The Dissertation Committee for Shihui Jia Certifies that this is the approved version of the following dissertation:

Simulation and optimization techniques applied in semiconductor Assembly and Test operations

Committee:

Jonathan F. Bard, Supervisor

Douglas Morrice

John Hasenbein

Aida Khajavirad

Zhufeng Gao

Simulation and optimization techniques applied in semiconductor Assembly and Test operations

by

Shihui Jia, 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, 2016

This dissertation is dedicated to my family.

Acknowledgements

I would like to thank my advisor Prof. Jonathan F. Bard for his guidance during my PhD research. He has shared his extensive knowledge and taught research skills to me, and has been a trustworthy and encouraging supervisor. He is an honorable, successful, and internationally recognized professor. His many great qualities as a researcher inspired me and pointed me in the right direction in order to have a successful career. I would like to thank Prof. Douglas Morrice who has patiently guided me and provided expert suggestions on the portion of my work related to simulation and statistical analysis. He has shown me how to extract insights from the analysis that have measurably improved the results. I also want to say thanks to the other members of my committee, Prof. John Hasenbein, Prof. Aida Khajavirad, and Dr. Zhufeng Gao, for their willingness to give me constructive feedback. During several years as PhD student, I am grateful to Texas Instruments for supporting my research. Dr. Rodolfo Chacon at TI provided invaluable industrial input including refining my research and sharing data from real manufacturing facilities.

I am sincerely grateful to my parents for their constantly caring and emotional support. They have always had faith in me and have encouraged me to pursue my dream.

Finally, I would like to thank my husband, Deyi Zhang. I have been extremely fortunate to have been able to share the joys and pains with him over my PhD years. He encouraged me when I become overwhelmed by the workload and stood by me through the good times and bad.

Simulation and optimization techniques applied in semiconductor Assembly and Test operations

Shihui Jia, PhD

The University of Texas at Austin, 2016

Supervisor: Jonathan F. Bard

The importance of back-end operations in semiconductor manufacturing has been growing steadily in the face of higher customer expectations and stronger competition in the industry. In order to achieve low cycle times, high throughput, and high utilization while improving due-date performance, more effective tools are needed to support machine setup and lot dispatching decisions. In previous work, the problem of maximizing the weighted throughput of lots undergoing assembly and test (AT), while ensuring that critical lots are given priority, was investigated and a greedy randomized adaptive search procedure (GRASP) developed to find solutions. Optimization techniques have long been used for scheduling manufacturing operations on a daily basis. Solutions provide a prescription for machine setups and job processing over a finite the planning horizon. In contrast, simulation provides more detail but in a normative sense. It tells you how the system will evolve in real time for a given demand, a given set of resources and rules for using them. A simulation model can also accommodate changeovers, initial setups and multi-pass requirements easily.

The first part of the research is to show how the results of an optimization model can be integrated with the decisions made within a simulation model. The problem addressed is defined in terms of four hierarchical objectives: minimize the weighted sum

of key device shortages, maximize weighted throughput, minimize the number of machines used, and minimize makespan for a given set of lots in queue, and a set of resources that includes machines and tooling. The facility can be viewed as a reentrant flow shop. The basic simulation was written in AutoSched AP (ASAP) and then enhanced with the help of customization features available in the software. Several new dispatch rules were developed. Rule_First_setup is able to initialize the simulation with the setups obtained with the GRASP. Rule_All_setups enables a machine to select the setup provided by the optimization solution whenever a decision is about to be made on which setup to choose subsequent to the initial setup. Rule_Hotlot was also proposed to prioritize the processing of the hot lots that contain key devices.

The objective of the second part of the research is to design and implement heuristics within the simulation model to schedule back-end operations in a semiconductor AT facility. Rule_Setupnum lets the machines determine which key device to process according to a machine setup frequency table constructed from the GRASP solution. GRASP_asap embeds a more robust selection features of GRASP in the ASAP model through customization. This allows ASAP to explore a larger portion of the feasible region at each decision point by randomizing machine setups using adaptive probability distributions that are a function of solution quality. Rule_Greedy, which is a simplification of GRASP_asap, always picks the setup for a particular machine that gives the greatest marginal improvement in the objective function among all candidates.

The purpose of the third part of the research is to statistically validate the relative effectiveness of our top six dispatch rules by comparing their performance on 30 real and randomly generated data sets. Using both GRASP and our ASAP discrete event simulation model, we have (1) identified the general order of dispatch rule performance, (2) investigated the impact of having setups installed on machines at time zero on rule

performance, (3) determined the conditions under which restricting the maximum number of changeover affects the rule performance, and (4) studied the factors that might simultaneously affect rule performance with the help of a common random numbers experimental design. In the analysis, the first two objectives, weighted key device shortages and weighted throughput, are used to measure outcomes.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Semiconductor devices are manufactured from wafers in a fabrication facility or fab in what are called *front-end* operations. Compared to other types of manufacturing, wafer fabrication is perhaps the most technologically complex and capital intensive due to long cycle times and the need to carry out a precise sequence of processing steps in a particle-free clean-room (Leachman 2002, Uzoy et al. 1992). After fabrication, the wafers are sent to an assembly and test (AT) facility where they are cut into chips, packaged, and tested in what are called *back-end* operations. During assembly, the chips are protected from environmental contamination by encasing them in plastic or ceramic material. Once the package is sealed, and tested for leaks and other defects, the end product is sent to final test. There, various operations are performed to guarantee that each device satisfies the customer's requirements before being shipped. During this process, a predefined sequence of steps is again followed that involves testing on several different machines. In many cases, the same machines are used at different steps giving rise to reentrant flow, a concept introduced by Graves et al. (1983) and common in wafer fabrication.

