteq CL$ is built for machine i with tooling setup $\lambda(i)$ from the compatible lots in *CL*. In the second step, we randomly choose one lot from top five ranked lots in $CL(i,\lambda(i))$ and check whether the time available on

machine i is sufficient to process it. If so, then the lot is removed from $CL(i,\lambda(i))$ and assign to the machine. The second step is repeated until all the lots in $CL(i,\lambda(i))$ are explored. In the implementation several simplifications were considered, depending on whether any new lots were added to CL at the current iteration. If not, then only the machines available at *current_time* need to be examined.

To describe the algorithm in more detail, we make use of the following additional notation. After giving the pseudocode, we discuss the individual steps. The details of the subroutines used in the algorithm are given in the Appendix B.

Indices and sets

$\Lambda(i)$	set of all tooling setups that are compatible with machine i; $\lambda \in \Lambda$
M^1	set of machines used in the solution of *first-pass algorithm*; $i \in M^1$
L	set of lot ids in "wip.csv"; $l \in L$
L^1	set of combinations of lot id and logpoint that are assigned to some machine in the solution of *single-pass algorithm*; $(l, logpoint(l)) \in L^1$
$L^1(i)$	set of combinations of lot id and logpoint that are assigned to machine i in the solution of *single-pass algorithm*; $(l, logpoint(l)) \in L^1(i)$
$L^2(i)$	set of combinations of lot id and logpoint that are assigned to machine i by the *multipass algorithm* that are not assigned by the *single-pass algorithm*; $(l, logpoint(l)) \in L^2(i)$
CL	candidate list of higher-pass lots: set of combinations of lot id, logpoint of available higher-pass lots, e.g., second-pass lots with corresponding first-pass lots finished, or third-pass lots with corresponding second-pass lots finished. Note for first-pass lots, the completion time for the preceding logpoint is taken as 0; $(l, logpoint(l), \in CL$

$CL(i,\lambda)$ set of combination of lot id and logpoint for available higher-pass lots that can be processed by machine i with tooling setup λ; $(l, logpoint(l), \in CL(i,\lambda(i))$

FL set of combinations of lot id and logpoint associated with finished lots in the solution provided by the *multipass algorithm*; $(l, logpoint(l)) \in FL$

$R(i,\lambda,l,logpoint(l))$ set of subroutes that use machine i to process lot l at logpoint with tooling setup λ

Algorithmic symbols

current_time time at which a machine finishes its assigned lots

last_time most recent time at which a machine other than the current machine finishes its assigned lots

next_logpoint(l) logpoint of the next step in the route of lot l. Note if the second step of lot l is finished, this means the logpoint of the third step; if the current step of the lot l does not started to be processed yet, this just means the logpoint of the current step; $l \in L$

preceding_logpoint(l) logpoint of the preceding step in the route of lot l. Note if the second step of lot l is finished, this means the logpoint of the first step; if the current step of the lot l is not yet finished, this just means the logpoint of the current step; $l \in L$

$\lambda(i)$ tooling setup for machine i used in the solution of *single-pass algorithm*

$t_c(i)$ completion time of the last lot assgined to machine i; $i \in M$

$tcl(l, logpoint(l))$ completion time of a lot l with the step denoted by $logpoint(l)$; $l \in L$

M_Λ_T set of combinations of machine instance id, tooling setup, and completion time of the last lot finished on this machine; M_Λ_T = $\{(i, \lambda(i), t_c(i))$: machine i is set up according to $\lambda(i)$, and finishes its last assigned lot at time $t_c(i)$, $\forall\ i \in M\}$

$C\_M\_\Lambda\_T$ candidate list of machines in use and ranked according to the time they become free; $C\_M\_\Lambda\_T \subseteq M\_\Lambda\_L$

$C\_M\_\Lambda\_T\_2$ candidate machine list for phase II lot assignments; $C\_M\_\Lambda\_T\_2 \subseteq M\_\Lambda\_L$

Input data

logpoint(l) logpoint of the current step in the route of lot l; $l \in L$

$d(l)$ device contained in lot l, as determined from wip.csv file; $l \in L$; $d(l) \in D$

$H(i)$ planning horizon for machine instance i; $i \in M$

Unload_Load_Time time required to unload a finished lot and load the next lot

`Multipass_Algorithm`

Step 0 Initialization

WHILE $C\_M\_\Lambda\_T \neq \emptyset$

Step 1. Rank the elements in $C\_M\_\Lambda\_T$ in ascending order of $t_c(i)$. Choose the first element $(i_1, \lambda(i_1), t_c(i_1))$ in $C\_M\_\Lambda\_T$ and let *current_time* = $t_c(i_1)$.

Step 2. Update candidate lot list *CL*. Run `Building_CL_Algorithm` with *CL*, L^1, *FL*, *tcl*(l, *logpoint*(l)), $l \in L$, *last_time*, and *current_time* as input.

Step 3 Build candidate machine list $C\_M\_\Lambda\_T\_2$ for new lot assignments; set $C\_M\_\Lambda\_T\_2 = \emptyset$.

If (l, *logpoint*(l)) is added to *CL* in Step 2, then

FOR each $i \in M$

If $t_c(i) \leq current\_time$, then

add $(i, \lambda(i), t_c(i))$ to $C\_M\_\Lambda\_T\_2$.

Endif

ENDFOR

Else

add $(i, \lambda(i), t_c(i))$ to $C\_M\_\Lambda\_T\_2$.

Endif

Step 4 FOR each $(i, \lambda(i), t_c(i)) \in C\_M\_\Lambda\_T\_2$

Run `Assign_Lot_Algorithm` with input $(i, \lambda(i), t_c(i))$, CL, FL, R, $tcl(l, logpoint(l))$, $l \in L$ to update $t_c(i)$, CL, and FL;

If no lot is assigned to machine i, then

delete the corresponding mahcine $(i, \lambda(i), t_c(i))$ from $C\_M\_\Lambda\_T$.

Endif

ENDFOR

ENDWHILE

Complexity. The number of iterations is greater than or equal to the number of machines, $|M|$, because the "While" loop has to be executed at least once for each machine. In the worst case, the number of iterations is $|M| \times |L^1| \times max\_pass\_no$, where max_pass_no means the largest pass number of lots available for processing. During each iteration, ranking the elements in $C\_M\_\Lambda\_T$ in Step 2 takes $O(|M| \times \log|M|)$ time; the complexity of the other steps are analyzed in Appendix B. In sum, the worst case complexity of `Multipass_Algorithm` is $O(|M| \times |\Lambda| \times |L| + |M| \times \log|M| + |L| \times \log|L| + |L| \times |R|)$.

The subroutines used in `Multipass_Algorithm` are provided in Appendix B.

Sample output from the multipass algorithm. Assume that we have run `Single-Pass_Algorithm` and obtained the results in Figure 2, and that we are currently running `Multipass_Algorithm`. The candidate machine list $C\_M\_\Lambda\_T$ = {1, 2, 3, 4}. According to Step 1 of `Multipass_Algorithm`, the four machines are ranked based on the time they finish their assigned lots. As shown in Figure 3, Machine 4 is the first to finish. Denote the finish time as 1, so *current_time* = $t_c(4)$ = 1 and *last_time* = 0. At Step 2 we need to build the candidate lot list *CL*, initializing it as the empty set. Lots (9,1), (1,1), (3,1), (6,1), and (7,1) in this order are finished at time 1. Next, we apply the `Check_for_Next-Pass_Algorithm` to these five lots and as a result suppose that only lot (6,1) is at its last step; the others have at least one more step in their route. Thus we insert (9,2) , (1,2) , (3,2) , (7,2) into the candidate lot list *CL* in nondecreasing order of their benefit. Suppose the result is *CL* = {(7,2) , (1,2) , (3,2) , (9,2)}. As Step 3, we need to identify all machines available before or at time 1. Just machine 4 is available so C_M_Λ_T_2 = {4}. Step 4 assigns second-pass lots to the machines in C_$M_\Lambda_T_2$ so at this point we only check machine 4 for which $CL(4,\lambda(4)) = \emptyset$.

To build $CL(4,\lambda(4))$, we need to check each lot in *CL*. Take lot (1,2) as an example. If this lot can be processed by machine 4 with tooling setup 4 and sufficient capacity is available, then it is added into $CL(4,\lambda(4))$. For the example, suppose $CL(4,\lambda(4))$ = {(1,2), (3,2)}. Since we don't have five candidates we randomly select one of the two and assign it to machine 4. We then consider the lots remaining in $CL(4,\lambda(4))$, which is the second one and assign it as well to machine 4. Because at least one lot was assigned, we don't delete any machines from the set $C\_M\_\Lambda\_T$.

The current schedule is depicted in Figure 3. Now, given that $C\_M\_\Lambda\_T$ ={1, 2, 3, 4}, and hence is not empty, we need to repeat Steps 1 to 4 in `Multipass_Algorithm`. In Step 1, Machine 1 is identified as the next machine to

finish its assigned lots (see Figure 3), say, at time 2. As such, we set *last_time* = 1 and *current_time* = 2. In Step 2, lots (2,1) and (6,1) are seen to finish between times 1 and 2, and (6,1) is at its last step. Only lot (2,1) has at least one more step to go, and so is put on the candidate lot set *CL*. Lots (1,2) and (3,2) are already assigned to machine 4 and have been removed from *CL*. Thus *CL* = {(9,2), (7,2), (2,2)}. In Step 3, machine 1 is the only one that finishes its assigned lots before or at time 2. Thus C_M_Λ_T_2 = {1}. In Step 4, we check the compatibility of lots in *CL* with machine 1 and tooling setup 1. As a result, we find that lots (9,2) and (2,2) can be processed by machine 1 with tooling setup 1 so $CL(1,\lambda(1))$ = {(9,2), (2,2)}. Next, the `Assigning_Lot_Algorithm` is run. For this case, lots (9,2) and (2,2) are both assigned during the iteration and so are included in the new solution. Again, because lots were assigned to machine 1 it remains in *C_M_Λ_T*.

Figure 4 displays the schedule after round 1 with the second-pass lots added to machines 1 and 4. The *last_time* is set to 1 and the *current_time* is set to 2, which coincides with the completion of the lots assigned to machine. With the additional lots, machine 4's completion time comes after time 2. The iterations continue until *C_M_Λ_T* is empty, i.e., until all machines finish their assigned lots. The full schedule is shown in Figure 5.

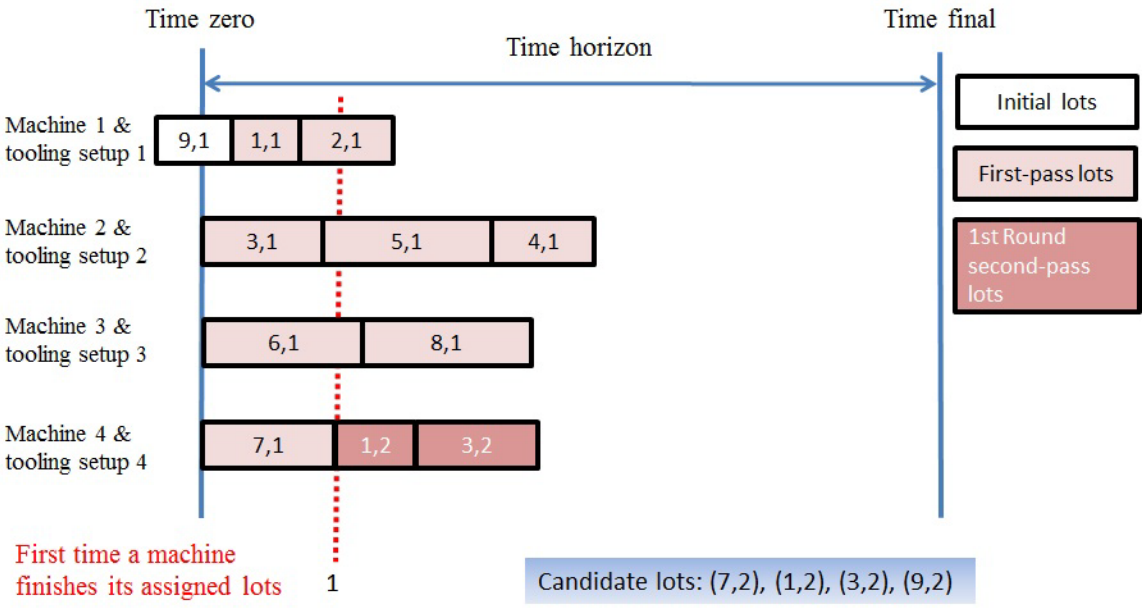

Figure 3: First time a machine finishes its assinged lots

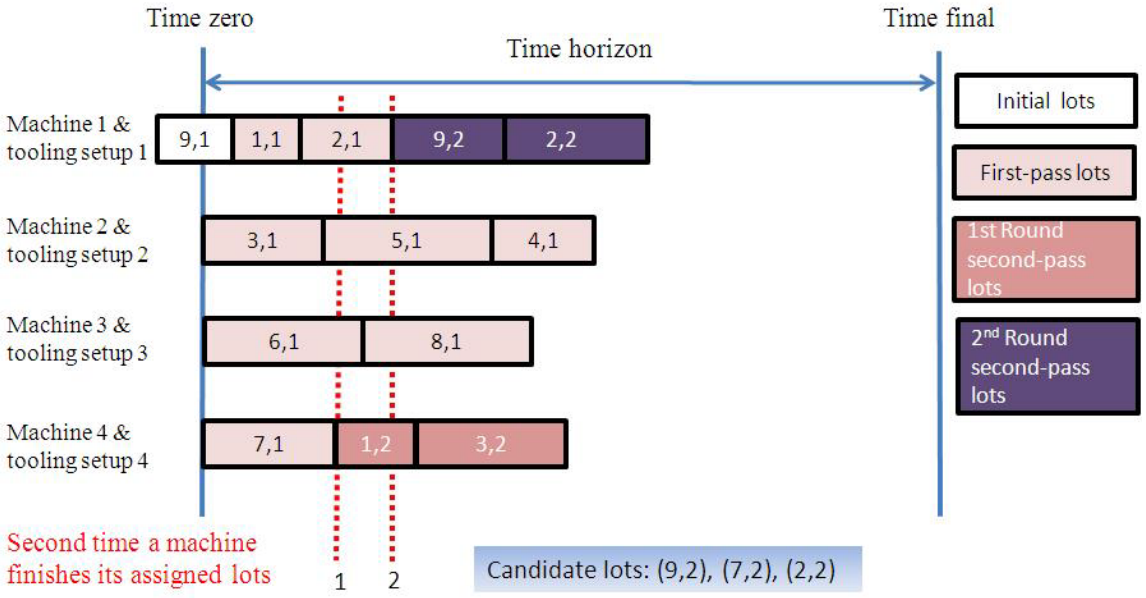

Figure 4: Second time a machine finishes its assigned lots

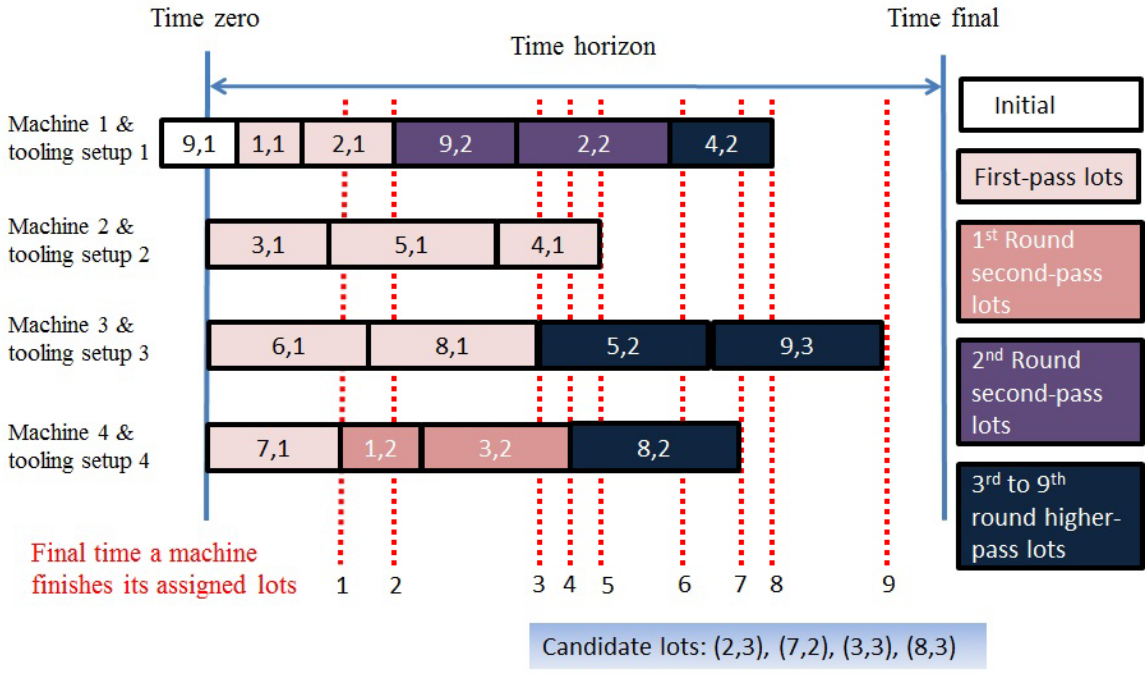

Figure 5 Schedule derived from `Multipass Algorithm` for example

6.2.3 Phase III: changeover algorithm

The single-pass and multipass algorithms make full use of the capacity of the current machine-tooling setups. For those machines whose schedules do not extend to the end of the planning horizon, assigning them additional lots requires a changeover, a process that must not only take into account machine-tooling-temperature-lot compatibility, but also the consistency of logpoint and operation number with the selected subroute, and the multipass sequence constraints.

To ensure that the sequence constraints are satisfied, we take an approach similar to that used in phase II where the candidate lots are dynamically updated. Now, however, all unassigned first-pass lots are included in the set of candidate lots when resetting machines. In addition, the set of available tooling is dynamically updated. The basic idea is to release the tooling on each machine when it finishes the last lot

assigned to it, update the candidate lot list, and try to reset the machine with the available tooling. If additional lots can be assigned to the machine when it is reconfigured with new tooling, then the change is made. Otherwise, it remains free. Whenever there is new tooling available or a new candidate lot, an attempt is made to reset the machine again. The same procedure is applied to all machines in the order in which they finish the lots assigned to them. The flowchart of `Changeover_Algorithm` is shown in Figure 6. All sets, indices and symbols that were defined for `Multipass Algorithm` are used here.

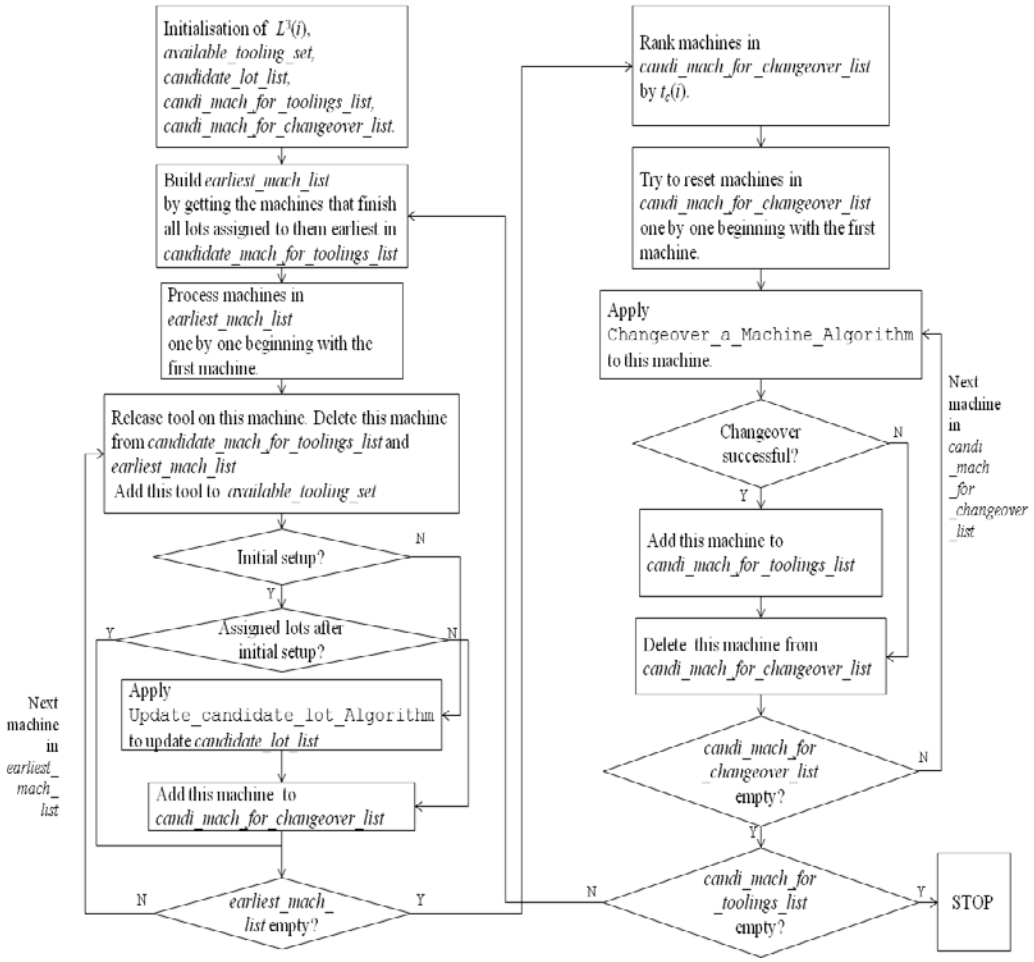

Figure 6: Flowchart for `Changeover_Algorith`

The details of the initialization step, `Update_Candidate_lot_Algorithm` and `Changeover_a_Machine_Algorithm` are given in Appendix C.

6.3 OUTPUT FILES

In addition to the phase I solution.csv file, the two additional files multi_solution.csv and multi_machine_time.csv are generated at the end of phases III. Table 4 lists the principal output files

Table 4: Output data files

File name	Description
multi_solution.csv	Solution generated by the three-phase methodology, includes all machine-tooling setups and multipass lot assignments.
multi_machine_time.csv	Records the operating time of each machine in the file multi_solution.csv.
solution.csv	Solution obtained in phase I by the *single-pass algorithm*; provides input to phases II and III.
time.csv	Records the operating time of all machines after phase I.
keydevice_production.csv	Specifies information on shortage of key devices at the end of phase I.
result_summery.txt	Records the total objective function value and the value of each term for all passes.
long_lot.csv	Lots not processed due to lack of machine capacity

In the multipass solution, it is necessary to list lot starting times to make sure that the starting time of a second-pass lot does not precede the completion time of its corresponding first-pass lot. Examples of the entries in the solution.csv file and multi_solution.csv file for the machine AMAT11-1 are shown in Tables 5 and 6,

respectively. In these tables, the first row contains the column headings. Subsequent rows represent lots and the machine-tooling setups used to process them. The columns labeled “Machine instance,” “Machine family name,” “Lot name,” “Logpoint,” “Lot weight,” “Device name,” “Tooling family name,” “Certification” and Pass no.” are self-explanatory. The column with the heading “Quantity” gives the number of devices in the lot. “Initial lot flag” indicates whether the lot is being processed as time zero. If the entry in this column is “Y,” then the entries in columns “Tooling family name” and “Certification” will be blank. “Setup time” specifies when the machine was or will be configured with tooling to operate at the indicated certification (temperature). “Completion time” indicates when the lot will finish its current pass. In Table 6, “Start time” refers to the time at which the lot on the machine is removed and the next one is loaded. This value is equal to the completion time of the previous lot.

Table 5: An example of "solution.csv" for one machine instance

Machine instance	Machine family name	Lot name	Logpoint	Quantity	Lot weight	Initial lot flag
AMAT11-1	ETS-1-64	4014923	7100	3480	2.51E+06	Y
AMAT11-1	ETS-1-64	4020631	7100	121	60100	N
AMAT11-1	ETS-1-64	4031897	7100	6590	45400	N
AMAT11-1	ETS-1-64	4020626	7100	6363	63000	N
AMAT11-1	ETS-1-64	4033780	7100	7313	1	N
AMAT11-1	ETS-1-64	4009555	7110	4108	72500	N

Device name	Tooling family name	Certification	Setup time	Completion time
XPS40055			11/5/2009 10:32	11/5/2009 10:47
XPA3123 D2	6481146C	1	11/5/2009 10:47	11/5/2009 11:01
TPA3121D2	6481146C	1	11/5/2009 10:47	11/5/2009 15:34
TPA3121D2	6481146C	1	11/5/2009 10:47	11/5/2009 19:59
TPA3124D2	6481146C	1	11/5/2009 10:47	11/5/2009 23:58
XPA3123 D2	6481146C	1	11/5/2009 10:47	11/6/2009 2:32

Table 6: An example of "multi_solution.csv" for one machine instance

Machine instance	Machine family name	Lot name	Logpoint	Quantity	Lot weight	Initial lot flag
AMAT11-1	ETS-1-64	4014923	7100	3480	2.51E+06	Y
AMAT11-1	ETS-1-64	4020631	7100	121	60100	N
AMAT11-1	ETS-1-64	4031897	7100	6590	45400	N
AMAT11-1	ETS-1-64	4020626	7100	6363	63000	N
AMAT11-1	ETS-1-64	4033780	7100	7313	1	N
AMAT11-1	ETS-1-64	4009555	7110	4108	72500	N
AMAT11-1	ETS-1-64	4020631	7110	121	60100	N
AMAT11-1	ETS-1-64	4035295	7110	8754	1	N
AMAT11-1	ETS-1-64	4039963	7100	3625	1.65E+06	N

Device name	Tooling family name	Certification	Setup time	Start time	Completion time	Pass no.
XPS40055			11/5/2009 10:32	11/5/2009 8:35	11/5/2009 10:47	1
XPA3123 D2	6481146C	1	11/5/2009 10:47	11/5/2009 10:47	11/5/2009 11:01	1
TPA3121D2	6481146C	1	11/5/2009 10:47	11/5/2009 11:01	11/5/2009 15:34	1
TPA3121D2	6481146C	1	11/5/2009 10:47	11/5/2009 15:34	11/5/2009 19:59	1
TPA3124D2	6481146C	1	11/5/2009 10:47	11/5/2009 19:59	11/5/2009 23:58	1
XPA3123 D2	6481146C	1	11/5/2009 10:47	11/5/2009 23:58	11/6/2009 2:32	1
XPA3123 D2	6481146C	1	11/5/2009 10:47	11/6/2009 2:32	11/6/2009 2:47	2
XPA3123 D2	6481146C	1	11/5/2009 10:47	11/6/2009 2:47	11/6/2009 8:05	2
XPS65160A	6469171D	1	11/6/2009 8:05	11/6/2009 8:05	11/6/2009 10:19	1

In Table 5, only one setup (6481146C, 1) for machine AMAT11-1 is observed since the entries were obtained from the phase I solution which does not allow for changeovers. In Table 6, we see that the machine was reset once towards the end of the planning horizon when the last device, XPS65160A, was scheduled. Also the machine was assigned two second-pass lots 4020631 and 4035295 while still operating under the first setup. The changeover went from tooling 6481146C under certification 1 to tooling 6469171D under certification 1 at 11/6/2009 8:05. Note that third- and higher-pass lots were considered by the algorithm but only second-pass lots were assigned, either because no candidate third-pass lots were available or there was insufficient time remaining in the planning horizon.

6.4 COMPUTATIONAL RESULT

Testing was done using both real and randomly generated data. The real data were provided by the Southeast Asian AT facility of the collaborating company and consisted of a typical instance, which served as the basis for randomly generating eight additional instances. In all cases, each instance contains 36 machines, 284 tooling pieces from 6 families, and 1 temperature (certification) setting. The number of lots varied as explained below.

The nine problem sets are divided into three groups of three instances each. The first group contains the original instance and two variants, each with 29 initial machines. The second group was derived from the first by randomly reclassifying nine initial machines as free and updating the "wip.csv" file accordingly. These changes result in 20 initial machines and 9 additional lots, or 1045 lots in total that require processing. The third group was created by randomly selecting 10 of the 29 machines as the initial machines, and then for each, choosing a feasible tooling setup from the "route.csv" file and an initial lot from the "wip.csv" file. The total number of lots is 1036 + 19 = 1055.

Problem 1: lots1036_1_group_1

Problem 2: lots1036_2_group_1

Problem 3: lots832_1_group_1

Problem 4: lots1045_1_group_2

Problem 5: lot1045_2_group_2

Problem 6: lot841_1_group_2

Problem 7: lot1055_1_group_3

Problem 8: lot1055_2_group_3

Problem 9: lot800_1_group_3

where the fields X, Y and Z in the term "lotsX_Y_group_Z" mean

X = number of lots

Y = 1: entries in the column labeled “Quantity” (number of devices) in the wip.csv file are used (see Table 5)

Y = 2: entries in the column labeled “Quantity” in the wip.csv file are randomly generated by using the random function: int(rand()*10000+100).

Z = group number 1, 2 or 3

The results obtained with the single-pass algorithm (phase I) were compared to those obtained with the three-phase scheme and reported in Tables 7 to 12. In all cases, the entries under the column heading "Percent diff" were calculated as follows.

Percent diff = 100%(*multipass value* – *single-pass value*) / *single-pass value*

Table 7 compares objective function values. To put the results on an equal footing, the first term in Eq. (2a) was calculated exactly as represented although at most a single step is permitted for each lot in WIP when the single-pass algorithm is run. Thus, large shortages are possible for key devices whose lots are in mid-route. Recall that the objective is minimization.

Table 7: Comparison of single-pass with multipass results

Prob. no.	Single-pass objective value (10^6)	Multipass objective value (10^6)	Percent diff (%)
1	29,232	25,152	-13.96
2	68,956	57,336	-16.85
3	17,214	15,643	-9.13
4	92,193	82,978	-10.00
5	67,718	56,473	-16.61
6	147,463	141,055	-4.35
7	89,597	84,397	-5.80
8	227,673	212,334	-6.74
9	29,085	26,137	-10.14
Avg.	85,459	77,945	-10.40

As seen in the Table 7, the three-phase scheme provides an average objective function decrease of 10.40%. The fairly large values in columns 2 and 3 are a result of the first term in Eq. (2a) which dominates the other three. In general, the weighted sum of key device shortages is a function of the number of key devices in WIP, their target values ($n_{ks}^{min\_key}$), the percentage of lots that have key devices, and the number of steps remaining in their routes. To get a more realistic picture of the advantage provided by the three-phase scheme, it is useful to compare shortages with respect to the last step in each route of each key device.

Recall that the target production of a key device refers to its last step only. To calculate the unweighted shortage it is necessary to subtract the quantity of each key device produced whose last step is included in the schedule from its target value. Of course, only positive shortages are counted. Table 8 presents the corresponding results. As can be seen, the percentage difference is 11.73% on average, a slight decrease.

Table 8: Comparison of total key device shortages

Prob. no.	Single-pass key device shortage	Multipass key device shortage	Percent diff (%)
1	292,660	243,867	-16.67
2	311,648	254,193	-18.44
3	461,952	416,246	-9.89
4	296,820	260,438	-12.26
5	301,360	243,092	-19.34
6	471,142	451,785	-4.11
7	283,539	264,640	-6.67
8	278,081	256,081	-7.91
9	282,356	253,400	-10.26
Avg.	331,062	293,749	-11.73

The first term in Eq. (2a), however, is the weighted sum of key device shortages, not the unweighted sum. Table 9 lists the relevant values for the two approaches along with their percentage differences in the last column. The results indicate that the average weighed sum of key device shortages obtained from the three-phase scheme is nearly 12% smaller than that obtained from the single-pass algorithm. Thus, the actual contribution of the three-phase scheme to the improvement in the solution is measurably higher than reflected in the objective function value in Table 7.

Table 10 specifies the objective function value contributed by lots processed by the single-pass algorithm and the three-phase scheme. The first column identifies the problem number. The second and third columns respectively specify the values of the weighted sum of lots processed by the single-pass algorithm and the three-phase scheme. The fourth column presents the percentage difference between the two latter values. Columns 5 – 8 list the contribution of each pass to the total weighted sum of lots processed by the three-phase scheme. The value in the third column is just the sum of the values obtained from the four passes.

The results in Table 10 indicate that the multipass scheme yields a nearly 40% increase in weighted throughput on average. It can also be observed that the weighted sum of first-pass lots processed is much higher than that for the second-pass lots, which in turn is much higher than for third-pass lots. In five out of the nine problem sets, no fourth-pass lots are processed. This decline is to be expected given the precedent relations between the various passes. Implicitly, first-pass lots are given priority over the higher-pass lots, in terms of weighted throughput.

Table 9: Weighted sum of key device shortages

Prob. no.	Single-pass weighted key device shortage (106)	Multipass weighted key device shortage (106)	Percent diff (%)
1	23,241	19,264	-17.11
2	54,310	43,770	-19.41
3	13,817	12,414	-10.15
4	73,379	64,345	-12.31
5	52,527	42,280	-19.51
6	115,886	111,060	-4.16
7	70,611	65,707	-6.95
8	176,241	161,071	-8.61
9	21,328	19,114	-10.38
Avg.	66,815	59,892	-12.07

Table 10: Weighted sum of lots processed

Prob. no.	Single-pass results (10^6)	Multipass results					
		Total (10^6)	Percent diff (%)	First pass (10^6)	Second pass (10^6)	Third pass (10^6)	Fourth pass (10^6)
1	32,589	43,191	32.53	33,738	8,626	827	0
2	33,027	47,375	43.44	36,808	10,567	0	0
3	15,360	19,810	28.97	16,873	2,928	9	0
4	35,792	53,875	50.52	35942	10,903	6,203	827
5	39,344	54,897	39.53	39,528	13,552	1,817	0
6	20,539	25,468	24	21,642	3,817	9	0
7	36,829	54,459	47.87	38,760	9,996	5,703	0
8	36,586	48,046	31.32	39,636	6,932	744	734
9	23,027	35,799	55.47	24621	4,936	3,948	2,294
Avg.	30,343	42,742	39.82	31,919	8,226	2,250	1,040

It should also be mentioned that for both approaches, all 36 machines were used to some extent over the 24-hour planning horizon. Table 11 lists the makespan and average machine operation time obtained from the single-pass algorithm and the three-phase scheme, and the percentage difference between them. The column headings are self explanatory. As can be seen, the makespan associated with all instances is essentially 24 hours, which implies that the capacity of at least one of the 36 machines is fully utilized. However, the average machine operation time for the multipass schedule is roughly 7% to 18% greater than that associated with the single-pass schedule, a much more insightful result. The difference is a measure of the increased efficiency that is realized when machine changeovers and the reuse of tooling are part of the analysis.

Table 12 lists the runtime of the single-pass algorithm (phase I), phases II plus III of the three-phase scheme, and the sum of all three phases. After extensive testing, the number of iterations of phase I was set to 500 and the number of iterations of phase II as

well as phase III was set to 100; that is, $n_{\mathrm{I}} = 500$, $n_{\mathrm{II}} = n_{\mathrm{III}} = 100$. In the implementation the GRASP is first run for 500 iterations to get the single-pass solution. Then, starting with the best phase I solution, phases II and III are repeated 100 times.

The statistics in Table 12 indicate that the three-phase scheme never takes more than thirty minutes. Although the number of iterations for phases II and III is only one-fifth of that for phase I, the runtime of phases II and III is nearly 40% more than that of phase I on average. Given this contrast, if it were desirable to reduce the computational effort, it would be best to focus on the second two phases.

Table 11: Comparison of average makespan and machine time

Prob. no.	Single-pass makespan (hr)	Multipass makespan (hr)	Percent diff (%)	Single-pass average mach time (hr)	Multipass average mach time (hr)	Percent diff (%)
1	24	24	0	21.47	23.106	7.62
2	24	24	0	20.42	23.4209	14.70
3	24	24	0	18.8	20.8207	10.75
4	24	24	0	20.55	23.1489	12.65
5	24	24	0	21.07	23.5307	11.68
6	24	24	0	18.99	20.9661	10.41
7	23.97	23.99	0.08	21.52	23.2414	8.00
8	23.98	23.99	0.04	21.82	23.3344	6.94
9	23.98	23.98	0	18.46	21.778	17.97
Avg.	23.99	23.99	0.01	20.34	22.59	11.19

Table 12: Runtime comparison

Prob. no.	Single-pass (s) $n_{\mathrm{I}} = 500$	Phase II and III (s) $n_{\mathrm{II}} = n_{\mathrm{III}} = 100$	Total (s)
1	936	920	1,856
2	950	936	1,886
3	638	940	1,578
4	835	931	1,766
5	824	968	1,792
6	573	932	1,505
7	597	1,017	1,614
8	502	990	1,492
9	275	669	944
Avg.	681	923	1,604

Chapter 7: Multipass Scheduling Scheme II

7.1 MATHEMATICAL MODEL

The model presented in this section is an extension of the single-pass model (1), which only considers first-pass lots. For model (1), the basic GRASP was developed to find solutions because the underlying MIP was too difficult to solve exactly with a commercial code. The multipass nature increases both the dimensionality of problem by taking the *pass* into account and the scale of the problem by considering virtual lots. Incorporating these factors as well as the precedence constraints implied by the pass requirements into the original MIP would have vastly increased the number of variables and constraints, which would have correspondingly increased the computational burden to the point where even small instances would not have been solvable

The full machine setup and scheduling problem cannot be modeled efficiently as a MIP when machine changeovers and lot sequencing considerations are included. As a consequence, we decomposed the problem into two parts. In the first part a variation of an assignment problem is solve which includes the objective function and all the constraints described in Section 3.2, except the sequencing constraints. In other words, the assignment problem determines how to best choose machine-tooling-temperature combinations and how to assign lots to machines, but not how sequence the lots. In the second part, a sequencing problem is solved that orders the lots on their assigned machines. Here, we do not permit machine setups to be modified or unassigned lots to be introduced. To maintain feasibility after an "optimal" sequence is found, it may be necessary to remove lots or insert idle time into the schedule. The objective of the sequencing model is to minimize the weighted sum of lots that have to be discarded from the solution of the assignment model plus the weighted sum of any added idle time. In

the Appendix D, the complexity of both problems is analyzed and each is shown to be $\mathcal{NP}$-hard in the strong sense.

7.1.1 Assignment model

To further clarify terminology, lot pass number p specifies which step of the route the lot is to next undergo. Assume device 1 has five steps in its route and lot 101 containing device 1 is to undergo the third step. Then, the set of passes, P, considered for lot 101 during the planning horizon, is {3, 4, 5} instead of {1, 2, 3}. For convenience, we call pass 3 of lot 101 as first pass of lot 101 although its pass number is really 3 in actuality. In the developments, we make use of the following notation.

Indices and sets

D	set of all devices; $j \in D$
K	set of key devices; $k \in K \subseteq D$
L	set of lots in WIP including initial lots; $l \in L$
M	set of machines (each machine is a member of a machine family); $i \in M$
P	set of all possible passes; $p \in P$
R	set of subroutes (each subroute is a specific machine–tooling–temperature combination); $r \in R$
T	set of tooling families; $t \in T$
TP	set of operating temperatures; $\tau \in TP$
$j(l)$	the device contained in lot l
$L(i)$	set of lots that can be processed on machine i during the upcoming pass or a future pass (virtual lots); $l \in L(i)$, $i \in M$
Λ	set of feasible tooling setups; $\lambda \in \Lambda$

$\Lambda(i)$ set of feasible tooling setups that are compatible with machine i; $\lambda \in \Lambda(i)$, $i \in M$

$\Lambda(i,l)$ set of feasible tooling setups that are compatible with machine i and can process lot l; $i \in M$, $l \in L(i)$

$\Lambda(i,t)$ set of feasible tooling setups that are compatible with machine i and contain tooling family t; $\lambda \in \Lambda(i, t)$, $i \in M$, $t \in T$

$\Lambda(i, t,\tau)$ set of feasible tooling setups that are compatible with machine i, contain tooling family t and run under temperature τ; $\lambda \in \Lambda(i, t,\tau)$, $i \in M$, $t \in T$, $\tau \in TP$

$P(j)$ set of all passes in the route for device j; $p \in P(j)$, $j \in D$

$P(l)$ set of passes considered during the planning horizon for each lot l. For example, assuming lot l is to undergo step 2 of its route and there are four steps in the route, then $P(l) = \{2, 3, 4\}$. Note that if lot l is an initial lot, then $P(l)$ will be {3, 4} with the initial pass ignored; $p \in P(l)$, $l \in L$

$p_0(l)$ upcoming pass for each lot l ; that is, $p_0(l)$ is the first element in the set $P(l)$. For example, assuming that step 2 in the route of the device in lot l is the next step, then pass 2 will be the upcoming pass; $p_0 \in P(l)$, $l \in L$

$p_1(l)$ the last pass for lot l ; that is, $p_1(l)$ is the last element in the set $P(l)$. For example, assuming a lot has a total of four passes, then pass 4 will be the last pass; $p_1 \in P(l)$, $l \in L$

$P(k)$ set of passes for each key device k; $P(k) \subset P$, $k \in K$

$M(j,p)$ set of machines that can process pass p of device j; $i \in M(j,p)$, $j \in D$, $p \in P(j)$

$M(l)$ set of machines that can process at least one pass of lot l; $i \in M(l)$, $l \in L$

$M(l,p)$ set of machines that can process pass p of lot l; $i \in M(l,p)$, $l \in L$, $p \in P(l)$

$P(l,i)$ set of passes considered during the planning horizon for each lot l such that these passes can be processed by machine i; $p \in P(l,i)$, $l \in L$, $i \in M(l)$

$N(t)$ set of temperatures that are compatible with tooling family t; $\tau \in N(t)$, $t \in T$

$L(i,j,p)$ set of lots (including virtual lots) containing device j and undergoing pass p that can be processed by machine i; $l \in L(i,j,p)$, $i \in M$, $j \in D$, $p \in P(j)$

$R(i,l,p)$ set of subroutes that use machine i to process pass p of lot l; $r \in R(i,l,p)$, $i \in M$, $l \in L(i)$, $p \in P(l,i)$

$R(i,l,\lambda,p)$ set of subroutes that use machine i to process pass p of lot l with tooling setup λ; $r \in R(i,l,\lambda,p)$, $i \in M$, $l \in L(i)$, $\lambda \in \Lambda(i,l)$, $p \in P(l,i)$

Parameters and data

$b_{\lambda t}$ number of tooling pieces from family t required by setup λ

C normalizing constant associated with the various key device shortages

H_i (capacity) number of hours available on machine i over the planning horizon; that is, the total machine hours less the amount of time used to process the initial lot if machine i has an initial lot

$n_{\tau t}^{tooling}$ number of tooling pieces from family t available under temperature τ

$n_t^{tooling}$ number of tooling pieces available from family t

$n_l^{devices}$ number of devices (chips) in lot l

$n_{pk}^{min\_key}$ minimum number of devices associated with key device k that are required to be processed over the planning horizon during pass p

ρ_{ilr} processing rate of lot l on machine i using subroute r (devices per hour)

w_{lp} weight (benefit) associated with processing lot l during pass p (function of lot age and the remaining planned cycle time)

ε_k^{short} weight (penalty) associated with shortage of key device k

ε_r penalty for choosing subroute r; for the preferred subroute $\varepsilon_r = 0$.

ε_M penalty on the number of machines used

ε_T penalty on the makespan

ST_i number of hours required to finish the initial lot on machine i if it has an initial lot, 0 otherwise

τ_l^{load} load and unload time for each lot l

τ_λ^{setup} setup time for each setup λ

Decision variables

x_{ilr}^p 1 if pass p of lot l is processed by machine i using subroute r, 0 otherwise; $i \in M$, $l \in L(i)$, $p \in P(l,i)$, $r \in R(i,l,p)$

$y_{i\lambda}$ 1 if machine i uses setup λ, 0 otherwise; $i \in M$, $\lambda \in \Lambda(i)$

Δ_{pk}^{short} shortage of key device k at pass p; $k \in K$, $p \in P(k)$

t^{max} latest completion time among all machines processing lots (makespan)

$t_{i\lambda}$ total time used by machine i with setup λ to process lots; $i \in M$, $\lambda \in \Lambda(i)$

t_i total time used by machine i to process lots; $i \in M$

Model

$$\text{Minimize} \quad \frac{1}{C}\sum_{k\in K}\sum_{p\in P(k)} \varepsilon_k^{short}\Delta_{pk}^{short} - \sum_{i\in M}\sum_{l\in L(i)}\sum_{p\in P(l,i)}\sum_{r\in R(i,l,p)}\left(w_{lp}-\varepsilon_r\right)x_{ilr}^p + \varepsilon_M\sum_{i\in M}\sum_{\lambda\in\Lambda(i)} y_{i\lambda} + \varepsilon_T t^{max} \tag{3a}$$

$$\text{subject to} \quad \sum_{i\in M(l,p)}\sum_{r\in R(i,l,p)} x_{ilr}^p \le 1, \quad \forall\ l \in L,\ p \in P(l) \tag{3b}$$

$$\sum_{\lambda\in\Lambda(i)} y_{i\lambda} \le 1, \quad \forall\ i \in M \tag{3c}$$

$$\sum_{p\in P(l,i)}\sum_{r\in R(i,l,p)} x_{ilr}^p \le \left|P(l,i)\right| \sum_{\lambda\in\Lambda(i)} y_{i\lambda}, \ \forall\ i \in M,\ l \in L(i) \tag{3d}$$

$$\sum_{i\in M}\sum_{\tau\in N(t)}\sum_{\lambda\in\Lambda(i,t,\tau)} b_{\lambda t} y_{i\lambda} \le n_t^{tooling}, \quad \forall\ t \in T \tag{3e}$$

$$t_i = \sum_{l\in L(i)}\sum_{p\in P(l,i)}\sum_{r\in R(i,l,p)}\left(\frac{n_l^{devices}}{\rho_{ilr}} + \tau_l^{load}\right)x_{ilr}^p + \sum_{\lambda\in\Lambda(i)}\tau_\lambda^{setup} y_{i\lambda}, \ \forall\ i \in M \tag{3f}$$

$$t_i \le H_i \sum_{\lambda \in \Lambda(i)} y_{i\lambda}, \quad \forall\, i \in M \tag{3g}$$

$$t^{max} \ge t_i, \quad \forall\, i \in M \tag{3h}$$

$$\sum_{i \in M(k,p)} \sum_{l \in L(i,k,p)} \sum_{r \in R(i,l,p)} n_l^{devices} x_{ilr}^{p} + \Delta_{pk}^{short} \ge n_{pk}^{min\_key}, \quad \forall\, k \in K, p \in P(k) \tag{3i}$$

$$\sum_{i \in M(l,p)} \sum_{r \in R(i,l,p)} x_{ilr}^{p} \ge \sum_{i \in M(l,p+1)} \sum_{r \in R(i,l,p+1)} x_{ilr}^{p+1}, \quad \forall\, l \in L, p \in P(l): p < p_1(l) \tag{3j}$$

$$\sum_{i \in M(l,p_0(l))} ST_i \left(\sum_{r \in R(i,l,p_0(l))} x_{ilr}^{p_0(l)} \right) + \sum_{\substack{p' \in P(l) \\ p' \le p}} \sum_{i \in M(l,p')} \sum_{r \in R(i,l,p')} x_{ilr}^{p'} \left(\frac{n_l^{devices}}{\rho_{ilr}} + \tau_l^{load} \right)$$

$$\le \sum_{i \in M(l,p)} (H_i + ST_i) \left(\sum_{r \in R(i,l,p)} x_{ilr}^{p} \right), \quad \forall\, l \in L, p \in P(l): p > p_0(l) \tag{3k}$$

$$x_{ilr}^{p} \in \{0,1\}, \forall\, i \in M, l \in L(i), p \in P(l), r \in R(i,l,p),$$

$$y_{i\lambda} \in \{0,1\},\ t_{i\lambda} \ge 0,\ \forall\, i \in M, \lambda \in \Lambda(i),$$

$$\Delta_{pk}^{short} \ge 0, \text{ integer}, \ \forall\, k \in K, \quad t^{max} \ge 0 \tag{3l}$$

There are four terms in the objective function (3a) given in order of importance. The first term is the weighted sum of key device shortage. In fact, these shortages are reduced only when the last pass of a lot containing the device is completed. To account for this in the model, we created shortage variables, Δ_{pk}^{short}, for each pass p of key device k, and set the production target $n_{pk}^{min\_key}$ of each pass to be the same as the target of the final product except if the pass p of device k is an initial lot. The target production for each pass of a device needs to be reduced by the amount contained in initial lots. If desired, the shortage penalty ε_k^{short} associated with key device k can be modified to be a function of the different passes. The second term is the weighted sum of lots processed, the third is number of machine used, and the fourth is the makespan. The magnitude of the penalties associated with the four terms reflects their relative importance. In Section 7.3.1, we explain how these penalties are calculated.

Constraints (3b) – (3e) account for resource availability and route selection. Constraints (3b) limit the number of machines and subroutes chosen for a lot at any pass

to at most one while (3c) limit the setups on a machine to at most one. Constraints (3d) indicate the relationship between lot l and machine i. First, if a machine is not set up with tooling, then no lot can be assigned to it; if machine i has been set up and lot l assigned to it, then the number of passes of lot l cannot be larger than $|P(l,i)|$, the number of passes of lot l that can be performed on machine i with its designated setup. Constraints (3e) ensure that the number of tooling pieces used from tooling family t does not exceed the number available.

Constraints (3f) – (3h) are associated with operation times. Constraints (3f) calculate the total time used by a machine for setup, loading and unloading, and lot processing, but not the time for processing initial lots. A single value, τ_l^{load}, is used for unloading lot l on its current machine and then loading the next lot onto that machine. At time zero, if a machine is not processing a lot, we assume the loading time of the first lot is zero. Constraints (3g) show that the operations of any machine i cannot extend beyond its available time, H_i. The makespan is computed by constraints (3h). Recall that the minimization of t^{max} is included in the objective function, so constraints (3h) are equivalent to $t^{max} = \max\{t_i, \forall\, i \in M\}$.

Combined with the first term in the objective function (3a), constraints (3i) calculate the quantity of key device shortages. Because Δ_{pk}^{short} is nonnegative, even if production of key device k is over its target $n_{pk}^{min\_key}$ at pass p, Δ_{pk}^{short} will still be 0. Thus, when the target key device k is reached, there is no incentive in the model to give priority to lots with k so the selection of those lots will only depend on their relative weights, w_{lp}.

Constraints (3j) enforce the precedence relations between each consecutive pair of passes. They require that if pass $p + 1$ of lot l is assigned to a machine, then pass p must also be assigned. For example, assume that the next pass for lot 103 is pass 2 and $P(103) = \{2, 3, 4\}$. If pass 2 of lot 103 is not assigned, then pass 3 and 4 cannot be assigned

either. Note that (3j) say nothing about the order in which the passes are executed so the solution to model (3) may not be feasible.

Constraints (3k) account for the total time associated with each pass p of lot l beyond the first pass $p_0(l)$. The essential difference between these constraints and (3f) – (3h) lies in the fact that the former are derived from the sequence of passes of a lot while the latter are derived from the lot queuing on a machine. Constraints (3k) ensure that the total processing time accumulated by lot l from pass $p_0(l)$ until pass p must be not greater than the hours available on the machine that processes pass p of lot l. Note that the starting point of the planning horizon is regarded as hour 0. The first term on the left-hand side (LHS) takes into account the starting time of pass $p_0(l)$ if it is performed on a machine that has an initial lot. The second term sums the processing, load and unload times of the passes from $p_0(l)$ up to p. To better understand (3k), suppose we are considering lot 103 such that $p_0(103)$ is pass 2 and $p_1(103)$ is pass 4, and that the processing time plus load and unload time for passes 2, 3, and 4 are 9, 10 and 5 hours, respectively. Then completing pass 4 requires at least 24 hours, that is, the sum of processing hours plus load plus unload hours of passes 2, 3 and 4. Now assume that the amount of time available on machine 1 is 24 hours and on machine 2, 23 hours. Thus pass 4 of lot 103 can be assigned to machine 1 but not machine 2. In fact, assigning pass 4 to machine 1 cannot be guaranteed to be feasible since the time when passes 2 and 3 are finished is unknown. The sequencing model addresses this issue. Finally, variable definitions are given in (3*l*).

Initial conditions. For model (3), an option exists that allows machines processing a lot at time zero to be reset when those lots are finished. The default is to maintain the initial setup. If the option is *not* selected, then we set $y_{i\lambda} = 1$ for all such machines i, where λ is determined from the input file " initialsetup.csv." Moreover, the last term on the RHS of

constraints (3f) that sums the tooling setup time τ_{λ}^{setup} is removed for the corresponding machines.

7.1.2 Sequencing model

Given the machine setups and lot assignments provided by the solution of model (3), we wish to sequence the lots on their assigned machines to ensure feasibility while maximizing the weighted sum of lots processed. At this stage, we do not permit machines to be reset nor do we allow lots to be switched from one machine to another or unassigned lots to be introduced into the production plan. Even with these restrictions, the problem is not straightforward because of the need to take into account start times and precedence relations between passes of the same lot. To assure feasibility, it will often be necessary to remove some lots from the production plan or to insert idle time between some lots.

Since both real and virtual lots may be in the assignment model solution, two lots may differ only by their pass number *p*. To distinguish these cases, we define a lot-pass combination that consists of the lot name *l* and a pass number *p*, collectively indexed by *g*. A *lot name* is an alphanumeric string. For modeling purposes, a single dummy lot-pass combination, g^{dum}, is created for the index of the predecessor of the first lot and the successor of the last lot on any machine. In addition to the notation introduced in Section 7.1.1, we also make use of the following

Indices and sets

$\overline{G}$ set of lot-pass combinations in the solution to the assignment model; $g \in \overline{G}$

$\overline{L}$ set of lots in the solution to the assignment model (each lot is identified by its name); $l \in \overline{L}$

$\bar{P}(l)$ set of passes associated with lot name l in the solution to the assignment model (not including the initial pass of initial lots); $p \in \bar{P}(l)$, $l \in \bar{L}$

$g(l,p)$ index of lot-pass combination (l,p)

g^{dum} dummy lot-pass combination with lot name $l = -1$ and pass number $p = -1$

l_g lot name associated with the lot-pass combination g (different g could contain the same lot)

p_g pass number associated with the lot-pass combination g

$\bar{M}$ set of machines used in the solution to the assignment model; $i \in \bar{M}$

$\bar{G}(i)$ set of lot-pass combinations assigned to machine i in the solution to the assignment model; $i \in \bar{M}$, $g \in \bar{G}(i)$

$\bar{L}(i)$ set of lot identified by their name assigned to machine i in the solution to the assignment model; $l \in \bar{L}(i)$, $i \in \bar{M}$

Parameters and data

H maximum machine capacity; that is, $H = \max_{i \in \bar{M}}\{H_i\}$

$\tau_g^{process}$ processing time for each lot associated with index g derived from the solution to the assignment model; if $x_{il_g r}^{p_g} = 1$, then $\tau_{l_g}^{process} = n_{l_g}^{devices} \Big/ \rho_{il_g r}$; $g \in \bar{G}$

τ_i^{setup} setup time for machine i when it is in the assignment model solution; if $y_{i\lambda} = 1$, then $\tau_i^{setup} = \tau_\lambda^{setup}$, $i \in \bar{M}$

ε^{delay} penalty for a 1-hour delay; $\varepsilon^{delay} = 1/(100H)$

ω_g^{seq} weight for the lot-pass combination p in the sequencing model; if the lot associated with g contains key device k, then $\omega_g^{seq} = \varepsilon_k^{short} + w_{lp}$, else $\omega_g^{seq} = w_{lp}$.

Decision variables

u_g 1 if lot-pass combination g is contained in the solution of the sequencing model, 0 otherwise; $g \in \overline{G}$

Δ_g amount of idle time (hours) inserted right before the lot associated with g is to begin processing; $g \in \overline{G}$

st_g starting time of lot associated with g; $g \in \overline{G}$

ct_g completion time of lot associated with g; $g \in \overline{G}$

$z_{i,g,g'}$ 1 if lot-pass combination g is processed right before lot-pass combination g' on machine i (i.e., lot associated with g' is the immediate successor of lot associated with g on machine i), 0 otherwise; $i \in \overline{M}$, $g \in \overline{G} \cup \{g^{dum}\}$, $g' \in \overline{G} \cup \{g^{dum}\}$, $g \neq g'$

Model

Maximize $$\sum_{g \in \overline{G}} \omega_g^{seq} u_g - \varepsilon^{delay} \sum_{g \in \overline{G}} \Delta_g \tag{4a}$$

subject to

$$(ST_i + \tau_i^{setup}) u_g + \Delta_g \leq st_g, \quad \forall\, i \in \overline{M},\ g \in \overline{G}(i) \tag{4b}$$

$$st_g \leq (ST_i + \tau_i^{setup}) u_g + \Delta_g + H_i (1 - z_{i,g^{dum},g}), \quad \forall\, i \in \overline{M},\ g \in \overline{G}(i) \tag{4c}$$

$$ct_g = st_g + (\tau_g^{process} + \tau_{l_g}^{load}) u_g, \quad \forall\, g \in \overline{G} \tag{4d}$$

$$ct_g \leq H_i + ST_i, \quad \forall\, i \in \overline{M},\ g \in \overline{G}(i) \tag{4e}$$

$$\sum_{\substack{g' \in \overline{G}(i) \cup \{g^{dummy}\} \\ g' \neq g}} z_{i,g,g'} = 1, \quad \forall\, i \in \overline{M},\ g \in \overline{G}(i) \cup \{g^{dum}\} \tag{4f}$$

$$\sum_{\substack{g' \in \overline{G}(i) \cup \{g^{dummy}\} \\ g' \neq g}} z_{i,g',g} = 1, \quad \forall\, i \in \overline{M},\ g \in \overline{G}(i) \cup \{g^{dum}\} \tag{4g}$$

$$ct_g + \Delta_{g'} \leq st_{g'} + H_i (1 - z_{i,g,g'}), \quad \forall\, i \in \overline{M},\ g \in \overline{G}(i),\ g' \in \overline{G}(i),\ g' \neq g \tag{4h}$$

$$ct_{g(l,p)} \leq st_{g(l,p')} + H(2 - u_{g(l,p)} - u_{g(l,p')}), \quad \forall\, l \in \overline{L},$$

$$p \in \overline{P}(l),\ p' \in \overline{P}(l),\ p' = p+1 \tag{4i}$$

$$u_{g(l,p')} \le u_{g(l,p)}, \quad \forall\, l \in \bar{L},\, p \in \bar{P}(l),\, p' \in \bar{P}(l),\, p' = p+1 \tag{4j}$$

$$st_g - ct_{g'} - H_i(1 - z_{i,g',g}) \le \Delta_g, \quad \forall\, i \in \bar{M},\, g \in \bar{G}(i),\, g' \in \bar{J}(i),\, g' \ne g \tag{4k}$$

$$\Delta_g \le \left(H_i - \tau_g^{process} - \tau_i^{setup} - \tau_{l_g}^{load}\right) u_g, \quad \forall\, i \in \bar{M},\, g \in \bar{G}(i), \tag{4l}$$

$$u_g \in \{0,1\},\ \Delta_g \ge 0,\ st_g \ge 0,\ ct_g \ge 0, \quad \forall\, g \in \bar{G}$$

$$z_{i,g,g'} \in \{0,1\},\ \forall\, i \in \bar{M},\, g \in \bar{G} \cup \{g^{dum}\},\, g' \in \bar{G} \cup \{g^{dum}\},\, g' \ne g \tag{4m}$$

The primary goal of model (4) is to process as many of the lots contained in the solution to model (3) as possible taking their relative importance into account. When necessary, idle time can be inserted right before a lot starts to be processed to ensure that the full schedule is feasible. The secondary goal is to minimize the total number of idle hours that are inserted, which is equivalent to minimizing the completion time on each machine. The objective function (4a) contains two terms corresponding these goals. The priority order is enforced by setting the penalty parameter ε^{delay} to a small positive number.

Constraints (4b) provide a lower bound on the time that machine i can start processing each lot assigned to it as indicated by the index $g \equiv g(l,p)$. The bound is the sum of starting time of the machine, its setup time, and the number of idle hours that are inserted right before the lot. Constraints (4c) provide an upper bound on the starting time of the first lot processed on machine i. If the lot associated g is the first lot processed on machine i, then it will be the immediate successor of the dummy lot g^{dum}, with $z_{i,g^{dum},g}$ = 1. Taking (4b) and (4c) together, we see that st_g is equal to the lower bound of the starting time for the lot associated with index g that gives $z_{i,g^{dum},g}$ = 1 on machine i. These constraints force the first lot on each machine i to start at $(ST_i + \tau_i^{setup})u_g + \Delta_g$, which is as early as possible.

Constraints (4d) indicate that the completion time of a lot is equal to the staring time plus the processing time (including loading and unloading time) of the lot. If the lot

associated with g is removed from the sequence, i.e., $u_g = 0$, then constraints (4d) imply that the starting time is just equal to the completion time for processing that lot so the capacity of machine i is unaffected. Constraints (4e) impose an upper bound on the completion time for processing the lot associated with g. The bound is the starting time plus the capacity of the machine to which the lot was assigned in the solution of model (3).

Constraints (4f) – (4j) are associated with lot sequencing. Constraints (4f) and (4g) respectively require that each lot assigned to machine i must have exactly one successor and one predecessor, which could be the dummy lot. That is, the last lot in the sequence will be followed by the dummy lot while the first lot will have the dummy lot as its predecessor. Constraints (4h) place a bound on the start time of the lot associated with g' equal to the completion time of its immediate predecessor, ct_g, plus the amount of idle time inserted right before it. For example, assume that the lots associated with g_1 and g_2 are assigned to machine 1 in the solution to the model (3). If g_1 is the immediate predecessor of g_2, then $z_{1,g_1,g_2} = 1$ and (4h) becomes $ct_{g_1} + \Delta_{g_2} \le st_{g_2}$. If g_1 is not the immediate predecessor of g_2, then (4h) is redundant since adding H_1 to st_{g_2} makes the RHS at least as large as the LHS.

Constraints (4i) ensure that the completion time of a lot cannot be greater than the starting time of its next pass. As can be seen, only if both $u_{g(l,p)}$ and $u_{g(l,p')}$ are equal to1 for $p' = p + 1$, will $ct_{g(l,p)} \le st_{g(l,p')}$ be enforced. Constraints (4j) enforce the precedence relations between passes; if any pass of lot has to be removed then its subsequent passes must also be removed.

Constraints (4k) and (4*l*) place bounds on the amount of idle time that can be inserted before a lot. Constraints (4k) limit the length of the hours before the lot associated with g to no more than the lot's starting time st_g minus the completion time of

the lot's immediate predecessor denoted by $ct_{g'}$. When $z_{i,g',g} = 1$, constraints (4k) in conjunction with the second term in the objective function (4a) ensure that $\Delta_g = st_g - ct_{g'}$. When $z_{i,g',g} = 0$, then the LHS of (4k) is sufficiently small to make the constraint redundant. Constraints (4l) provide upper bounds on the number of idle hours that can be inserted before a lot. When $u_g = 1$, the term in parentheses on the RHS represents the number of hours available on machine i minus the number required to finish processing lot l_g. Logically, any idle time inserted into the schedule must not exceed this value. If the lot associated with g is removed, then $u_g = 0$ forcing the idle time to be 0. Finally, all the variables in the model are defined in (4m).

7.2 SOLUTION METHODOLOGY

A three-phase methodology referred to as ASC (assignment, sequence and changeover) is used to solve the full AT scheduling problem. In the first phase, the assignment model (3a)–(3l) is solved to get an optimal machine-tooling configuration and lot assignments. The resulting production plan is used in the second phase where model (4a)–(4m) is sequence the assigned lots on their corresponding machines. In the third phase, a greedy randomized procedure (but not a full GRASP) is used to reconfigure machines to exploit their full capacity. A sample problem is provided throughout Section 7.2 to illustrate the output after each phase. The sample problem has four machines indexed from 1 to 4, and nine lots similarly indexed and divided into four sets: $L_1 = \{1,4,5,6\}$, $L_2 = \{2,3,8\}$, $L_3 = \{7\}$, $L_4 = \{9\}$. The lots in L_1 are to undergo passes 1 and 2, those in L_2 are to undergo passes 1 – 3, lot 7 in L_3 is to undergo pass 2, and lot 9 in L_4 is in process at time zero on machine 1.

7.2.1 Phase I: assignment model

Model (3) proved extremely difficult to solve to optimality within several hours with CPLEX so a number of tightening variables and constraints were added to the formulation. The augmented model led to much reduced runtimes and improved results. The following notation is used in the presentation here.

Indices and sets

F set of machine families; $f \in F$

$M(f)$ set of machine instances that are members machine family f; $i_1, i_2, \ldots, i_{n_f} \in M(f)$, $f \in F$

Decision Variables

$\overline{y}_i$ 1 if machine i is set up with some tooling, 0 otherwise; $i \in M$

Constraints

$$\sum_{\lambda \in \Lambda(i)} y_{i\lambda} = \overline{y}_i, \qquad \forall\ i \in M \tag{3m}$$

$$\overline{y}_{i_q} \le \overline{y}_{i_{q+1}}, \qquad \forall\ f \in F,\ \ q \in \{1, 2, 3 \ldots n_f - 1\} \tag{3n}$$

$$\overline{y}_i \in \{0, 1\}, \qquad \forall\ i \in M \tag{3o}$$

A machine is counted as "selected" if it is set up with tooling and assigned lots. Constraints (3m) are a stronger version of (3c) and have proven effective in finding feasible solutions during branch and bound (e.g., see Jarrah et al. 1994). Setting $\overline{y}_i = 0$ will result in all variables of $y_{i\lambda}$ equal to zero, which improves the efficiency of branching scheme. For the machine instances in the same machine family, the order in which they are selected is controlled by the symmetry breaking constraints (3n). The machine instance with larger index is selected first. Without constraints (3n), the optimization code will randomly choose a machine instance from a machine family, which exponentially increases the number of configurations that would have to be explored. For instance, assume machines 1, 2, and 3 belong to the machine family. Also assume that

in the derived solution only two machine instances in this family are used. With constraints (3n), this will be just machines 2 and 3 rather than all three combinations of the three machines taken in pairs.

The solution of model (3a) – (3o) for the sample problem is shown in Figure 7 where it is assumed that the index number of the optimal tooling setup is just the same as the machine to which it is assigned, and that only one temperature is feasible. The notation (l,p) refers to the lot-pass combination (l,p) with lot name l and pass number p. The bars in the figure indicate the lot-pass combinations that were assigned to each machine in the solution. Those not assigned are listed below the graph. At time zero, for example, machine 1 is processing pass 1 of lot 9, and is assigned the lot-pass combinations (1,1), (2,1), (2,2), (4,2), (9,2) for the remainder of the planning horizon. The solution presented, of course, only represents lot assignments so the lot sequences are arbitrary. There are many equivalent solutions not all of which are feasible. In fact, it can be seen that pass 4 of lot 9 starts on machine 2 before pass 2 starts on machine 1. The arrangement of lots only shows the number of hours consumed on the respective machines. Finally, the lot-pass combinations (2,3), (7,2), (3,3) and (8,3) were not assigned to any machine, either because they are not compatible with any of the four setups or the machine capacity was insufficient to work them in.

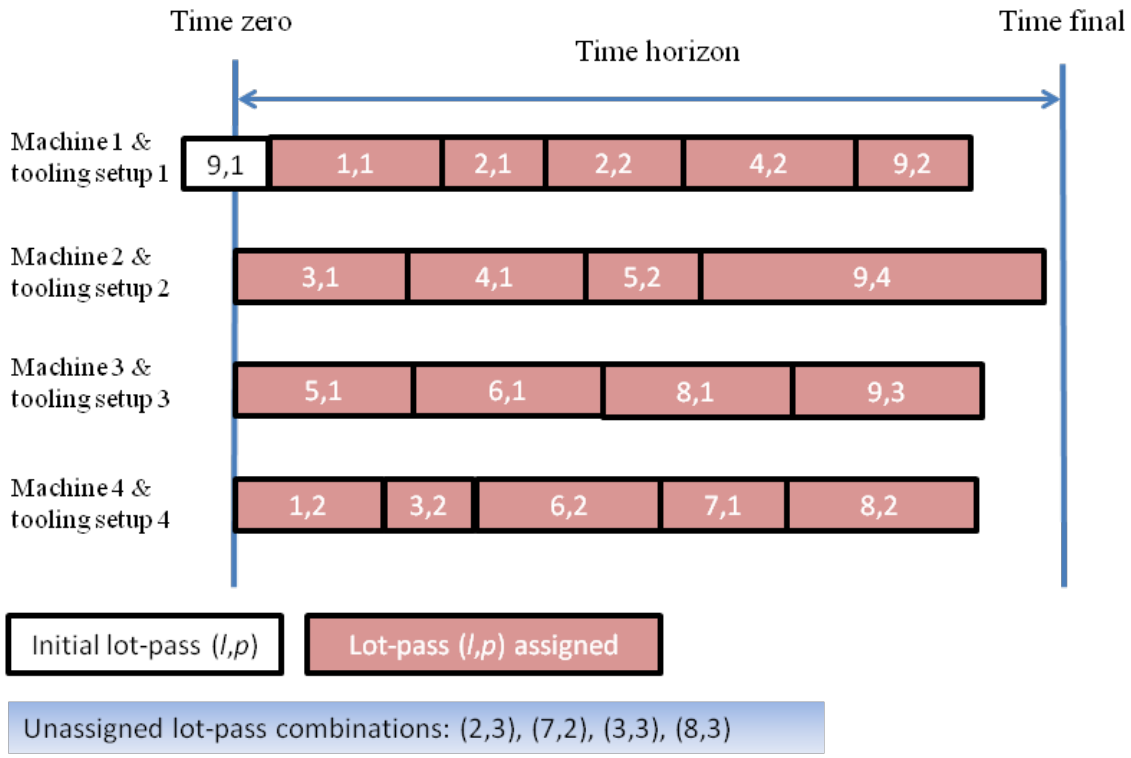

Figure 7: Sample results of Phase I

7.2.2 Phase II: sequencing model

Model (4) is much smaller than the assignment model in terms of number of variables and constraints as shown in the next section. Although a proof that the sequencing problem is strongly NP-hard is provided in the Appendix D, all the instances we investigated were easy to solve with CPLEX.

The solution to the sequencing model for the sample problem is given in Figure 8. All precedence requirements are seen to be satisfied between passes even if a lot are processed on different machines. For example, the starting time of lot-pass (1,2) on machine 4 was earlier than the completion time of lot-pass (1,1) on machine 1 in Figure 1, but now its starting time is feasible with respect to the new sequence. The same applies to (9,2) and (9,3). However, on machine 2, it was necessary to remove lot-pass (9,4) to

achieve feasible sequences on machines 1 and 3. If pass 3 of lot 9 was placed in a earlier position on machine 3, say, before lot-pass (6,1) or (8,1), then it would have been necessary to remove either lot-pass (6,2) or (8,2) instead of (9,4). Doing so would have been suboptimal with respect to their contribution to the objective function value (4a). A final point to make about the solution was the need to insert idle time before lot-pass (3,2) on machine 4. Starting it any earlier would have resulted in a conflict with the completion of lot-pass (3,1) on machine 2.

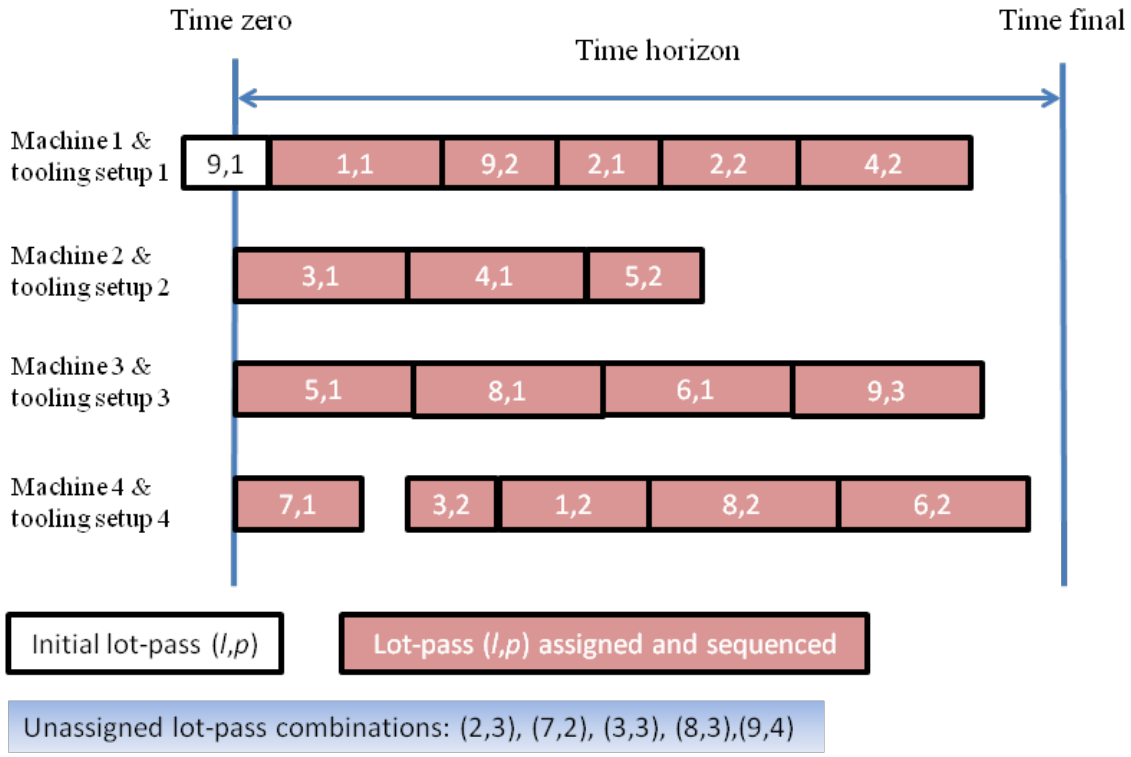

Figure 8: Sample results of Phase II

7.2.3 Phase III: changeover algorithm

Within the planning horizon, the machine-tooling combinations derived in Phase I cannot process more lots than those contained in the solution to model (3). However, it is often possible to exploit the unused capacity of a particular machine by reconfiguring it

with different tooling after it completes its assigned lots. This is the purpose of Phase III which is called after the sequencing problem is solved. During the computations, changeovers are considered for all machines when they become idle regardless of their initial condition at time zero.

The changeover algorithm is adapted from the procedures described by Section 6.2.3. After the production plan is determined by the sequencing model, the time when each machine will finish its assigned lots is determined. The calculations tell us when the machine becomes *empty* and its tooling becomes *available*. All unprocessed real lots are also denoted as *available* and all virtual lots corresponding to the next pass of those lots that were just completed are similarly termed *available*. Of course, there might be other empty machines at the moment a particular machine finishes its assigned lots. This may be due to the fact that they were either empty at time zero or that they could not be reconfigured before the current moment due to lack of tooling or suitable lots. Regardless, check which machines are empty, and which tooling and lots are available at the moment a machine finishes all its assigned lots. Note that if a lot was previously assigned to some machine but processing had not yet started at the current moment, that lot would still be viewed as unavailable.

Now, each time a machine becomes empty we apply a greedy randomized procedure to decide whether any of the empty machines can be reset, which tooling is best, which lots to assign to which machines, and how to sequence the lots. The flowchart for the changeover algorithm is depicted in Figure 6, where $t_c(i)$ indicates completion time of the last lot assgined to machine i and $L^3(i)$ is the set of lot-pass combinations assigned to machine i during this phase of the computations (previously assigned lots are not included). A detailed explanation of each component in the flowchart can be found in the Appendix C.

When the changeover algorithm is applied to the sample problem, we get the results presented in Figure 9. Machine 2 was the first machine that finished all its assigned lots. When it became idle, it was reset with different tooling, which took a small amount of time as indicated by the blank area between lot-pass (5,2) and lot-pass (7,2). This allowed us to assign lot-pass combinations (7,2) and (2,3) to machine 2. Machine 1 was the next machine to finish its originally assigned lots but could not be reset due to either lack of compatible tooling and lots, or insufficient capacity to finish any of the lots in WIP. The same was true for machines 3 and 4, and for machine 2 after finishing lot-pass (2,3). Consequently, we were only able to exploit a portion of the idle capacity of machine 2 in this phase of the computations.

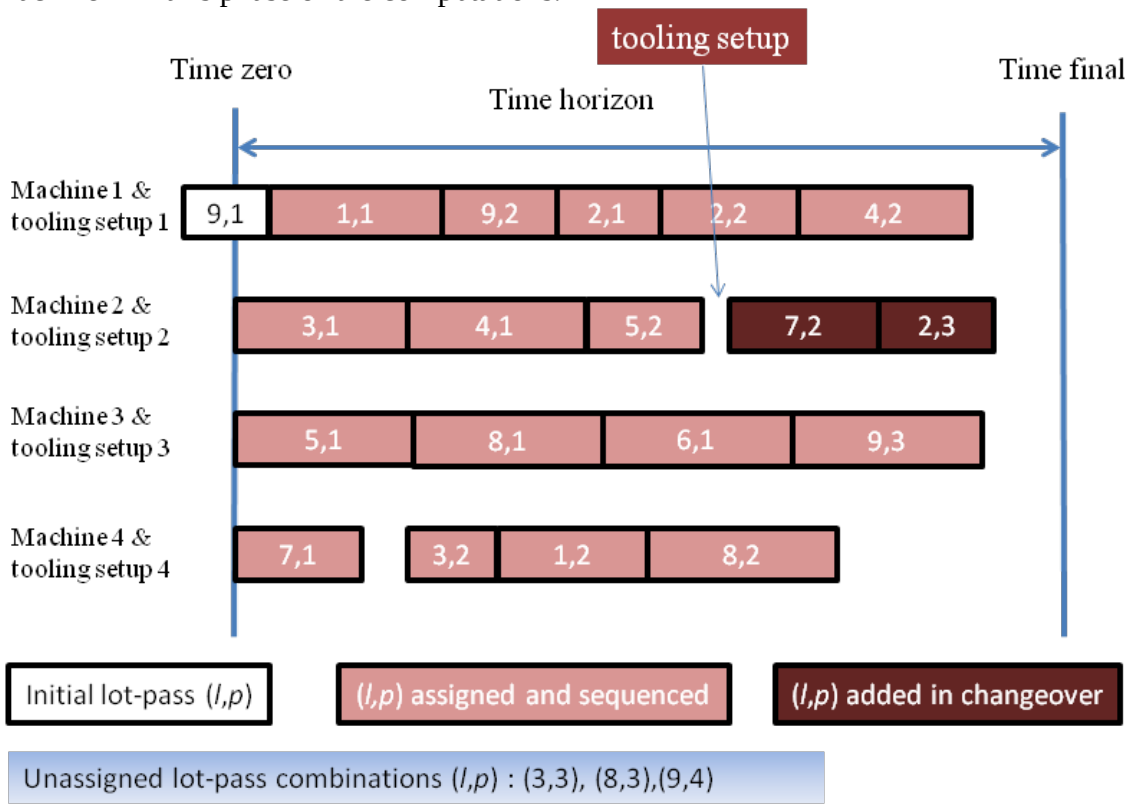

Figure 9: Sample results of Phase III

7.3 COMPUTATIONAL RESULTS

To demonstrate the performance of the three-phase methodology, we conducted a series of numerical tests and compared the results with those obtained with Scheme I. The latter contains a GRASP, multipass heuristic, and a changeover procedure. For conciseness, we refer to it as the GRASP in Section 7.3. For both methodologies, the same data were used. Recall that in the first phase of ASC, machines are configured with tooling and lots are assigned to machines. In the second phase, the assigned lots are sequenced but the machine-tooling setups remain static. Some lots assigned in the first phase may be removed or delayed in the second phase to satisfy the precedence requirement between passes. In the GRASP, feasibility is maintained throughout so no lots are removed. In the third phase, machines may be reset if doing so allows additional lots to be processed. The GRASP also accommodates changeovers.

Besides the comparisons between the GRASP and the ASC, we are also interested in evaluating the degree of infeasibility of the assignment model. Feasible solutions are guaranteed only after the sequencing model is solved. In the computations, if a machine is running a lot at time zero, we maintain its current tooling setup for the first two phases; only in the third phase are changeovers permitted. The same rule applies to the GRASP.

Both methodologies were implemented in C++ and run in the High Performance Computing laboratory of the Mechanical Engineering Department at University of Texas. The lab has a cluster of twelve Dell Poweredge 2950 workstations, each with 2 dualcore, hyperthreading 3.73 GHz Xeon processors and 24 GB of shared memory, and each running Red Hat Linux. CPLEX 12.4 was used to solve all mixed integer programs.

Both real and randomly generated data were used in the experiments. The real data were provided by the AT facility of the collaborating company and consisted of a typical instance, which served as the basis for randomly generating eight additional

instances. In all cases, each instance contains 36 machines, 284 tooling pieces from 6 families, and 1 temperature (certification) setting. The number of lots varied from a low of 800 to a high of 1045 [for further explanation, see Section 6.4]. The input data, itself, is contained in eleven csv files, the most important being "machines.csv," "tooling.csv," "route.csv" and "wip.csv." A full description of each can be found in Deng et al. (2010).

The computational results are presented in the next two subsections in a series of tables. The entries in the first column indicate the problem number and those in the last

7.3.1 GRASP vs. ASC

Table 13 compares the final objective function values of the GRASP and the ASC model. In the case of the latter, a real time upper limit of 2400 seconds was placed on the assignment model and 1200 seconds on the sequencing model. The changeover component was run once for each problem and took about 15 seconds. In the case of the former, no runtime limit was set but an upper limit of 200 iterations was imposed on the computations. This value was determined after extensive testing and reflects a tradeoff between runtime and solution quality. The changeover procedure was run after every iteration. The last column of the table records the percentage difference between the two approaches and was calculated as follows:

$$Percent\ \Delta = 100 \times (GRASP\ value - ASC\ value) / GRASP\ value$$

Table 13: Comparison of GRASP with ASC results

Prob. no.	GRASP objective value (10^{10})	ASC objective value (10^{10})	Percent Δ (%)
1	3.83	0.85	77.88
2	9.01	2.64	70.71
3	1.66	1.04	37
4	9.50	2.33	75.46
5	6.92	1.92	72.23
6	14.1	8.58	39.04
7	8.07	2.42	70.02
8	21.8	5.15	76.38
9	14.0	8.37	40.4
Avg.	9.88	3.70	62.12

As seen in Table 13, the average objective function value obtained by solving the ASC model is 62.12% less than the average provided by the GRASP. Recall that key device shortages dominate the computations so the large values on the order of 10^{10} reflect the weights used to enforce the preemptive nature of the objective function (below, each term is analyzed separately). For all nine problems, the three-phase methodology showed significant improvement. Taking problem no. 1, for example, the objective function values for the GRASP and ASC are 3.83×10^{10} and 8.47×10^{9}, respectively; an improvement of 77.88%.

Tables 14 – 17 provide respective comparisons for each of the four weighted objective function terms: sum of key device shortages, sum of lots processed, number of machines used, and machine time or makespan. The weights associated with these terms decrease by several orders of magnitude from one to the other. For the lots, their weights, w_{lp}, are contained in the input file "wip.csv." The remaining weights and coefficients are calculated as follows.

- Normalizing constant in key device shortage term: C = sum of weight of all regular lots / (10×max{weight of single device over all regular lots})

- Shortage of key device k: ε_k^{short} = sum of weights of all regular lots + sum of weights of regular lots containing device k
- Number of machine used: ε_M = minimum positive lot weight
- Machine time: ε_T = minimum positive lot weight / maximum time horizon

Table 14 indicates that the ASC methodology provides an average reduction of 61.77% in key device shortages compared with the GRASP solution. The average percent difference in Table 14 is close to that in Table 13 due to the dominance of this term. For problem no. 1, for example, the GRASP and ASC values in Table 13 are 3.83×10^{10} and 0.85×10^{10}, respectively, while the corresponding values in Table 14 are 3.89×10^{10} and 0.901×10^{10}. The slight increase is due to the second term (lots processed) whose objective is really one of maximization.

Table 15 presents the comparisons for the weighted sum of lots processed. The average percent difference is 5.86%, much smaller than the difference obtained for the overall objective function value. In fact, in some cases the ASC methodology processed fewer lots than the GRASP. This could have been anticipated by the preemptive weights in the objective function, which allow either approach to give priority to key devices at the expense of regular lots. For problem no. 2, for example, the three-phase methodology processed 0.71% fewer lots than the GRASP, but by comparison was able to reduce the key device shortages by 70.71%. Similar results were observed for four of the nine problem instances. However, the percent differences in Table 15 vary widely so no general conclusions can be drawn.

Table 14: Comparison of weighted sum of key device shortages

Prob. no.	GRASP weighted sum of key device shortage (10^{10})	ASC weighted sum of key device shortage (10^{10})	Percent Δ (%)
1	3.89	0.901	76.86
2	9.07	2.70	70.26
3	1.69	1.06	37.07
4	9.57	2.39	75.06
5	6.98	1.98	71.68
6	14.1	8.60	39.05
7	8.12	2.48	69.45
8	21.9	5.20	76.22
9	14.1	8.40	40.31
Avg.	9.93	3.75	61.77

Table 15: Comparison of weighted sum of lots processed

Prob. no.	GRASP weighted sum of lots (10^{8})	ASC weighted sum of lots (10^{8})	Percent Δ (%)
1	5.42	5.32	1.86
2	6.03	6.07	-0.71
3	2.41	1.80	25.09
4	6.05	5.34	11.71
5	5.28	5.40	-2.27
6	2.55	1.61	36.9
7	5.39	6.31	-17.04
8	4.91	4.76	2.99
9	3.20	3.38	-5.76
Avg.	4.58	4.44	5.86

Tables 16 and 17 highlight the comparisons for the number of machines used in the solutions and the average time those machines were running during the planning horizon, respectively. More specifically, Table 16 refers to the number of machines that are set up with tooling and are assigned lots. As seen, the number of machines used by the ASC methodology is 3.7% less on average than that of the GRASP. In Table 17,

machine time denotes the makespan or how long a machine runs from the start of the planning horizon until all its assigned lots are completed. The "average machine time" is the average over all machines with assigned lots even though they may be idle for a portion of the time. Recall that it may not be possible to immediately process a lot on its assigned machine until its previous pass is finished on some other machine. As a result, a machine may be idle for a while, waiting for its next lot to arrive. The machine time includes this idle time.

The average machine time reported in the solution to the ASC model is 2.26% less on average than for the GRASP. In all but two of the nine instances, the three-phase methodology used less machine time than the GRASP. When the statistics in Tables 13, 16 and 17 are viewed collectively, the advantage of the ASC approach is evident.

Table 16: Comparison of number of machines

Prob. no.	GRASP number of machines	ASC number of machines	Percent Δ (%)
1	36	35	2.78
2	36	35	2.78
3	36	32	11.11
4	36	35	2.78
5	36	35	2.78
6	36	32	11.11
7	36	36	0
8	36	36	0
9	36	36	0
Avg.	36	34.67	3.70

Table 17: Comparison of average machine time

Prob. no.	GRASP average machine time (hr)	ASC average machine time (hr)	Percent Δ (%)
1	22.97	21.43	6.69
2	23.39	21.6	7.64
3	20.72	20.69	0.14
4	23.29	22.27	4.38
5	23.52	22.73	3.34
6	20.82	21.82	-4.79
7	23.52	22.96	2.38
8	23.37	23.03	1.45
9	22.07	22.27	-0.9
Avg.	22.63	22.09	2.26

Table 18 reports the CPU time used by the two approaches. The lower values associated with the GRASP for this metric are partly due to the fact that the ASC model was solved with CPLEX which was set to run up to four threads at a time, while the GRASP was programmed to use only a single thread. The CPU time counts time used by all threads. Without going into too much detail, in an ideal situation, when the real time advances 10 seconds, a four-thread code uses roughly 40 seconds, as opposed to 10 seconds for the single-thread code. In the planning environment, it may be necessary to trade superior performance for reduced CPU time, or at least reduced real time. For problem nos. 1 – 8, the three-phase methodology required all of the allotted 3600 seconds.

The above results are inclusive of the reductions realized after machine changeovers take place. In the case of the ASC methodology, the third phase provides a 7.56% improvement, on average, over the second phase solution. For the GRASP, the improvement is 8.25% on average.

Table 18: Comparison of CPU time

Prob. no.	GRASP CPU time (s)	ASC CPU time (s)	Percent Δ (%)
1	797.40	10,691	-1240.73
2	790.85	11,011	-1292.3
3	654.57	12,434	-1799.57
4	660.34	11,638.5	-1662.5
5	685.71	11,315.7	-1550.22
6	577.36	12,261.3	-2023.68
7	536.61	10,478.6	-1852.74
8	509.28	9,744.28	-1813.34
9	424.91	792.51	-86.51
Avg.	626.34	10,040.77	-1480.18

7.3.2 Assignment vs. Sequencing Solutions

The assignment model (3a) – (3l) represents a relaxation of the full problem because it ignores the order in which lots are processed and it excludes changeovers. Nevertheless, it is still a large scale MILP and difficult to solve. The sequencing model (4a) – (4m) takes the solution to the assignment model as input and produces an optimal sequence for the assigned lots. This may require removing or delaying some lots to achieve feasibility. However, we do not allow the tooling to be changed or new lots to be assigned. The time we allotted to solve either model reflects their relative difficulty.

Tables 19 and 20 summarize the size and performance statistics for all nine instances. As seen from Table 19, the assignment model averages 26,538 constraints and 42,451 variables, the majority of which are binary. CPU times averaged 6,116 seconds and the optimality gap ranged from 0.18 to 17.48%, averaging 8.94%. The comparable statistics for the sequencing model are given in Table 20, which indicates much smaller instances. The average of number of constraints is 5,486 and the average number of variables is 3,231. The CPU times and optimality gaps were similarly smaller,

averaging 3,894.62 seconds and 2.10%, respectively. Looking at problem 1, for example, the assignment model has roughly five times the number of constraints and thirteen and half times the number of variables as the sequencing model, and achieved an optimal gap of 9.01% in about 6,354 CPU seconds. In contrast, the sequencing model reached an optimal gap of 1.23% in 4,320 CPU seconds.

Table 19: Input and output statistics for assignment model

Prob. no.	No. of constraints	No. of variables	CPU time (s)	Optimal gap (%)
1	27,717	44,092	6,354.44	9.01
2	27,717	44,092	6,627.65	10.48
3	24,507	39,974	8,058.66	1.24
4	27,844	44,305	7,283.04	9.12
5	27,844	44,305	7,154.08	16.44
6	24,634	40,187	7,846.21	1.50
7	28,004	44,625	6,056.38	15.01
8	28,004	44,625	5,590.86	17.48
9	22,569	35,858	75.24	0.18
Avg.	26,538	42,451	6,116.28	8.94

Table 20: Input and output statistics for sequencing model

Prob. no.	No. of constraints	No. of variables	CPU time (s)	Optimal gap (%)
1	5,546	3,272	4,319.73	1.23
2	5,584	3,294	4,362.26	2.50
3	4,404	2,626	4,350.87	1.46
4	5,870	3,446	4,333.55	2.01
5	5,699	3,338	4,129.34	4.76
6	4,725	2,810	4,391.01	1.38
7	7,535	4,382	4,386.04	2.69
8	4,168	2,484	4,108.14	2.03
9	5,847	3,426	670.66	0.82
Avg.	5,486	3,231	3,894.62	2.10

Table 21 reports the differences in the number of lots processed and makespan of the solutions provided by the two models. The entries under column heading "No. of lots processed" denotes how many lots are assigned in the solution provided by the respective models. The entries in the column with heading "No. lots removed" is the difference between the two previous columns. The sequencing model removed between 2 to 12 lots from the assignment model solutions. For all nine problem instances the average number of lots removed was 5.78 lots.

The major heading "Total machine time" refers to the sum of the machine time of all machines used in the solutions of the two models. The entries in the last column "Total idle in sequence" is the sum of all idle hours between two consecutive lots on all machines in the solution to the sequencing model. Note that this is not just the difference between the "total machine time" derived from the assignment and sequencing models because some lots are removed by the sequencing model.

The percent difference for total machine time in Table 21 is calculated as follows:

$$Percent\ \Delta = 100 \times (sequencing\ value - assignment\ value) / assignment\ value$$

The statistics indicate that the sequencing model produced a 3.6% decrease on average in total machine time. For problem 9, however, the solution shows an increase of 0.68%, which results from an increase in total idle hours. This value is greater than the decrease in the number of hours associated with the removed lots. On average, the sequencing model solutions contain 6.42 total idle hours.

Table 21: Comparison of lots processed and machine time

Prob. no.	No. of lots processed			Total machine time			Total idle in sequence (hr)
	Assignment	Sequencing	No. lots removed	Assignment (hr)	Sequence (hr)	Percent Δ (%)	
1	246	243	3	750.84	728.64	-2.96	0.73
2	248	240	8	756.64	699.02	-7.62	0.12
3	210	206	4	654.98	634.90	-3.07	7.76
4	254	249	5	779.30	759.70	-2.51	6.43
5	243	231	12	798.36	737.13	-7.67	14.52
6	221	218	3	697.92	686.59	-1.62	1.86
7	305	297	8	827.14	799.23	-3.37	2.42
8	204	197	7	803.45	769.12	-4.27	12.67
9	247	245	2	764.80	769.97	0.68	11.28
Avg.	242	236	5.78	759.27	731.59	-3.60	6.42

Table 22 presents comparisons of the objective function values and the weighted sum of key device shortages provided by the assignment and sequencing model solutions. Recall that the sequencing model has a different objective function than the assignment model so to create a valid frame of reference, the assignment model solution was taken as the benchmark. In the table, the entries under the heading "Objective function value" and the subheading "Sequencing" are obtained by substituting the solution of the sequencing model into the objective function of the assignment model. Both models have a minimization objective. The results indicate that the sequencing model solutions are 18.24% greater on average than the assignment model solutions, and that the weighted sum of key device shortages increases by an average of 17.75 % from phase 1 to phase 2. This conforms with our expectations since the two terms are highly correlated due to the relative priority assigned to key device shortages, and the fact that the assignment model is a relaxation of the full problem. To obtain feasibility it was

necessary for the sequencing model to remove some lots from the assignment model solution.

Table 23 presents the last of the comparisons for the weighted sum of lots processed and the number of machines used, the second and third objective function terms in (3a). As in Table 22, the entries under the “Sequencing” subheadings were obtained by substituting the sequencing model solution values into the corresponding assignment model objective function terms. As might have been expected, the sequencing model processes fewer (weighted) lots than the assignment model, but averaged only 1.1% less over all nine instances. When all of the lots assigned to a machine are removed by the sequencing model, the machine is regarded as "removed." The entries under the heading "No. machines removed" indicate that this never happened. The number of machines is the same in the solutions of both models

Table 22: Comparison of objective function value and weighted sum of key device shortages

Prob. no.	Objective function value			Weighted sum of key device shortages		
	Assignment (10^{10})	Sequencing (10^{10})	Percent Δ (%)	Assignment (10^{10})	Sequencing (10^{10})	Percent Δ (%)
1	0.793	0.887	11.86	0.847	0.941	11.11
2	2.15	2.69	25.36	2.20	2.74	24.74
3	1.01	1.05	4.43	1.03	1.07	4.34
4	2.18	2.52	15.46	2.23	2.57	15.08
5	1.57	2.23	41.47	1.63	2.28	39.94
6	8.48	8.65	2.05	8.50	8.67	2.05
7	2.20	2.70	22.47	2.27	2.76	21.85
8	4.34	6.09	40.31	4.38	6.13	39.87
9	8.31	8.37	0.75	8.34	8.40	0.75
Avg.	3.45	3.91	18.24	3.49	3.95	17.75

Table 23: Comparison of weighted sum of lots and number of machine used

Prob. no.	Weighted sum of lots processed			No. of machine used		
	Assignment (10^8)	Sequencing (10^8)	Percent Δ (%)	Assignment	Sequencing	No. machines removed
1	5.33	5.32	-0.2	35	35	0
2	5.26	5.22	-0.59	35	35	0
3	1.73	1.71	-1.08	31	31	0
4	5.34	5.32	-0.47	35	35	0
5	5.44	5.20	-4.31	35	35	0
6	1.61	1.60	-0.62	32	32	0
7	6.18	6.16	-0.28	36	36	0
8	4.51	4.40	-2.34	36	36	0
9	3.27	3.27	0	35	35	0
Avg.	4.30	4.25	-1.10	34.44	34.44	0

From Tables 13 – 17 and 21 – 23, we can see how the solution changes through the assignment, sequencing, and changeover phases of the methodology. Take problem 1 as an example. The objective function value obtained by solving the assignment model is 7.93×10^9, as indicated in Table 22. When the sequencing model is solved, three lots are removed and 0.73 idle hours are added, as seen in Table 21. Consequently, the objective function value increased by 11.86% to 8.87×10^9. During the changeover procedure, some machines are reset with new tooling and assigned new lots. Table 13 shows that the objective function value decreased to 8.47×10^9. A similar progression can be seen in the value of the weighted sum of key device shortages, the first objective function term in (3a).

Chapter 8: Real-time Decision Support for AT Operations

The output of model (1) for basic AT is used to establish machine setup and lot assignment targets for daily operations. To aid shop floor personnel in reaching these targets we have developed two complementary procedures for prioritizing choices whenever an oppor tunity for machine changeovers becomes available. At TI's facilities, the data needed to support the review process are obtained from their factory database system. A query to the system returns a snapshot of WIP, current machine setups, tooling availability, and lot loadings. From the lot loadings, we can calculate the time at which the active machines along with their tooling will become free. It is assumed that only a handful of changeovers are possible due to limited personnel, and that preemption is not allowed.

During the review process, which may be as frequent as every 15 minutes, if a machine is currently set up in accordance with the target solution, it, as well as the lots assigned to it, are omitted from the analysis. Of the remaining machines, those that are free or will be become free in, say, Δ^{free} minutes are given the highest priority. Typically, rule-based procedures are used at this level of control (e.g., see Dolgui and Proth 2010, Liu et al. 2011) but are seldom effective from a global perspective. Experience at TI's facilities suggests that much greater output can be realized by continually adjusting machine setups to match the maximum capacity solution.

Our procedures were designed with this goal in mind and consist of two parts: (i) a comparison between the current tooling setup and the target tooling setup (i.e., the maximum capacity solution) for all available machines; (ii) construction of a priority list that provides recommendations for changing over the machines that are not set up in accordance with the target solution. To be precise, a *setup* refers to a combination of a

tooling piece(s) from a particular family and a corresponding certification or temperature at which lots are to be tested. The phrase "when a machine becomes free" means the time at which the machine under consideration finishes processing its current lot and, by assumption, can release the tooling installed on it.

In the methodology, the waiting time parameter Δ^{free} serves two purposes. First, it specifies the time increment starting from the current time, t_0, for defining priority classes. Machines that can be reset with their target tooling between t_0 and $t_0 + \Delta^{free}$ are placed in the top class and ranked based on their weight (discussed presently). Similarly, machines that can be reset between $t_0 + \Delta^{free}$ and $t_0 + 2\Delta^{free}$ define the second class and so on. Second, it specifies the maximum amount of time a machine can wait for tooling to become available or to finish its current lot in order for it to be placed in the top class. If a machine cannot be reset within Δ^{free} hours from t_0 for any reason, then it is said to be *unavailable* but is still ranked based on either its weight or the time when it will become available.

8.1 COMPARISON OF CURRENT AND MAXIMUM CAPACITY SOLUTIONS

The first algorithm reads the initial setup and target solution files to determine which machines are eligible for changeover during the current review process. The main complication arises from the fact that identical machines exist in the same family but their initial setups might not be aligned with the target solution. To avoid overlooking equivalent setups it is necessary to perform a series of comparisons and relabeling operations. The steps are delineated below but first we describe the output.

Table 24 displays a portion of the `Compare_Algorithm` output. Column 1 lists the machine families and column 2 identifies the machine instance 'id.' Columns 3 and 4 give the current tooling family and certification on the machine, respectively. If

these fields are empty, it means that the machine is not set up at t_0. The target tool family and certification are identified in columns 5 and 6. Column 7 (second portion of the table) specifies when the machine will finish its current lot. If the entry is "Now," this means that the machine is currently empty (no lots are running on it); otherwise the day and time are given. Column 8 specifies whether or not the machine needs to be reset – a determination based on the current and target setups.

Column 9 lists a machine whose target setup is equivalent to the setup of the machine in the row under consideration. For example, row 3 is associated with AMAT27-1 and is currently configured with a tooling piece from family 6473283C. An equivalent machine, AMAT02-1, has the same setup in the target solution. If the field is empty, it means no such machine exists in the solution provided by model (1). The last two columns 10 and 11 give the target setup. If a machine does not need to be reset, then the target will be "Cur_Tooling" and "Cur_Certification." In those cases in which there is an entry in the Equivalent_Machine column, there is no need for a resetup because the current setup on the machine is just the target setup of another machine from the same family. As such, the machine under consideration can switch the target with the machine indicated in in the Equivalent_Machine column. This is handled by re-indexing the machines. For example, assume that machines i_1 and i_2 are currently set up with tooling-temperature configurations λ_1 and λ_2, respectively, while the target solution calls for i_2 to be set up according to λ_1 and i_1 according to λ_3. If these two machines are in the same family and hence interchangeable, then their indices in the target solution should be reversed as long as there are no other machines in that family set up according to λ_3.

Table 24: Example of output from comparison algorithm

Machine_Family	Machine_Instance	Cur_Tooling	Cur_Certification	Tar_Tooling	Tar_Certification
ETS-1-64	AMAT01-1			6492377B	1
ETS-1-64	AMAT02-1			6473283C	1
ETS-1-64	AMAT27-1	6473283C	1	6463103B	1
ETS-1M-64	AMAT23-1	6463103B	1	6473198B	1
ETS-2-64	AMAT35-1			6469171D	1

When_Free	Resetup_Needed	Equivalent_Machine	Final_Tar_Tooling	Final_Tar_Certification
Now	Y		6492377B	1
Now	Y		6463103B	1
11/5/2010 15:16	N	AMAT02-1	Cur_Tooling	Cur_Certification
Now	Y		6490924B	1
Now	Y		6466496A	1

8.2 COMPARISON ALGORITHM

`Compare_Algorithm`

Step 0. Let M = {all machines}, M_1 = {machines with initial setups}, M_2 = {machines without initial setups}, $n = |M_1|$, and m_k denote the k^{th} machine in M.

Step 1. Read "initialsetup.csv" (see Table 1). For each machine in M_1, record its family id, instance id, corresponding tooling family id, and certification. If a machine's tooling family id and certification are empty, then mark the machine as belonging to the set M_2; otherwise, mark it as belonging to the set M_1.

Step 2. Read "solution.csv". For each machine in M_1, record the family id, instance id, tooling family id, and certification for the setup associated with the first lot assigned to the machine that is not an initial lot. Let this information be the machine's target. If a machine has not been assigned any lot other than the initial lot to process, then record the tooling family id and certification for the initial lot as the machine's target. Record the time when the machine is expected to finish its initial lot. If a machine doesn't have an initial lot or can finish the initial lot before the beginning of the planning horizon denoted by t_0, then the time "When_Free" is marked as "Now."

Step 3. For each machine in M_1, compare its target setup with its initial status. Set $k = 1$. Do while $k \leq n$.

3a. If machine $m_k \in M_2$, then a "Y" is entered in the column in Table 24 labeled "Resetup_Needed"; put $k \leftarrow k+1$ and continue. Otherwise, go to Step 3b.

3b. Check whether the target setup of machine m_k is the same as its initial tooling setup. If yes, then an "N" is entered in the column labeled

"Resetup_Needed," put $k \leftarrow k+1$, and go to Step 3a; otherwise, go to Step 3c.

3c. Check whether the initial tooling setup of machine m_k is the same as the target setup of some other machine in the same family whose target has not yet been achieved. If yes, then an "N" is entered in the column labeled "Resetup_Needed" and the other machine's target tooling setup switch to the target tooling setup of machine m_k; otherwise the entry is "Y," we put $k \leftarrow k+1$, and go to Step 3a.

End while (at termination, let $\bar{M}$ denote the set of eligible machines)

8.3 PRIORITY LIST CONSTRUCTION

The `Priority_List_Algorithm` provides an ordered list of recommendations for resetting all eligible machines. The format is illustrated in Table 25. Columns 1 and 2 list the name of the machine family and machine instance id, respectively. Column 3 gives the length of time relative to the current time when a machine processing an initial lot will finish that lot and become available. For each machine, columns 4 and 5 provide the target setup obtained from the `Compare_Algorithm`. Column 6 indicates the number of hours from the current time t_0 when a changeover is possible.

Table 25: Example of priority computations†

Machine_Family	Machine_Instance	Time_to_Mach_Free (hr)	Final_Tar_Tooling
ETS-1M-64	AMAT23-1	0	6490924B
ETS-1-64	AMAT01-1	0	6492377B
ETS-1-64	AMAT02-1	0	6463103B
ETS-2-64	AMAT35-1	0	6466496A

Final_Tar_Cert	Time_to_Reset (hr)	Tool_Source	Resetup_Priority	Benefit_value
1	0	Inventory	1	7.42E+06
1	0.401	AMAT17-1	2	6.35E+06
1	0	Inventory	3	1.14E+06
1	0.396	AMAT30-1	18	0.12E+06

†Although AMAT27-1 appears in Table 24, it is absent from Table 25 because it doesn't need to be reset

Column 7 specifies the source of the target tooling. If the entry is "Inventory," it means that the required tooling will be available when the machine finishes the lot that it is currently processing. By implication, if the machine is idle, then the tooling should also be available. If the entry is the name of a machine instance, e.g., "AMAT30-1," it means that a changeover cannot occur until the machine instance in the corresponding row finishes its current lot and releases its tooling. If the entry is "null," it means that the target tooling is not expected to become available in the time remaining in the planning horizon. This situation occurs when the needed tooling is neither in "inventory" nor on any machine(e.g. when some toolings are broken).

Column 8 gives the priority for resetting each available machine in decreasing order. These values are calculated with the algorithm described in the next subsection. Column 9 indicates the benefit that would result if a machine were reset to match the target tooling and temperature specified by the solution to model (1). It is the largest

possible weighted sum of lots that would be processed on that machine according to the maximum capacity solution.

8.3.1 Rules for setting the priorities

The values in the last column of Table 25 indicate the benefit of a setup between t_0 and the end of the planning horizon. The machines that can be reset in the interval [t_0, $t_0 + \Delta^{free}$] fall into the highest priority class and take precedence over those that either becomes available or whose target tooling becomes available after $t_0 + \Delta^{free}$. The same logic applies for machines that can be reset between $t_0 + \Delta^{free}$ and $t_0 + 2\Delta^{free}$ with respect to those that cannot be reset in this interval, and so on, until the end of the planning horizon is reached.

The procedure for setting the priorities includes the following steps: compute the *benefit* value, as described in the next section, of resetting each machine that is free in the interval [t_0, $t_0 + \Delta^{free}$]; determine which machines can be reset in the interval [t_0, $t_0 + \Delta^{free}$] and when they can be reset (tooling may not be available so a machine that is free may not be able to be reset); rank the machines that can be reset in the interval [t_0, $t_0 + \Delta^{free}$] based on their benefit value and assign lots to the machines. The same procedure is applied to the machines that are free between $t_0 + \Delta^{free}$ and $t_0 + 2\Delta^{free}$, including the machines that are free but not reset in the previous interval, and so on until the end of the planning horizon. Further explanation is now given.

8.3.2 Calculations

To perform the rankings, we first compute a benefit value for machines that can be reset in the upcoming time interval and then use the above rules to construct an order. The following algorithm is used for this purpose.

`Benefit_Calculation_Algorithm`

Step 0. Initialization

(a) From the file "solution.csv," identify all machine-family-tooling-setup combinations used in the target solution. Let the following set contain these combiantions: MF_TF_CERT = {(mf, tf, $cert$) : machine family mf , tooling family tf, and certification $cert$ is used in the target solution}.

(b) For each machine family, identify machine intances eligible for changeover during the current time interval according to comparion result as well as those previously identified but not reset. Let $MF\_MI(mf)$ = {mi : machine instance mi belongs to machine family mf and is free during the current time interval}.

(c) For each tooling family, identify which tooling instances are available during the current time interval. Let $TF\_TI(tf)$ = {ti : tooling instance ti belongs to tooling family tf and is available during the current time interval}. The tooling available can be determined from the data in the files "tooling.csv" and "initialsetup.csv," and the results for the calculations associated with the previous time intervals.

(d) Identify all lots associated with the target solution that have not yet been assigned to a machine using the data in the file "solution.csv" and the results from the previous time intervals. Denote the set of available lots as L.

Step 1. Compute the weighted sum of available lots that can be processed for each combination (mf, tf, $cert$) $\in$ MF_TF_CERT according to the target solution. Rank the elements in MF_TF_CERT based on the weighted sum of lots in descending order. Break ties arbitrarily.

Step 2. From the ranked list *MF_TF_CERT*, choose the first (*mf*, *tf*, *cert*) such that there exists an $mi \in MF\_MI(mf)$ and a $ti \in TF\_TI(tf)$ that can both operate under certification *cert*. If no eligible (*mf*, *tf*, *cert*) can be found, then stop.

Step 3. Choose the machine instance $mi \in MF\_MI(mf)$ that becomes available at the earliest time, and an arbitrary tooling instance $ti \in TF\_TI(tf)$. If several machine instances are free at the same time, choose the one whose target tooling family is *tf*.

Step 4. Rank the lots in the target solution that are processed by (*mf*, *tf*, *cert*) from highest to lowest based on *weight_rate*, given by

$$weight\_rate = \frac{weight}{CUR\_QTY / PPH}$$

where

weight = entry in the column "weight" in WIP file (see Table 3), which is the benefit obtained by processing the corresponding lot

CUR_QTY = number of devices contained in the lot

Step 5. From the ranked lot list found in Step 4, select the lots from top that can be processed within the time horizon remaining for the *mi* chosen in Step 3. Sum the weights of the chosen lots to get the *benefit* value associated with the combination (*mf*, *tf*, *cert*).

Step 6. If the target tooling setup for *mi* as determined by `Compare_Algorithm` is different than the tooling family and certification associated with (*mf*, *tf*, *cert*), then there must be some other machine instance in *mf* whose target tooling setup is just (*tf*, *cert*). Swap the target toolings between these two machine instances.

Step 7. Update *MF_MI*(*mf*), *TF_TI*(*tf*) and *L*. Mark the corresponding machine instance, lots and tooling as unavailble for calculating subsequent *benefit* values. Go to Step 1.

Step 0 identifies which machines, toolings and lots that are free in the current time interval. Step 1 ranks the machine-family-tooling-setup combinations in the target solution. In Step 2, the first machine-family-tooling-setup combination from the ranked list is chosen provided that both the machine and its specified tooling are free. Step 3 indicates how to select the machine instance and tooling intance for the setup chosen in Step 2. Step 4 ranks the lots availabe for computing benefits. The *weight* and *CUR_QTY* values can be obtained directly from the WIP file. To get the *PPH* , the route used for processing the lot must be identified from "solution.csv," as can be seen in Table 2. In Step 5, the benefit value is calculated by assigning lots to the machine under consideration. Note that the lots are only "assigned" for the purposes of the calculation. In Step 6 the tooling on the current machine is switched with the tooling on an identical instance to match *tf* if need be. Step 7 updates the availability of machines, tooling and lots and removes the chosen setup from future consideration.

The algorithm terminates with a benefit value for each machine that can be reset in the upcoming interval. We now discuss how the changeover prioity list is actually constructed.

Criteria for determining when a machine is reset. For purposes of determining if and when a changeover is possible, three machine classes are defined. The first corresponds to the case where sufficient tooling is or will be available in the time interval when the machine becomes available. If a machine in this class is currently idle, then the target tooling must be in inventory or will be freed up in the current time interval. The second class is similar to the first but now only a subset of the required tooling is or will be

available in the current time interval when the machines become free so they compete for the tooling. We need to decide which machines are reset immediately, which ones will be reset within the current time interval, and which ones will be reset later when additional tooling is released. This determination is based on the benefit value associated with the competing machines. A higher benefit translates into greater consideration. The third class includes those machines whose tooling is not available in the time interval when they become free, which may be at the current time or in the future. In this case, the target tooling is in use so all machines sharing the tooling must wait until the next time interval before any action is taken, and, by definition fall into a secondary priority class.

After each machine that is not currently set up in accordance with its target tooling is placed in one of the three classes, a priority list is constructed that makes use of the results obtained from the `Benefit_Calculation_Algorithm`. The procedure is as follows.

`Priority_List_Algorithm`

Step 0. Set $\bar{L} = \varnothing$. Initialize priority list.

Step 1. Identify a subset of the machines M_0 eligible for resetting within the interval $[t_0, t_0+\Delta^{free}]$.

Step 2. Calculate the benefit for all machines in M_0 using `Benefit_Calculation_Algorithm`.

Step 3. Determine when a machine in M_0 should be reset following the logic in Section 8.3.1.

Step 4. Identify the machines that cannot be reset in the interval $[t_0, t_0+\Delta^{free}]$ and remove them from M_0.

Step 5. Rank the machines in M_0 according to their calculated *benefit* from highest to lowest.

Step 6. Add the machines to the priority list $\bar{L}$. Assign lots to the machines using the greedy randomized procedure discussed in Section 8.3.3. Record the tooling and lots assigned to the machines. Mark the corresponding lots, machines and tooling pieces as unavailable for resetting.

Step 7. Put $t_0 \leftarrow t_0 + \Delta^{free}$. If t_0 is still within the planning horizon, then go to Step 1; otherwise, stop.

To illustrate how the algorithm works, let Δ^{free} = 1 (hr) and the end of the planning horizon be $t_0 + 2$. Table 26, 27 and 28 list all the machines, tooling, and initial setups for this example. In Table 26, columns 1 and 2 list the name of the machine family and machine instance id, respectively. Column 3 gives the length of time relative to t_0 when a machine now processing a lot will finish that lot and become free. For each machine, columns 4 and 5 provide the target setup obtained from the `Compare_Algorithm`.

In Table 27, columns 1 and 2 list the name of the tooling family and tooling instance id, respectively. In the example, there are three tooling families and four tooling instances. In Table 28, column 1 gives the machine instance id. Column 2 and 3 provide the name of tooling family and the certification respectively for the initial setup of the corresponding machine. Initially, machine instance AMAT03-1 is configured with tooling instance T02-1 and AMAT04-1 is configured with T03-1. Therefore T02-1 and T03-1 are not available until AMAT03-1 and AMAT04-1 finish their initial lots.

Table 26: Machines for example of `Priority_List_Algorithm`

Machine_Family	Machine_Instance	Time_to_Mach_Free (hr)	Final_Tar_Tooling	Final_Tar_Cert
ETS-1M-64	AMAT01-1	0	6490924B	1
ETS-1-64	AMAT02-1	0	6490924B	1
ETS-1-64	AMAT03-1	1.2	6463103B	1
ETS-2-64	AMAT04-1	1.6	6466496A	1

Table 27: Tooling for example of `Priority_List_Algorithm`

Tooling_Family	Tooling_Instance
6490924B	T01-1
6490924B	T02-1
6463103B	T03-1
6466496A	T04-1

Table 28: Initial tooling setups for example of `Priority_List_Algorithm`

Machine_Instance	Initial_Tooling_Family	Initial_Tooling_Instance	Initial_Certification
AMAT03-1	6490924B	T02-1	1
AMAT04-1	6463103B	T03-1	1

Applying the `Priority_List_Algorithm` to the example, we see at Step 1 from the column "Time_to_Mach_Free" in Table 26, that AMAT01-1 and AMAT02-1 are free within the interval $[t_0, t_0+\Delta free]$. This gives M_0 = {AMAT01-1, AMAT02-1}. At Step 2, we calculate the benefit value for these two machines using the `Benefit_Calculation_Algorithm` and get 2500 for AMAT01-1's and 2000 for AMAT02-1. Next, at Step 3, we determine when each machine in M_0 should be reset. The target tooling for both AMAT01-1 and AMAT02-1 are the family 6490924B but only instance T01 is available between t_0 and $t_0+\Delta free$ (that is, T01 is not in use at t_0).

Thus only one machine instance can be reset within $[t_0, t_0+\Delta free]$. Since AMAT01-1's benefit is greater than AMAT02-1's, the former is chosen to reset at time t_0.

At Step 4, AMAT02-1 is removed from M_0 giving M_0 = {AMAT01-1}. Since there is only one machine in M_0, by default, it is place in the first position on the priority list $\bar{L}$ at Step 5. Lots are then assigned to AMAT01-1, and the lots, machine and tooling instances are marked as unavailable. Putting $t_0 \leftarrow t_0 + \Delta free = t_0 + 1$ which is less than the planning horizon denoted by $t_0 + 2$, the algorithm returns to Step 1 and repeats. The remaining calculations can be obtained from the authors. The results are shown in Table 29.

Table 29: Example results for `Priority_List_Algorithm`

Machine_Family	Machine_Instance	Time_to_Mach_Free (hr)	Final_Tar_Tooling
ETS-1M-64	AMAT01-1	0	6490924B
ETS-2-64	AMAT04-1	1.6	6466496A
ETS-1-64	AMAT02-1	0	6463103B
ETS-1-64	AMAT03-1	1.2	6463103B

Final_Tar_Cert	Time_to_Reset (hr)	Tool_Source	Resetup_Priority	Weight
1	0	Inventory	1	2500
1	1.6	Inventory	2	3000
1	1.2	AMAT03-1	3	2000
1	1.6	AMAT04-1	4	2000

A second approach for constructing the priority list has also been developed. It emphasizes the time when a machine can be reset without regard to Δ^{free}. The implementation involves replacing Step 5 above with the following.

Step 5. Rank the machines first based on the time when they can be reset with their target tooling and then by their benefit. The second criterion is used to break ties.

The results of the computations are contained in the summary file "priority.summery.txt." The following data are provided.

Total number of machines that need resetting: total number of machines in "priority_list.csv" that are candidates for resetting. This value excludes machines that are currently set up with the target tooling.

Total number of machines that need resetting within time limit: number of machines that become idle between t_0 and $t_0 + \Delta^{free}$.

Number of machines ranked within the time limit: number of machines that become free and whose target tooling is available between t_0 and $t_0 + \Delta^{free}$. These machines are ranked.

Number of machines not ranked within time limit: number of machines that are idle between current time t_0 and $t_0 + \Delta^{free}$ but cannot be reset in this interval because their target tooling is not available.

8.3.3 Greedy randomized procedure for lot assignments

In Step 6 of the `Priority_List_Algorithm`, lots are assigned to machines for the given tooling setup. Rather than making these assignments in decreasing order of the value of their *weight_rates*, as calculated in Step 4 of `Benefit_Calculation_Algorithm`

A greedy randomized proedure is used to diversify the search for good solutions. A purely myopic approach is likely to suffer from "end" effects as the number of options deceases when only a few lots remain. For example, assume that we have a machine m_1 with 24 hours remaining, and four lots with *weight_rates* ordered such that $l_1 = 100 = l_2 =$

$100 > l_3 = 99 = l_4 = 99$. The times required for processing l_1, l_2, l_3, and l_4 are 13, 8, 5 and 5 hours, respectively. The greedy solution would have l_1, l_2 assigned to m_1. A better solution would have l_1, l_3, and l_4 assigned to m_1.

Our methodology consists of a construction phase only. Starting with the first machine on the priority list $\bar{L}$ that has not been assigned any lots, we randomly choose a lot from the top five candidates in a ranked list until all the available time on the current machine is consumed. The result is recorded and the process is repeated for a total of 10 iterations. The assignments asssociated with the highest weighted sum of lots found during the 10 iterations represents the solution for the machine. The procedure is repeated applied to each machine in $\bar{L}$ that is available in the current time period.

It should be mentioned that an improvement phase based on swaps could have been added after all machines are assigned lots to get a fulfledged GRASP. Although theoretically, swapping target tooling between two machines and reassigning lots to the machines could result in better solutions, we found that after some initial experimentation the effort spent just trying to stay feasible within a neighborhood proved too time consuming to be effective.

`Greedy_Lot_Assignment_Algorithm`

Step 1. (Initialization)

1a. For the machine-tooling setup under consideration, read the file "solution.csv" to identify all lots processed by the same machine family and tooling setup in the target solution. Exclude any lots on machines at t_0 (i.e., initial lots) and those that have already been assigned.

1b. Choose routes from the file "route.csv" for the lots found in Step 1a. The routes correspond to the same machine family, tooling family, and

certification as the machine-tooling setup under consideration. For each lot choose the route that has the highest processing rate in terms of *PPH*.

1c. Compute *weight_rate* for lots found in Step 1 in the same way as in the Step 4 of `Benefit_Calculation_Algorithm`. Rank the lots based on *weight_rate*.

Step 2. Set $k = 1$. Record the ranked lot list in Step 1c as *lot_list_original*. Let *lot_assigned_opt* record the optimal weighted lots assigned to the machine. Let *weighted_sum_opt* = 0.

Step 5. If $k > 10$, stop, Otherwise, let *lot_list_temp* = *lot_list_original*, and create a new list, call it *lots_assigned*, which holds the lots assigned to the machine. Let *weighted_sum* = 0;

Step 6. If the number of lots in *lot_list_temp* is 0, then go to Step 8. Otherwise, randomly choose a lot from the top 5 (or from the remaining number, if less) of *lot_list_temp*.

Step 7. Remove the lot found in Step 6 from *lot_list_temp*. If the time required for processing the lot found in Step 6 is less than the time remaining for the machine, add its weight to *weighted_sum*, and place in *lot_assigned*; otherwise, discard it. Go to Step 6.

Step 8. Put $k \leftarrow k + 1$. If *weighted_sum* > *weighted_sum_opt*, put *weighted_sum_opt* = *weighted_sum*, *lot_assigned_opt* = *lot_assigned*. Go to Step 5.

8.4 MIXED-INTEGER PROGRAMMING MODEL FOR REAL-TIME CONTROL PROBLEM

To validate the results from our real-time control heuristic, we have also developed two competing MIPs that provide "optimal" changeover and scheduling recommendations similarly based on real-time data inputs. Each makes use of the

maximum capacity solution obtained from (1a) – (1i). The first tries to minimize the weighted sum of the absolute deviations from the target solution and the second tries to maximize the weighted sum of processed lots. The assumptions underlying the MIPs are the same as those enumerated before section 8.1 within Chapter 8. If a machine is currently set up in accordance with the target solution, it, as well as the lots assigned to it, are omitted from the analysis. Of the remaining machines in M, only those that are free or will be become free in Δ^{free} minutes are included in the problem instance. The corresponding formulations are given in Appendix E.

8.5 COMPUTATIONAL EXPERIMENTS

The complementary algorithms described from Section 8.1 to Section 8.3 were tested using the same data described in Section 6.4. The basic data set consists of 36 machines, 284 tooling pieces from 6 families, 1036 lots, and 1 temperature (certification) setting, and served as the basis for randomly generating eight additional data sets. Randomly generated data based on this real case is also used [for further explanation, see Section 6.4]. For discussion purposes, machines processing lots at t0 are called "initial machines" and generally have tooling installed on them.

Procedures, `Compare_Algorithm` and `Priority_List_Algorithm`, were implemented in C++ and run under Ubuntu Linux on a Dell Poweredge 2950 workstation with 2 dual core hyperthreading 3.73 GHz Xeon processors and 8 GB of memory. CPLEX 12.1 was used to solve the two MIPs described in Appendix E without changing its default settings.

8.5.1 Results for *`Compare_Algorithm`*

In all the testing, the waiting time limit Δ^{free} was set to 1 hr. The results appear in Tables 30 – 32 whose column headings are defined as follows.

Total no. machines: Total number of machines in the problem instance including the machines that are already set up according to the target solution

Total no. tools: Total number of tooling pieces including those in use

No. need reset: Number of machines that are candidates for changeover during the remainder of the planning horizon

Free within Δ^{free}: Number of machines that become free between t_0 and $t_0 + \Delta^{free}$

Reset within Δ^{free}: Number of machines that can be reset between t_0 and $t_0 + \Delta^{free}$

Not reset within Δ^{free}: Number of machines that are idle between t_0 and $t_0 + \Delta^{free}$ but cannot be reset because target tooling is not available

Table 30: Compare results for group 1

Problem no.	Total no. machines	Total no. tools	No. need reset	Free within Δ^{free}	Reset within Δ^{free}	Not reset within Δ^{free}
1	36	284	23	12	11	1
2	36	284	25	16	14	2
3	36	284	22	13	12	1

Table 31: Compare results for group 2

Problem no.	Total no. machines	Total no. tools	No. need reset	Free within Δ^{free}	Reset within Δ^{free}	Not reset within Δ^{free}
4	36	284	29	20	19	1
5	36	284	29	22	20	2
6	36	284	25	21	19	2

Table 32: Compare results for group 3

Problem no.	Total no. machines	Total no. tools	No. need reset	Free within Δ^{free}	Reset within Δ^{free}	Not reset within Δ^{free}
7	36	284	33	29	29	0
8	36	284	34	29	27	2
9	36	284	34	29	29	0

In all but one case, 67% or more of the 36 machines needed to be reset at t_0. For the base case, problem 1, 12 machines are available in the upcoming hour and 10 of them can be reset. This suggests that there is ample tooling available to permit the changeovers. In all cases, no more than 4 machines that are candidates for changeover cannot be reset within the hour (see last column of tables).

8.5.2 Results for Priority_List_Algorithm

Table 33 reports the results for Problem no. 1 listed in Table 30. Of the 23 machines that are candidates to be reset, column 3 indicates that 12 are available for changeover in the upcoming hour. Column 4 gives the target tooling and column 5 indicates the time from t_0 when either the tooling or machine will become free. Column 6 identifies the source of the tooling. For machines AMAT30-1 and AMAT15-1 in rows 12 and 18, the tooling is currently installed on two other machines and will not be available in the hour. For the remaining 21 cases, the tooling is currently available so any delay noted in column 5 is a consequence of the machine still processing a lot. The "Benefit_value" entered in the last column is the measure used to rank the changeovers. The large values reflect the wide variation in lot weights; lots with key devices may be assigned weights that are several orders of magnitude greater than regular lot

Table 33: Priorities for resetting available machines

Machine family	Machine instance	Time to machine free (hr)	Final target tooling	Time to reset (hr)	Tool source	Reset priority	Benefit value
ETS564	AMAT13-1	0	6462741B	0	inventory	1	8.50E+07
ETS-2-64	AMAT33-1	0	6504853A/6487463C	0	inventory	2	5.37E+07
ETS-0-64	AMAT31-1	0	6473198B	0	AMAT16-1	3	2.95E+07
ETS-1M-64	AMAT22-1	0	No_Tooling	0	inventory	4	2.33E+07
ETS-1M-64	AMAT26-1	0	6469171D	0	inventory	5	1.16E+07
ETS-1-128	AMAT16-1	0	6501065B	0	inventory	6	4.74E+06
ETS564	AMAT06-1	0	6440109A	0	inventory	7	831902
ETS-2-64	AMAT35-1	0	6481146C	0	inventory	8	199202
ETS564	AMAT05-1	0	6453620A	0	inventory	9	58000
ETS564	AMAT04-1	0	6442302C	0	inventory	10	23404
ETS564	AMAT07-1	0.045	6462741B	0.045	inventory	11	1
ETS-0-64	AMAT30-1	0.308056	6473198B	1.46861	AMAT21-1	12	2.95E+07
ETS-1M-64	AMAT21-1	1.46861	No_Tooling	1.46861	inventory	13	2.33E+07
ETS-1M-64	AMAT25-1	1.65611	6469171D	1.65611	inventory	14	1.16E+07
ETS-1-64	AMAT08-1	1.02222	6466496A	1.02222	AMAT30-1	15	4.45E+06
ETS-1-64	AMAT14-1	1.78694	6481146C	1.78694	inventory	16	263600
ETS-2-64	AMAT32-1	2.43278	6483172A	2.43278	inventory	17	3.07E+07
ETS-1-64	AMAT15-1	2.51472	6473283C	2.74694	AMAT34-1	18	4.86E+06
ETS-2-64	AMAT34-1	2.74694	6485429B	2.74694	inventory	19	81302
ETS-1M-64	AMAT23-1	3.72167	No_Tooling	3.72167	inventory	20	2.33E+07
ETS-1-64	AMAT11-1	3.68083	6479471B	3.68083	inventory	21	219300
ETS-1-128	AMAT12-1	7.75333	6501065B	7.75333	inventory	22	3.67E+06
ETS-1-64	AMAT19-1	8.75167	6490924B	8.75167	inventory	23	1.42E+06

8.5.3 Comparison of heuristic and MIP results

The nine problems listed in Tables 34 – 36 were analyzed and the results compared to the solutions obtained for Models (5) and (6) presented in Appendix E. For the MIPs, the maximum number of changeovers permitted was 10 and a 1-hour time limit was placed on the CPLEX runs. Tables 34 and 36 detail the output statistics obtained with the priority heuristic and CPLEX for Model 5, respectively. The objective values reported are those computed from (5a) and (6a); that is, $\sum_{l\in L}\sum_{r\in R(l)} w_{lr}\left|\sum_{i\in M(l,r)\subseteq\mathcal{M}(l,\lambda)} x_{ilr} - \sum_{i\in\mathcal{M}(l,\lambda)} X_{ilr}\right|$ and $\sum_{i\in M}\sum_{l\in L(i)}\sum_{r\in R(i,l)} w_{lr}x_{ilr}$, where M is the set of machines that are available for changeover in the upcoming hour. For problem no. 1, for example, the machines in M are those in Table 33 whose entry in column 3 is less than or equal to 1.

The first six columns in Tables 34 and 35 are self-explanatory. The 'Model 5 objective' value in column 2 is the sum of the weights of the target lots that are *not* processed by the same route as in the target solution as determined by substituting the heuristic solution into (5a). For the first 10 machines in Table 33, the weight sums are 0 because all targets are met. Thus, the value 6697×10^4 for problem no. 1 in Table 34 corresponds to the sum of the weights of lots that are not processed by the first 10 machines in Table 33. In contrast, the 'Total weight of lots' value in column 3 represents the sum of the weights of all the lots that can be assigned to the machines that are available within the hour, but only up to a maximum of n^{setups} which has a value of 10 in all the computations. Note the fact that exactly 10 machines can be reset in the solution contained in Table 33 is a coincidence.

Column 7 reports the average number of hours that the lots assigned to the reset machines will consume within the planning horizon, which is 24 hours for all cases.

Columns 8 and 9 give the maximum machine time and minimum machine time used, respectively, for the reset machines. The results indicate that for both the heuristic and CPLEX the full capacity of at least one machine in each problem set is used; the average over all cases was 20.6 hours. For the heuristic, there are several problems where some of the machines have significant unused capacity, while the CPLEX results indicate that there is at least one machine in all cases with significant unused capacity. The run times reported in the last column are negligible for the heuristic and anywhere from 33 to 278 sec for CPLEX which found the optimal solution for each problem set.

Table 34: Heuristic changeover results

Prob. no.	Model 5 objective (10^4)	Total weight of lots (10^4)	No. of lots	No. lots with key devices	No. of change-overs	Average mach. time (hr)	Maximum mach. time (hr)	Minimum mach. time (hr)	Run time (sec)
1	6697	20760	85	79	10	17.42	23.79	5.37	1
2	7190	19743	90	80	10	21.25	23.97	12.87	1
3	2627	4985	39	37	10	13.03	23.47	0.90	1
4	4169	30961	101	97	10	22.39	23.97	12.85	0
5	5827	31220	107	105	10	23.25	23.99	22.09	0
6	3476	9420	67	66	10	22.07	23.95	12.85	1
7	8010	28100	86	82	10	23.49	23.95	22.21	1
8	8079	28966	112	110	10	22.83	23.91	20.35	0
9	8881	17378	78	73	10	22.92	23.97	20.97	1

Table 35: CPLEX changeover results for Model 1

Prob. no.	Model 5 objective (10^4)	Total weight of lots (10^4)	No. of lots	No. lots with key devices	No. of change-overs	Average mach. time (hr)	Maximum mach. time (hr)	Minimum mach. time (hr)	Run time (sec)	Solution status
1	6698	20759	83	77	10	22.54	23.92	18.19	141	optimal
2	7180	19753	84	84	10	22.54	23.92	18.19	182	optimal
3	2621	4991	40	38	10	13.33	23.79	0.90	97	optimal
4	4173	30957	102	98	10	22.42	23.95	12.85	206	optimal
5	5827	31220	107	105	10	23.24	23.92	22.09	213	optimal
6	3428	9469	68	67	10	22.22	23.95	12.85	159	optimal
7	7931	28179	95	91	10	23.49	23.98	22.21	378	optimal
8	8078	28966	112	110	10	22.83	23.91	20.35	392	optimal
9	8881	17378	78	73	10	22.87	23.97	20.97	328	optimal

Table 36 identifies the percentage differences between the CPLEX and heuristic solutions provided in Tables 34 and 35, respectively. The results are nearly identical for the 'Model 5 objective' values and the ‘total weight of lots’ values in columns 2 and 3 respectively. For problem nos.1 and 4, although the solution status returned by CPLEX is "optimal," the heuristic produced better results. This is possible because the tolerance used for the case in CPLEX is 0.21%, which is higher than the percentage differences in Table 36 for these two problems. One additional point of interest is that the CPLEX processes a few more lots with key devices on average than the heuristic. In the analysis here, we do not distinguish between lot types. Only the lot weights are used to guide the changeovers, and while lots with key devices generally have large weights, they are not necessarily dominant.

Table 36: Percentage difference between CPLEX and heuristic solutions for Model 5†

Prob. no.	Model 5 objective	Total weight of lots	No. of lots	No. lots with key devices	No. of change-overs	Average mach. time (hr)	Maximum mach. time (hr)	Minimum mach. time (hr)
1	0.01	0	-2.41	-2.60	0	22.72	0.54	70.48
2	-0.14	0.05	-7.14	4.76	0	5.72	-0.21	29.25
3	-0.23	0.12	2.5	2.63	0	2.25	1.35	0
4	0.1	-0.01	0.98	1.02	0	0.13	-0.08	0
5	0	0	0	0	0	-0.04	-0.29	0
6	-1.4	0.52	1.47	1.49	0	0.68	0	0
7	-1	0.28	9.47	9.89	0	0	0.13	0
8	-0.01	0	0	0	0	0	0	0
9	0	0	0	0	0	-0.22	0	0
Average	-0.30	0.11	0.54	1.91	0	3.47	0.16	11.08

† Values reported are calculated as follows: 100% × (CPLEX – heuristic) / CPLEX

The CPLEX results for Model 6 are presented in Table 37 and the percentage differences with the heuristic solutions are given in Table 38. In these runs, the idea of aiming for the target solution is abandoned in favor of the greedy objective of maximizing the weighted sum of the lots processed. As expected, the CPLEX solutions always provide larger objective function values, averaging 21.28% above the heuristic solutions. Also, CPLEX processes more lots in general but fewer key lots. In neither case was preemptive priority given to key lots as it was in Model (5) for the target setting run. Moreover, the heuristic is not trying to achieve the Model (6) objective of maximizing the weighted sum of the lots processed. Instead, it is trying to identify changeovers that come as close as possible to the target solutions.

Although the feasible regions of Models 5 and 6 are identical, CPLEX had much more difficulty finding optimal solutions when the objective was to maximize the weighted sum of lots selected for processing, and was only able to arrive at an optimum

within 1 hour for three of the nine cases. This can be seen in the last two columns of Table 37. There is no theoretical explanation for this result but empirically we have found in the past that it is easier to solve optimization problems with variable targets than when the objective function is more general (e.g., see Zhang and Bard 2006). The use of an absolute value objective function focuses the search for the optimum around the target values in the feasible region thus reducing its effective size. This has a "convexifying" effect on the problem so local optimum are often global optima. To some extent, this can be seen when (6a) is linearized by replacing each term of the form $|x - X|$ with z and adding the constraints $z \geq x - X$ and $z \geq X - x$ to the model.

Table 37: CPLEX changeover results for Model 6

Prob. no.	Total weight of lots (10^4)	No. of lots	No. lots with key devices	No. of change-overs	Average mach. time (hr)	Maximum mach. time (hr)	Minimum mach. time (hr)	Run time (sec)	Solution status
1	25069	83	69	10	20.74	23.86	9.10	43	optimal
2	26277	92	90	10	22.24	23.97	11.26	3664	feasible
3	7403	44	39	10	13.02	23.51	3.89	28	optimal
4	36209	79	76	10	22.26	24.00	11.84	396	optimal
5	36108	89	86	10	22.24	23.99	11.26	3681	feasible
6	13731	71	70	10	20.43	23.98	3.88	507	optimal
7	33493	79	77	10	22.51	24.00	13.84	3777	feasible
8	35995	102	98	10	23.47	24.00	22.67	3786	feasible
9	22199	70	68	10	22.48	24.00	13.84	3763	feasible

Table 38: Percentage difference between CPLEX and heuristic solutions for Model 6†

Prob. no.	Total weight of lots	No. of lots	No. lots with key devices	No. of change-overs	Average mach_time (hr)	Maximum mach_time (hr)	Minimum mach_time (hr)
1	17.19	-2.41	-14.49	0	16.01	0.29	40.99
2	24.87	2.17	25.56	0	15.02	0.42	66.61
3	32.66	11.36	0	0	-18.74	-1.19	76.86
4	14.49	-27.85	-17.11	0	8.49	0.21	57.09
5	13.54	-20.22	-11.63	0	3.1	0.38	51.87
6	31.4	5.63	4.29	0	8.27	0.13	-30.93
7	16.1	-8.86	-18.18	0	-4.35	0.25	-60.48
8	19.53	-9.8	-10.2	0	1.24	0.21	2.69
9	21.72	-11.43	1.47	0	-4.76	0.13	-60.48
Average	21.28	-6.82	-4.48	0	2.70	0.09	16.03

† Values reported are calculated as follows: 100% × (CPLEX – heuristic) / CPLEX

Chapter 9: Future Work

Although significant progress has been made in solving the AT multipass scheduling problem, improvements are still possible. For Scheme I described in Chapter 6, at least two areas of opportunity present themselves. First, the sequential nature in which passes are scheduled gives priority to the current lots in WIP regardless of their step number. If it were possible to take a more expansive view of the processing requirements of a route, this might allow us to schedule all passes of high priority lots at the beginning of the planning horizon. Doing so could decrease shortages of some key devices, but at the expense of increasing shortages of other key devices in the current or next planning horizon. The tradeoffs would have to be carefully weighed. The second area concerns the computational effort. For some instances considered in Section 6.4, 500 iterations for phase I and 100 iterations for phases II and III produced long runtimes. The implication is that larger instances might require more than 100 iterations for phases II and III to guarantee good solutions. Determining the best settings to balance solution quality with runtime is a matter that will be addressed in any upcoming pilot project aimed mainly at fixing data formats and user interface requirements

For Scheme II present in Chapter 7, the computations showed that on average 6,116 CPU seconds using four threads were required to reach an optimality gap of 8.94%. Future work to improve the Scheme II includes the development of a customized algorithm to solve the phase 1 assignment model, the primary bottleneck. We believe that a column generation approach will prove much more effective than CPLEX for this problem. In addition, it would be useful develop a real-time control algorithm for making adjustments to the schedule in light of uncertain events such as machine breakdowns and

newly arriving orders, or when more changeovers are called for in a specific time period than can be performed by the available crew.

Finally, the real-time algorithm detailed in Chapter 8 could be updated to take multiple passes into account.

Appendix A: Procedure to Account for Logpoint and Operation Number in Multipass Scheme I

1. When reading the input files, the logpoint and operation number information are recorded for each lot in the “wip.csv” file and each route in “route.csv” file.
2. When the single-pass algorithm chooses candidate routes for a lot or a machine-tooling-lot assignment (including initial machine tooling and lot), not only the device but also the logpoint and operation number of the lot must be consistent with those of the routes. This leads to smaller sets of candidate routes for a lot or a machine-tooling-lot assignment in the single-pass algorithm than in the original GRASP.
3. When the single-pass algorithm chooses candidate lots for a machine or a machine-tooling setup, it picks candidate routes first and then the candidate lots based on the candidate routes. Not only device but also logpoint and operation number of the candidate lots must be consistent with those of one of the candidate routes for the machine or the machine-tooling setup. This results in smaller sets of candidate lots for a machine or a machine-tooling setup in the single-pass algorithm than in the original GRASP.
4. When the optimal machine-tooling-lot assignment for the first-pass lots is written to the “solution.csv” file, the logpoint and “Pass_num” of a lot is added in the record that represents the lot. Since all the lots are first-pass lots by convention, the "Pass_num" for these lots are just 1.

Appendix B: : Pseudocodes for Subroutines in Multipass Scheme I

B.1 INITIALIZATION

At Step 0 the files "solution.csv" and "route.csv" are read to initialize the necessary data structures and parameters.

`Initialization_for_Multipass_Algorithm`

Input: solution.csv and route.csv

Output: $C\_M\_\Lambda\_T$, M_Λ_T, $L^1(i)$, $tcl(l, logpoint(l))$, $last\_time = 0$, $current\_time = 0$, FL, R and $R(i, \lambda, l, logpoint(l))$

Step 1. Read solution.csv; set $M\_\Lambda\_T = \emptyset$ and $L^1(i) = \emptyset$.

1a. FOR each $i \in M^1$

Identify machine-tooling setup id $\lambda(i)$ and completion time $t_c(i)$ of the last lot assigned to the machine. Add $(i, \lambda(i), t_c(i))$ to M_Λ_T. Indentify the lots $(l, logpoint(l))$ assigned to machine i and add them to $L^1(i)$.

ENDFOR

1b. FOR each $l \in L^1$

Identify the completion time of l and the corresponding logpoint in the solution.csv file, and get $tcl(l, logpoint(l))$

ENDFOR

Step 2. Initialize two variables recording the current time and last time (most recent time) that some machine finished all lots assgined to it; $last\_time = 0$, $current\_time = 0$.

Step 3. Set candidate lot list $CL = \emptyset$ and finished lot list $FL = \emptyset$.

Step 4. Read route.csv file and initialize R and $R(i,\lambda,l,logpoint(l))$ for $i \in M$, $\lambda \in \Lambda(i)$, $l \in L$

Step 5. Initialize $C\_M\_\Lambda\_T$. Add $(i,\lambda(i),t_c(i))$ into the set $C\_M\_\Lambda\_T$ for all $i \in M$.

Steps 2 and 3 can be completed in constant time. The worst case for Step 1 is $O(|M^1|\times|\Lambda|+|L^1|)$, the worst case for Step 4 is $O(|M|\times|\Lambda|\times|L|)$, and the worst case for Step 5 is $O(|M|)$. Thus the complexity of the procedure is $O(|M|\times|\Lambda|\times|L|)$.

B.2 UPDATING CANDIDATE LOT LIST

At the beginning of the planning horizon, all higher-pass lots are unavailable by definition. When a machine finishes its assigned lots, higher-pass lots may become available except for those at the final step in their route since no more processing is required.

`Building_CL_Algorithm`

Input: *last_time, current_time*, *CL*, L^1, *FL*, $tcl(l,logpoint(l))$, $l \in L$

Output: updated *CL*

Step 1. Try to add the lots that are scheduled by the *single-pass algorithm* and finish processing between *last_time* and *current_time* to the candidate lot list *CL.*

FOR $(l, logpoint(l)) \in L^1$

If *last_time* = 0, then

If $last\_time \le tcl(l,logpoint(l)) \le current\_time$, then

run `Check_for_Next-Pass_Algorithm` with input $(l, logpoint(l))$.

Endif

Else

If $last\_time < tcl(l,logpoint(l)) \le current\_time$, then

run `Check_for_Next-Pass_Algorithm` with input (l, *logpoint*(l)).

Endif

Endif

If (l, *next_logpoint*(l)) is returned, then

add (l, *next_logpoint*(l)) to the candidate lot list *CL*.

Endif

ENDFOR

Step 2. Try to add the lots that are scheduled by `Multipass_Algorithm` and finish between *last_time* and *current_time* to the candidate lot list *CL*.

FOR (l, *logpoint*(l)) $\in$ *FL*

If *last_time* < *tcl*(l, *logpoint*(l)) $\leq$ *current_time*, then

run `Checking_for_Next_Pass_Algorithm` to (l, *logpoint*(l)).

If (l, *next_logpoint*(l)) is returned, then

add (l, *next_logpoint*(l)) to the candidate lot list *CL*.

endif

endif

ENDFOR

Step 1 can be executed in $O(|CL| \times |R|)$ time and Step 2 in $O(|FL| \times |R|)$ time. Thus, the complexity of `Building_CL_Algorithm` is $O((|CL| + |FL|) \times |R|)$.

B.3 CHECKING WHETHER NEXT PASS EXISTS

The essence of `Building_CL_Algorithm` is to check whether a scheduled lot is at the end of its route or if additional processing is required. The check is performed by comparing the current logpoint with the last logpoint for the device in the routing table.

`Check_for_Next-Pass_Algorithm`

Input: (l, *logpoint*(l))

Output: (l,*next_lotpoint*(l)) or $\varnothing$

Step 1. Check whether there exists a "next pass" for the lot (l,*logpoint*(l)): Read route.csv file and identify all subroutes $r \in R(i, \lambda, l, logpoint(l))$ for device $d(l)$; rank the subroutes in an ascending order based on *logpoint*(l).

Step 2. If there exists any logpoint greater than *logpoint*(l), then

identify the logpoint closest to *logpoint*(l) and denote it by *next_logpoint*(l); go to Step 3;

else

Return $\varnothing$, stop.

endif

Step 3. Create a new combination of lot id and logpoint (l, *next_logpoint*(l)) and return.

The complexity of the algorithm is $O(|R|)$.

B.4 GREEDY RANDOMIZED PROCEDURE FOR ASSIGNING LOTS

Given the candidate machine list *C_M_Λ_T_2* and the candidate lot list *CL* at Step 4 of the `Multipass Algorithm`, we must decide which lots to assign to which machines. Each machine in *C_M_Λ_T_2* is considered in sequence starting with the first one, say, i_1. Rather than assigning lots to i_1 in decreasing order of their benefit value, though, a greedy randomized procedure is used to diversify the search for good solutions. The main idea is to randomly choose a lot from the top five candidates in the ranked list $CL(i_1, \lambda(i_1)) \subseteq CL$. The process is repeated until no more assignments are possible, at which point, the next machine i_2 is considered.

In the computations, some lots become available earlier than the machine under consideration, and other lots later. Thus we need to keep track of when a lot become

available, i.e., when the step associated with its preceding logpoint is finished. This information is recorded in $(l, next\_logpoint(l))$, $tcl(l, preceding\_ logpoint(l)) \in CL$.

The advantage of this randomized approach over a purely greedy approach is that it expands the search area allowing us to explore more of the feasible region. For example, assume that we have a machine i_1 with 18 hours remaining, and four lots in $CL(i_1, \lambda(i_1))$ with benefit values ordered such that $l_1 = l_2 = 100 > l_3 = l_4 = 99$. The times required for processing l_1, l_2, l_3 and l_4 are 8, 8, 5 and 5 hours, respectively. The greedy solution would assign l_1, l_2 to i_1. A better solution would be to assign l_1, l_3, and l_4 to i_1. This procedure is outlined below.

`Assign_Lot_Algorithm`

Input: $(i, \lambda(i), t_c(i))$, CL, FL, R, $tcl(l, logpoint(l))$

Output: Updated CL, FL, $L^2(i)$, $tcl(l, logpoint(l))$

Step 1. Rank the lots in CL in a descending order based on their benefit contribution to (1a).

Step 2. Build $CL(i, \lambda(i))$, the candidate lot list that can be processed by machine i with tooling setup $\lambda(i)$.

FOR each $(l, logpoint(l)) \in CL$, beginning with the first element,

If $R(i, \lambda, l, logpoint(l)) \neq \emptyset$, then

find the subroute $r \in R(i, \lambda, l, logpoint(l))$ with the largest PPH;

calculate time required to finish the lot $(l, logpoint(l))$ as follows:

$required\_time$ = Quantity/ PPH + $Load\_Unload\_Time$

If $required\_time \leq H(i) - t_c(i)$, then

add $(l, logpoint(l))$ to $CL(i, \lambda(i))$

Endif

Endif

ENDFOR

Step 3. Rank the lots in $CL(i,\lambda(i))$ in non-increasing order based on their benefit value

Step 4. WHILE ($|CL(i,\lambda(i))| \geq 1$)

Let $num\_candidate = \min\{|CL(i,\lambda(i))|, 5\}$, then

randomly choose a lot (l, $logpoint(l)$) from top $num\_candidate$ in the $CL(i,\lambda(i))$;

Calculate time required to finish lot (l, $logpoint(l)$) as follows:

$required\_time$ = Quantity/ $PPH + Load\_Unload\_Time$.

Let $start\_time = \max\{t_c\_temp, tcl(l, preceding\_logpoint(l))\}$

If $required\_time \leq H(i) - start\_time$, then

put $t_c(i) \leftarrow start\_time + required\_time$;

add (l, $logpoint(l)$) to FL;

add (l, $logpoint(l)$) to $L^2(i)$;

put $t_c(i) \leftarrow t_c(i) + required\_time$;

set $tcl(l, logpoint(l)) = t_c(i)$;

Endif

Delete (l, $logpoint(l)$) from CL;

ENDWHILE

Step 1 has complexity $O(|CL| \times \log(|CL|))$. In the worst case, Step 2 takes $O(|CL| \times |R|)$ time, Step3 takes $O(|CL| \times \log(|CL|)$ time, and Step 4 takes $O(|CL|)$. Thus, complexity of the algorithm is $O(|CL| \times \log(|CL|) + |CL| \times |R|)$.

Appendix C: Pseudocodes for Changeover Algorithm Subroutines

C.1 Initialization

Four sets: *available_tooling_set*, *candidate_lot_list*, *candi_mach_for _toolings_list*, and *candi_mach_for _changeover_list* need to be initialized. The results give the status of machine setups and lot assignments at the beginning of the planning horizon. Note the *candidate_lot_list* associated with the *changeover algorithm* is larger than *CL* associated with the *multipass algorithm* since unassigned first-pass lots are now taken into consideration.

Sets and indices

candidate_lot_list	set of combinations of lot id and logpoint – this set stores available lots that can be assigned to reset machines; $(l, logpoint(l)) \in candidate\_lot\_list$
candi_mach_for _toolings_list	set of combinations of machine instance id, tooling setup, and completion time of the last lot finised on this machine. This set stores the machines that still have toolings on them; $(i, \lambda(i), t_c(i)) \in candi\_mach\_for\_toolings\_list$
candi_mach_for _changeover_list	set of combinations of machine instance id and completion time of the last lot finished on this machine – stores the machines that are eligible to be reset; $(i, t_c(i)) \in candi\_mach\_for\_changeover\_list$
available_tooling_set	set of combinations of tooling family *tf* and number of tooling pieces n_{tf} available from the tooling family *tf*; $(tf, n_{tf}) \in available\_tooling\_set$

$L^3(i)$ set of combinations of lot id and logpoint that are assigned to machine *i* during the *changeover algorithm* (excludes lots assigned by the *single-pass* or *multipass algorithms*); $(l, logpoint(l)) \in L^3(i)$.

Initialization

Step1. Set *available_tooling_set* = Ø. For each tooling family *tf*, perform

1a. Read tooling.csv, and identify *total number* of tooling pieces for tooling family *tf*.

1b. Read initialsetup.csv, and identify *number of used* of tooling pieces for tooling family *tf*.

n_{tf} = *total number* − *number of used*

1c. Read solution.csv, and identify the *number of machine* that setup with tooling family *tf* and their initial setup were not tooling family *tf*. Then the number of tooling pieces available for *tf* is updated as:

n_{tf} = *total number* − *number of machine*

1d. Add (tf, n_{tf}) to *available_tooling_set*

Step 2. Set *candidate_lot_list* = Ø.

2a. Read wip.csv and identify all lots and their logpoints, and add $(l, logpoint(l))$ to the set *candidate_lot_list*. Mark all lots in wip.csv as first-pass lots.

2b. Read solution.csv and identify all assigned lots and their logpoints, and delete $(l, logpoint(l))$ from the set *candidate_lot_list*. Mark all lots in solution.csv as first-pass lots.

Step3. Set *candi_mach_for_toolings_list* = Ø, and

$L^3(i) = Ø$ for all $i \in M$.

3a. Read solutions.csv to identify the machines that are setup or that have initial setup. Add these machines to *candi_mach_for _toolings_list*

3b. Read machines.csv and solutions.csv to identify the machines that are idle at the beginning of the time horizon. These machines are those don't have any lots to processs during time horizon, and don't have initial lots to process or have finished initial lots before the start of the time horizon. Add these machines to *candi_mach_for _resetup_list*

C.2 Updating candidate_lot_list

When a machine finishes all of the lots assigned to it, we need to update the set *candidate_lot_list*. The corresponding procedure is similar to the one used to update *CL* in `Multipass_Algorithm`, but whenever we find a (*l*, *next_logpoint*(*l*)) by running `Checking_for_Next_Pass_Algorithm`, we need to check whether this lot has already been assigned to a machine by `Multipass_Algorithm` for subsequent processing. Since this information is stored in *FL*, we simply check whether (*l*, *next_logpoint*(*l*)) is in *FL*.

`Update_Candidate_Lot_List_Algorithm`

Input: $(i, \lambda(i), t_c(i))$, $L^1(i)$, $L^2(i)$, $L^3(i)$, *FL*, *candidate_lot_list*,

Output: Updated *candidate_lot_list*

Step 1: FOR each lot $(l, logpoint(l)) \in L^1(i) \cup L^2(i) \cup L^3(i)$, apply

`Check_for_Next_Pass_Algorithm` to (*l*, *logpoint*(*l*)).

If (*l*, *next_logpoint*(*l*)) is returned, then

check whether (*l*, *next_logpoint*(*l*)) is in *FL*.

If not, then add (*l*, *next_logpoint*(*l*)) to *candidate_lot_list*

Endif

ENDFOR

C.3 Changeover a machine

When a machine is empty, i.e., no tooling is installed on it, this machine is eligible to be reset. Whenever a new tooling piece becomes available or a lot finishes a step in its route, we call the `Changeover_a_Machine_Algorithm` to check whether the machine can be reset at the current time. If so, then we will apply the `Assign_Lot_Algorithm` discussed in Appendix B.4 to assign lots.

`Changeover_a_Machine_Algorithm`

Input: $(i, t_c(i))$, $\Lambda(i)$, *candidate_lot_list*, *available_tooling_set*, *R*, *Load_Unload_Time*

Step 1. Rank the lots in *candidate_lot_list* by the benefit value of l corresponding to (l, *next_logpoint*(l)), *tcl*(l, *preceding_ logpoint*(l)). Let *best_benefit* = 0, *candidate_lot_list_best = candidate_lot_list*, *FL_best = FL*, $L^3(i)$*_best* = $L^3(i)$, *tcl*(l, *logpoint*(l))*_best = tcl*(l, *logpoint*(l)).

FOR each $\lambda \in \Lambda(i)$,

Step 2. Check *available_tooling_set* to see whether there are enough tooling pieces for λ.

If yes, then go to Step 3.

Step 3. Let *candidate_lot_list_temp = candidate_lot_list*, *FL_temp = FL*, *R_temp = R*, *tcl*(l, *logpoint*(l))*_temp* = *tcl*(l, *logpoint*(l)). Apply `Assign_Lot_Algorithm` with $(i, \lambda, t_c(i))$, *candidate_lot_list_temp*, *FL_temp*, *R_temp*, *tcl*(l, *logpoint*(l))*_temp* as inputs. Get updated *candidate_lot_list_temp*, *FL_temp*, $L^3(i)$*_temp*, *tcl*(l, *logpoint*(l))*_temp*.

Calculate the contribution to (1a) based on the updated *FL_temp*, $L^3(i)$*_temp*, and denote the contribution as *benefit_temp*. If *benefit_temp* > *best_benefit*, *candidate_lot_list_best = candidate_lot_list_temp*, *FL_best = FL_temp*, $L^3(i)$*_best* = $L^3(i)$*_temp*, *tcl*(l, *logpoint*(l))*_best = tcl*(l, *logpoint*(l))*_temp*.

ENDFOR

Step 4. $candidate\_lot\_list = candidate\_lot\_list\_best$, $FL = FL\_best$, $L^3(i) = L^3(i)\_best$, $tcl(l, logpoint(l)) = tcl(l, logpoint(l))\_best$

Appendix D: Complexity of the Assignment Model and Sequencing Model

Both the assignment model and the sequencing model are large-scale MIPs, but only the former turned out to be difficult to solve to optimality. In this section, we explore the computational complexity of the two models and show that they are both strongly $\mathcal{NP}$-hard. In addition, we investigate various versions of the sequencing model to gain insight into its quick convergence. The analysis indicates that simple cases are either polynomial or pseudopolynomial solvable suggesting that some instances of the full problem may still be easy to solve. Our results are based on analogies to either structured integer programs or standard machine scheduling problems whose complexities are well established, so no formal proofs will be given. In most cases, we will cite results from either Garey and Johnson (1979) or Brucker and Knust's website (http://www.informatik.uni-osnabrueck.de/knust/class/). For example, the single machine scheduling problem (SMSP) with arbitrary setup times between jobs is equivalent to the traveling salesman problem, which is strongly $\mathcal{NP}$-hard. Therefore, so is the SMSP.

Machined-tooling-lot assignment problem. Although model (3) has a number of complicating constraints which are included to keep track of resources, time, and machine capacity, it is only necessary to consider a simplified version to see its complexity.

Proposition 1. The problem represented by model (3) is $\mathcal{NP}$-hard in the strong sense.

Proof. We show that the generalized assignment problem (GAP), which is known to be strongly $\mathcal{NP}$-hard, can be polynomially transformed into the machine-tooling-lot assignment problem (MTLAP). For n jobs and m machines each with capacity c_i, $i = 1,\ldots,m$, recall that the GAP is given by

$$\text{Maximize}\left\{\sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}x_{ij} : \sum_{j=1}^{n} a_{ij}x_{ij} \le c_i, i = 1,...,m;\ \sum_{i=1}^{m} x_{ij} \le 1, j = 1,...,n;\ x_{ij} \in \{0,1\}, \forall i, j\right\}$$

where w_{ij} is the benefit when job j is executed by machine i, a_{ij} is the capacity of machine i required to perform job j, and $x_{ij} = 1$ if job j is a assigned to machine i, 0 otherwise. Now, given an instance of GAP an instance of MTLAP can be created as follows. Assume that the objective is to maximize the weighted sum of lots processed, each machine is already set up with tooling, and that there is only a single route for each lot on each machine (that is, subroutes don't exist so the subscribe r is not needed). For all machines i, set $c_i = H_i$, and let a_{il} be as in the GAP (the capacity required to process lot l on machine i will depend on the tooling assigned it). Because the transformation from GAP to MTLAP is one-to-one, the result follows immediately. ■

Sequencing problems. In this subsection, we make use of the standard three-field machine scheduling notation: $\alpha|\beta|\gamma$, where α indentifies the machine environment and contains a single entry, β describes processing characteristics and places restrictions on the jobs, and may be empty, and γ is the objective to be minimized and usually contains a single entry (see Pinedo 2012). For our purposes, the following parameters are relevant.

α : single machine (1); identical machines in parallel (*Pm*); unrelated machines in parallel (*Rm*)

β : ready time of lot l (r_l); deadline of lot l (d_l); processing time of lot l (p_l); precedence constraints (*prec*); sequence in which lots must be processed (*chains*), common due date (D); lot l must be processed on one of a subset of machines (M_l)

γ : makespan ($C_{\max}$); completion time on machine i (C_i); binary indicator that lot l is late (U_l); tardiness of lot l (T_l); weight of lot l (w_l)

Model (4) is a parallel machine scheduling problem with three objectives: (i) minimize the weighted sum of unprocessed key lots; (ii) maximize the weighted sum of processed lots; and (iii), minimize the sum of completion times over all machines, subject to precedence constraints, deadlines and lot-machine assignments. If we consider the first objective, then the problem can be represented by $Rm \mid prec,\ d_l,\ M_l \mid \Sigma_l\, w_l U_l$. The inclusion of M_l in field β accounts for the fact that each lot l must be processed on a specific machine as determined by the solution to model (3). Although this problem does not appear to have been previously studied, several simplifications lead us to the following result.

Proposition 2. The problem represented by model (4) is $\mathcal{NP}$-hard in the strong sense.

Proof. We consider the case of a single machine with ready times and the objective of minimizing the weighted sum of unprocessed lots, $1 \mid r_l \mid \Sigma_l\, w_l U_l$. This problem is $\mathcal{NP}$-hard in the strong sense (Pinedo 2012) and can be obtained by fixing the lots on all machines but one and assuming that the lots to be sequenced on the remaining machine have only one pass to complete. Accordingly, the second- and higher-pass lots among them will have ready times (and deadlines). By the restriction principle (Garey and Johnson 1979) then, we can conclude that the problem represented by model (4) has the same complexity. ■

Interestingly, the result also holds for the second objective function (because it is really equivalent to the first), the third objective of minimizing the sum of completion times on a single machine, $1 \mid r_l \mid \Sigma_l\, C_l$, as well as when all the lots have unit processing time, chains are present, and the objective is to minimize the unweighted sum of unprocessed lots; that is, $1 \mid chains,\ p_l = 1 \mid \Sigma_l\, U_l$. Note that chains correspond to two or more of the same lot on the same machine but with different pass numbers. When there are no restrictions on the lots, however, and the objective is to minimize the weighted

sum of unprocessed lots, the problem, $1||\Sigma_j w_j U_j$, is $\mathcal{NP}$-hard in the ordinary sense. Finally, for our purposes, when the lots have unit processing time but there is a ready time for each, the problem of minimizing the weighted sum of unprocessed lots on a single machine, $1|\ r_j, p_j = 1|\Sigma_j w_j U_j$, becomes even easier and is polynomially solvable. In general, all SMSP with ready times and due dates are strongly $\mathcal{NP}$-hard (Garey and Johnson 1979). As mentioned, such a problem might arise if all the lot start times were fixed on all but one machine. The lots assigned to the remaining machine would have ready times and deadlines that could be derived from schedules of the same lots undergoing different passes in their route on the fixed machines. Of course, if all the lots on a machine are unique without predecessor or successor passes on other machines giving $1||\Sigma_j U_j$, any sequence is optimal in isolation as long as the pass order is preserved.

Appendix E – Mathematical Programming Models for Real-Time Decision Making

Our first model tries to determine new setups and lot assignments that are as close as possible to those of the target solution; the second is aimed at maximizing the weighted sum of lots selected for processing. Both models are derived from (1a) – (1i) and use the same notation. Several new symbols are defined below.

Sets

$L(i)$ set of available lots that can be processed on machine i

$L(i,\lambda)$ set of available lots that can be processed on machine i with tooling setup λ

$\mathcal{M}$ set of all machines

$\mathcal{M}(l)$ set of all machines that can process lot l

$\mathcal{M}(l,r)$ set of all machines that can process lot l using route r

M set of machines that are eligible for changeover (each machine falls into a machine group); $M \subseteq \mathcal{M}$

$M(l)$ set of machines that can process lot l

$M(l,r)$ set of machines that can process lot l using route r

$R(l)$ set of routes that can process lot l

$R(i,l)$ set of routes that use machine i to process lot l

$R(i,l,\lambda)$ set of routes that use machine i to process lot l with tooling setup λ

$\Lambda(i)$ set of tooling setups that can be installed on machine i

$\Lambda(i,t)$ set of tooling setups that can be installed on machine i that use a tooling piece from family t

Parameters and data

n_t^{tooling} number of tooling pieces from family t that are available or will become available in the next Δ^{free} hours

n^{setups} maximum number of changeovers permitted

τ_l^{load} load the unload time for each lot l

$X_{ilr}, Y_{i\lambda}$ values of the target solutions for x_{ilr} and $y_{i\lambda}$, respectively;

Model 5

$$\phi^1 = \text{Minimize} \quad \sum_{l\in L}\sum_{r\in R(l)} w_{lr}\left|\sum_{i\in M(l,r)\subseteq\mathcal{M}(l,\lambda)} x_{ilr} - \sum_{i\in\mathcal{M}(l,\lambda)} X_{ilr}\right| \tag{5a}$$

$$\text{subject to} \quad \sum_{i\in M(l)}\sum_{r\in R(i,l)} x_{ilr} \le 1, \quad \forall\ l\in L \tag{5b}$$

$$\sum_{\lambda\in\Lambda(i)} y_{i\lambda} \le 1, \ \forall\ i\in M \tag{5c}$$

$$\sum_{l\in L(i,\lambda)}\sum_{r\in R(i,l,\lambda)} x_{ilr} \le \left|L(i,\lambda)\right| y_{i\lambda}, \quad \forall\ i\in M, \lambda\in\Lambda(i) \tag{5d}$$

$$\sum_{i\in M}\sum_{\lambda\in\Lambda(i,t)} b_{\lambda t} y_{i\lambda} \le n_t^{tooling}, \quad \forall\ t\in T \tag{5e}$$

$$\sum_{i\in M}\sum_{\lambda\in\Lambda(i)} y_{i\lambda} \le n^{setups} \tag{5f}$$

$$\sum_{l\in L(i,\lambda)}\sum_{r\in R(i,l,\lambda)}\left(\frac{n_l^{devises}}{\rho_{ilr}} + \tau_l^{load}\right) x_{ilr} \le H_i y_{i\lambda}, \ \forall\ i\in M, \lambda\in\Lambda(i), \tag{5g}$$

$$x_{ilr}\in\{0,1\}, \ \forall\ i\in M, \quad l\in L(i), \ r\in R(i,l) \tag{5h}$$

$$y_{i\lambda}\in\{0,1\}, \quad \forall\ i\in M, \lambda\in\Lambda(i) \tag{5i}$$

The objective in (5a) is to minimize the weighted sum of the absolute deviations of the assignment variables x_{ilr} from the target solution X_{ilr} (here, $w_{lr} = w_l - \varepsilon_r$). This is an indirect way of guiding the setup variables $y_{i\lambda}$ to the target solution $Y_{i\lambda}$. It is necessary to sum x_{ilr} and X_{ilr} over all $i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)$ to take into account the fact that different machines in the same family are essentially identical. If route r is used to process lot l on machine m_1 in the maximum capacity solution, then the solution to model 5 will meet the target for lot l when it is processed on any machine in the same family as m_1 which is set up in accordance with route r. As required by constraints (5b), the equivalent of

(1b), each lot can be processed by at most one route on one machine so $\sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} x_{ilr}$ (and $\sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} X_{ilr}$) is either 0 or 1 in (5a). When lot l is assigned to machine $i \in M(l)$, then the tooling associated with one of the routes $r \in R(i,l)$ must be installed on that machine and the temperature set accordingly.

Constraints (5c) are the same as (1c) and restrict each machine i to at most one tooling configuration from the set $\Lambda(i)$. Constraints (5d) allow up to $|L(i,\lambda)|$ lots to be processed on machine i but prevent a particular lot from being processed on that machine unless an appropriate setup $\lambda \in \Lambda(i)$ is made. In conjunction with (5b), no more than one route $r \in R(i,l,\lambda)$ can be selected for lot l.

Constraints (5e) are a simplified version of (1d) where it is now assumed that the temperature is implicitly accounted for in the definition of λ. They restrict the total number of tooling pieces assigned to machines from family t to the number of pieces available as specified by the parameter n_t^{tooling}. Constraint (5f) in conjunction with (5c) limit the number of changeovers to at most n^{setups}. Constraints (5g) ensure that the total amount of time for processing lots on each machine is less than or equal to the amount of time available. Logical restrictions are placed on the variables in (5h) – (5i).

Nevertheless, when solving Model 5, should $\phi^1 = 0$ at optimality in (5a), then $\sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} y_{i\lambda} = \sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} Y_{i\lambda}$ for all $\lambda \in \Lambda$. A formal statement of this observation follows.

Proposition 1. If the optimal solution to Model 5 yields $\phi^1 = 0$, then there exists an optimal solution with $\sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} y_{i\lambda} = \sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} Y_{i\lambda}$ for all $\lambda \in \Lambda$.

Proof. Although a route for a particular device, (and hence a lot) is not unique, each route maps into a particular machine family, tooling family, and temperature. Therefore, if $\sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} x_{ilr} = \sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} X_{ilr} = 1$, implying that the corresponding term in (5a) is zero, then there must exist a machine $i \in M(l,r)$ with r fixed, and a machine $i' \in$

$M(l, r)$ with r fixed (i' may be or not equal to i), must have a unique setup λ such that $y_{i\lambda} = Y_{i'\lambda} = 1$. Otherwise, it would not possible to process lot l on machine i using route r. Alternatively, if $\sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} x_{ilr} = \sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} X_{ilr} = 0$, which implies that $\sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} Y_{i\lambda} = 0$, then we can safely set $\sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} y_{i\lambda} = 0$ without adversely affecting the solution to Model 5. That is, there are no other lots $\bar{l} \in L(i)$ that could be processed on machine i using routing r such that $\sum_{i \in M(\bar{l},r) \subseteq \mathcal{M}(\bar{l},\lambda)} x_{i\bar{l}r} = \sum_{i \in M(\bar{l},r) \subseteq \mathcal{M}(\bar{l},\lambda)} X_{i\bar{l}r} = 1$ because this would require $\sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} y_{i\lambda} = \sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} Y_{i\lambda} = 1$. ■

A second alternative to the objective function in (6a) is to minimize the weighted sum of the two absolute value terms, $\left| \sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} x_{ilr} - \sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} X_{ilr} \right|$ and $\left| \sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} y_{i\lambda} - \sum_{i \in M(\lambda) \subseteq \mathcal{M}(l,\lambda)} Y_{i\lambda} \right|$, but this presents the problem of choosing the appropriate weights. Regardless of objective function, though, the individual absolute value terms in (6a) can be linearized without introducing any additional variables or constraints by recognizing that $\sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} X_{ilr}$ is binary for all i, l, r. When the latter term is 0, we replace the corresponding absolute value term with $\sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} x_{ilr}$; when it is 1, we replace the absolute value term with $1 - \sum_{i \in M(l,r) \subseteq \mathcal{M}(l,\lambda)} x_{ilr}$.

Model 6

The above model focuses on the target setups and assignments indicated in the solution of (1a) – (1i) and derives a new solution for the x and y variables. This solution only considers the resources available at the current time, call it τ, along with those that will become available in the upcoming interval $\tau + \Delta^{\text{free}}$. The second model takes an opportunistic approach and tries to achieve the greatest benefit in the upcoming round of changeovers.

$$\phi^2 = \text{Maximize} \sum_{i \in M} \sum_{l \in L(i)} \sum_{r \in R(i,l)} w_{lr} x_{ilr} \quad (6a)$$

$$\text{subject to (5b) – (5i)} \quad (6b)$$

The objective function in (6a) corresponds to the weighted sum of the lots selected for processing. Nevertheless, the solution to (6a) – (6b) may be nearsighted because it doesn't consider subsequent setups and resource requirements; it ignores the target solution which is optimal for the full planning horizon not just for the interval τ to $\tau + \Delta^{\text{free}}$. When τ is close to 0, the objective function value ϕ^2 is likely to be larger than the equivalent value provided by the heuristic but as τ approaches the end of the planning horizon, the opposite is likely to be true. In either case, however, we have the following theoretical result.

Proposition 2. The machine setup and lot scheduling (MSLS) problem represented by either Model 5 or Model 6 is NP-complete in the strong sense.

Proof. We begin with the 3-dimensional matching (3DM) problem, which Garey and Johnson (1979) indicate is NP-complete in the strong sense, and reduce it to an instance of MSLS.

3DM instance: Set $\Theta \subseteq W \times X \times Y$, where W, X, Y are disjoint sets have the same number q of elements.

Question: Does Θ contain a matching; i.e., is there a subset $\Theta' \subseteq \Theta$ such that $|\Theta'| = q$ and no two elements of Θ' agree in any coordinate (no two elements of Θ' share the same components)

Given an instance of 3DM, we create an instance of MSLS by associating the set of machines M with W, the set of lots L with X, and the set of tooling families T with Y. For this instance, we assume that each tooling family has only a single tooling piece, all

temperatures requirements are identical, and that each of the sets M, L and T has exactly q elements; that is, $|M| = |T| = |L| = q$. We also assume that each machine can process at most one lot and that $w_{lr} = 1$ for all $(i,l,t) \in M \times L \times T$. The one-to-one association of (i,l,r) with (i,l,t) is possible because each route corresponds to a unique machine-tooling combination.

Given this instance of MSLS, the equivalent question is whether there is a feasible machine-tool assignment such that the total payoff is no less than q? A "yes" answer to this question means that we can find a solution to MSLS such that q or more lots can be processed. By implication then, solving the restricted version of MSLS implies that we can solve 3DM and vice versa. To conclude, we observe that the transformation from 3DM to MSLS can be done in linear time and that any proposed solution to MSLS can be checked in polynomial time. ■

References

Bard, J.F., Deng, Y., Chacon, R. and Stuber, J. (2010). Midterm planning to minimize deviations from daily target outputs in semiconductor manufacturing. *IEEE transactions on semiconductor manufacturing,* 23(3), 456-467.

Bard, J.F., Gao, Z., Chacon, R. and Stuber, J. (2013). Daily scheduling of multi-pass lots at assembly & test facilities. *International Journal of Production Research*, 51(23-24).

Bard, J.F. and S. Rojanasoonthon (2006). A branch & price algorithm for parallel machine scheduling with time windows and job priorities. *Naval Research Logistics*, 53(1), 24-44.

Bard, J.F. and Wan, L. (2006). The task assignment problem for unrestricted movement between workstation groups. *Journal of Scheduling*, 9(4), 315-341.

Chen, J. and Chen, F.F. (2008). Adaptive scheduling and tool flow control in flexible job shops, *International Journal of Production Research*, 46(15), 4035-4059.

Chen, J.S., Pan, J.C.-H. and Lin, C.M. (2008). A hybrid genetic algorithm for the re-entrant flow-shop scheduling problem. *Expert Systems with Applications*, 34(1), 570-577.

Chen, J.S., Pan, J. C.-H., and Wu, C.K. (2008). Hybrid tabu search for re-entrant permutation flow-shop scheduling problem. *Expert Systems with Applications*, 34 (3), 1924-1930.

Chiang, D.M., Guo, R.S. and Pai, F.Y. (2008). Improved customer satisfaction with a hybrid dispatching rule in semiconductor back-end factories, *International Journal of Production Research*, 46(17), 4903-4923.

Chiang, T.C., Shen, Y.S. and Fu, L.C. (2008). A new paradigm for rule-based scheduling in the wafer probe centre. *International Journal of Production Research*, 46(15), 4111-4133.

Chung, S.H., Tai, Y.T. and Pearn, W.L. (2009). Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes. *International Journal of Production Research*, 47(18), 5109-5128.

Dabbas, R.M. and Fowler, J.W. (2003). A new scheduling approach using combined dispatching criteria in wafer fabs. *IEEE Transactions on semiconductor manufacturing,* 16(3), 501-510.

Demirkol, E., Uzsoy, R. and Ovacik, I.M. (1995). Decomposition algorithms for scheduling semiconductor testing facilities. *IEEE/CPMT International Electronics Manufacturing Technology Symposium*, 199-204.

Demirkol, E. and Uzsoy, R. (1997). Performance of decomposition methods for complex workshops under multiple criteria. *Computers & Industrial Engineering*, 33(1-2), 261-264.

Demirkol, E. and R. Uzsoy (2000). Decomposition methods for reentrant flow shops with sequence-dependent setup times. *Journal of Scheduling*, 3(3), 155-177.

Deng, Y., Bard, J.F., Chacon, R. and Stuber, J. (2010). Scheduling back-end operations in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 23 (2), 210-220.

Denton, B., Viapiano, J. and Vogl, A. (2007). Optimization of surgery sequencing and scheduling decisions under uncertainty. *Health Care Manage Science*, 10, 13-24.

Dolgui, A. and Proth J.M. (2010). *Supply Chain Engineering: Useful Methods and Techniques*. Springer, Heidelberg.

Feo, T.A., Venkatraman, K. and Bard, J.F. (1991). A GRASP for a difficult single machine scheduling problem. *Computers & Operations Research*, 18(8), 635-643.

Fischetti, M. and Lodi, A. (2003). Local branching. *Mathematical Programming*, 98(1), 23-47.

Fowler, J.W., Phillips, D. T. and Hogg, G. L. (1992). Real time control of multiproduct bulk-service semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 5, 158–163.

Freed, T., Doerr, K.H. and Chang, T. (2006). In-house development of scheduling decision support systems: case study for scheduling semiconductor device test operations. *International Journal of Production Research*, 45(21), 5075-5093.

Garey, M.R., Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York

Graves, S.C., Meal, H.C., Stefek, D. and Zeghmi, A.H. (1983). Scheduling of re-entrant flow shops. *Journal of Operations Management*, 3, 197-207

Gupta, A.M. and Sivakumar, A.I. (2006). Job shop scheduling techniques in semiconductor manufacturing. *The International Journal of Advanced Manufacturing Technology*, 27, 1163-1169

Hwang, H. and Sun, J.U. (1997). Production sequencing problem with reentrant work flows and sequence dependent setup times. *Computers & Industrial Engineering*, 33(3-4), 773-776

Jarrah, A.I.Z., Bard, J.F. and deSilva, A.H. (1994). Equipment selection and machine scheduling in general mail facilities. *Management Science*, 40(8), 1049-1068

Jia, J. and Mason, S.J. (2009). Semiconductor manufacturing scheduling of jobs containing multiple orders on identical parallel machines. *International Journal of Production Research*, 47(10), 2565-2585.

Kang, Y.H. and Shin, H.J. (2010). An adaptive scheduling algorithm for a parallel machine problem with rework processes. *International Journal of Production Research*, 48(1), 95-115.

Kim, Y., Bang, J.Y., An, K.Y., and Lim, S.K. (2008). A due-date-based algorithm for lot-order assignment in a semiconductor wafer fabrication facility. *IEEE Transactions on Semiconductor Manufacturing*, 21(2), 209-216.

Knutson, K., Kempf, K., Fowler, J.W. and Carlyle, M. (1999). Lot-to-order matching for a semiconductor assembly and test facility. *IIE Transactions on Scheduling & Logistics*, 31(11), 1103-1111.

Kubiak, W., Lou, S. X. C. and Wang, Y. (1996). Mean flow time minimization in reentrant job shops with a hub. *Operations Research*, 44(5), 764-776.

Leachman, R.C.(2002). Application of mathematical optimization to semiconductor production planning, in M. Resende, P. Pardolos (eds.), *Handbook of Applied Optimization*. Oxford University Press, New York, 746-762.

Leachman, R.C. and Carmon, T. (Freed) (1992). "On capacity modeling for production planning with alternative machine types," *IIE Transactions on Scheduling & Logistics*, 24(4), 62-72.

Leachman, R.C., Jeenyoung K. and Lin V. (2002). SLIM: short cycle time and low inventory in manufacturing at Samsung electronics. *Interfaces*, 32(1), 61-77

Lee, H.Y. and Lee, T.E. (2006). Scheduling single-armed cluster tools with reentrant wafer flows. *IEEE Transactions on Semiconductor Manufacturing*, 19(2), 226-240.

Lee, Y.H. and Kim T. (2002). Manufacturing cycle time reduction using balance control in the semi-conductor fabrication line, *Production Planning & Control*, 13(6), 529-540.

Li, S., Tang, T. and Collins D.W. (1996). Minimum inventory variability scheduler with applications in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 9, 1–5.

Lin, D. and Lee, C.K.M. (2011). A review of the research methodology for the re-entrant scheduling problem. *International Journal of Production Research*, 49(8), 2221−2242.

Liu, H., Zabinsky, Z.B. and Kohn, W. (2011). Rule-based forecasting and production control systems design utilizing a feedback control structure. *IIE Transactions on Operations Engineering & Analysis*, 43(2), 143-152.

Mazzola J.B. and Neebe A.W. (1986). Resource-constrained assignment scheduling. *Operations Research*, 34(4), 560-572.

Mönch L., Fowler J.W., Dauzère-Pérès S., Mason S.J. and Rose O. (2011). A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. *Journal of Scheduling*, 14(6), 583−599.

Monkman, S.K, Morrice, D.J. and Bard, J.F. (2008). A production scheduling heuristic for an electronics manufacturer with sequence dependent setup costs. *European Journal of Operational Research*, 187(3), 1100-1114.

Or, I. (1976). Traveling salesmen-type combinatorial problems and their relation to the logistics of blood banking. Ph.D. thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

Quadt, D. and Kuhn,H. (2009). Capacitated lot-sizing and scheduling with parallel machines, back-orders and setup carry-over, *Naval Research Logistics*, 56(4), 366-384.

Ovacik, I.M. and Uzsoy, R. (1994). Rolling horizon algorithms for a single machine dynamic scheduling problem with sequence dependent setup times, *International Journal of Production Research*, 32, 1243−1263.

Ovacik, I.M. and Uzsoy, R. (1995). Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup times. *International Journal of Production Research*, 33(11), 3173-3192.

Ovacik, I.M. and Uzsoy, R. (1997). *Decomposition methods for complex factory scheduling problems*. Kluwer Academic, Boston

Panwalkar, S. S. and Iskander, W. (1977). A Survey of Scheduling Rules. *Operations Research*, 25(1), 45-61.

Pearn, W.L., Chung, S.H., Chen, A.Y. and Yang, M.H. (2004). A case study on the multistage IC final testing scheduling problem with reentry. International Journal of Production Economics, 88(3), 257–267.

Pfund, M., Fowler, J.W., Gadkari, A. and Chen, Y. (2008). Scheduling jobs on parallel machines with setup times and ready times. *Computers & Industrial Engineering*, 54(4), 764-782.

Pinedo, M.L. (2012). *Scheduling: Theory, Algorithms, and Systems*, Fourth Edition. Springer, New York

Pinto, J.M. and Grossmann, I.E. (1998). Assignment and sequencing models for the scheduling of process systems. *Annals of Operations Research*, 81,433– 466.

Rojanasoonthon, S. and Bard, J.F. (2005). A GRASP for parallel machine scheduling with time windows. *INFORMS Journal on Computing*, 17(1), 32–51.

Song, Y., Zhang, M.T., Yi, J., Zhang, L. and Zheng, L. (2007). Bottleneck station scheduling in semiconductor assembly and test manufacturing using ant colony

optimization, *IEEE Transactions on Automation Science and Engineering*, 4(4), 569-578.

Tu, Y.M., Chao, Y.H., Chang, S.H. and You, H.C. (2005). Model to determine the backup capacity of a wafer foundry, *International Journal of Production Research*, 43(2), 339–359.

Uzsoy, R., Lee, C.Y. and Martin-Vega, L.A. (1992). A review of production planning and scheduling models in the semiconductor industry part I: system characteristics, performance evaluation and production planning," *IIE Transaction on Scheduling & Logistics*, 24(4), 47-60.

Van, Z.P. (2000). Microchip Fabrication: A Practical Guide to Semiconductor Processing, 4th Edition, McGraw-Hill, NY.

Wein, L.M. (1988). Scheduling semiconductor wafer fabrication, *IEEE Transactions on Semiconductor Manufacturing*, 1(3), 115–130.

Zhang, X. and Bard, J.F. (2006). Comparative Approaches to Equipment scheduling in high volume factories. *Computers & Operations Research* 33(1), 132-157.

Zhang, M.T., Niu, S., Deng, S., Zhang, Z., Li, Q. and Zheng, L. (2007). Hierarchical capacity planning with reconfigurable kits in global semiconductor assembly and test manufacturing, *IEEE Transactions on Automation Science and Engineering*, 4(4), 543-552.