When scheduling AT operations, the goals are to achieve low cycle times, high throughput, and high utilization without violating agreed upon delivery dates. The first attempt to use optimization technology to achieve these goals was undertaken by Deng et al. (2010) who formulated the scheduling problem as a mixed-integer program (MIP) with the following four objectives given in order of priority: (1) minimize the shortage of critical devices, (2) maximize the weighted throughput of the remaining lots, (3) minimize the number of machines used, and (4) minimize the makespan. Solutions were obtained with a reactive greedy randomized adaptive search procedure (GRASP) designed to examine a diversity of machine-tooling combinations and lot assignments

over many iterations (the literature on GRASP is extensive; e.g., see Festa and Resende (2009) for an annotated bibliography of algorithms, and Feo et al. (1991) and Monkman et al. (2008) for manufacturing applications). In the original version of the model, Deng et al. assumed that the machines could only be set up once and that no jobs were running at time zero. An extension used here allows for initial setups as well as changeovers during the planning horizon, which can be anywhere from 24 hours to 7 days. The development of the original and updated versions of the GRASP stemmed from our unsuccessful efforts to solve the MIP with a commercial code.

Another feature that was sidestepped in original research was the need for lots to be processed multiple times as they progress through the facility. In fact, each lot must undergo a series of steps or operations defined by its *route* that are spaced no more than a predetermined number of minutes apart. When creating schedules, it is necessary to look ahead and take into account machine and tooling requirements for all steps in a route. Such multi-pass requirements are synonymous with reentrant flow where a job may return to a machine several times before its completion.

In a follow-on paper, Bard et al. (2013) present an updated methodology for dealing with the multi-pass requirements associated with a route. Whether more than one operation is actually scheduled, though, depends on customer demand, the relative priority of the lots in queue, and the current configuration of the shop. To decide on the best machine-tooling setups and how to assign lots to machines, a three-phase heuristic was used. In phase I, a single-pass algorithm derived from the GRASP in Deng et al. (2010) is run to obtain a tentative solution. Initial machine configurations are examined and the completion times of the lots in process at time zero are determined. For a given planning horizon, this calculation also determines the amount of time remaining on each machine. In phase II, the single-pass solution obtained in phase I is adjusted by inserting

second and higher-pass steps of the assigned lots into the schedule. In phase III, an attempt is made to reset machines when they finish processing the lots assigned to them, and then augment their schedules with additional lots. Using the GRASP logic, each phase is repeated a fixed number of times using randomization to ensure that diverse portions of the feasible region are explored.

In a parallel effort, we have also developed a discrete event simulation model that similarly schedules AT lots over a given planning horizon. The motivation for the simulation is several-fold. First, it allows more flexibility in machine setups and in applying lot assignment rules that may be more familiar, and in fact, preferred by shop supervisors. It is well known that trying to introduce new analytic techniques to support operations often meets with strong resistance. Simulation, however, has been used widely by industry and hence is likely to be more favorably received. This is especially true in semiconductor manufacturing where it provides the computational foundations for planning and scheduling; e.g., see Asmundsson et al. (2006), Health and Morrice (2007), Hung and Leachman (1996), Lin and Lee (2011). Second, simulation can easily accommodate changeovers, which are difficult to include in optimization models while maintaining any degree of tractability. Although we are able to deal with changeovers in our updated GRASP, the quality of the resultant solutions is open to question because no good benchmarks for comparative purposes exist. Third, simulation allows uncertainty to be incorporated into the analysis. If it were desirable, for example, to consider machine reliability, variable processing times, and changing lot priorities, we would be at a loss to do so with an optimization approach. Fourth, simulation offers a comprehensive view of the shop floor since starts, completions, and changeovers are reported as they occur.

With this in mind, the first purpose and contribution of this research is to provide a comparison of the strengths and weakness of using simulation and optimization to

schedule daily assembly and test operations. While there has been considerable research in combining the two methodologies to gain synergy (e.g., see Bispo and Tayur 2001, Lee et al. 2013, Xu and Nelson 2013), this is not our main goal. In our experience, it is rare that a manufacturer has the expertise and resources to support an integrated approach. Shop personnel generally rely on simple rules of thumb to construct schedules and sequence jobs, often validating them with a simulation of the facility (Aldakhilallah and Ramesh 2001, Lin and Lee 2011, Pfund et al. 2006, Sels et al. 2012). Nevertheless, as part of our validation process, we demonstrate how the two techniques can work together. The second purpose of this research is to describe our simulation model, which was built with AutoSched AP and is currently being used by the sponsoring company at several of their AT facilities.

The key to schedule AT operations is to choose the right machine-tooling configurations and also the lots assignments to machines. To decide on the best machine-tooling configurations and how to assign lots to machines, a three-phase heuristic was implemented. As a real-time alternative tool to the GRASP, Bard et al. (2015) developed a deterministic discrete event simulation model using AutoSched AP (ASAP) that similarly schedules AT lots over a given planning horizon. The built-in rules in ASAP performed poorly compared to the enhanced GRASP, though, so three new dispatch rules were formulated. Rule_First_setup initializes the simulation with the setups obtained with the GRASP. Rule_All_setups enables a machine to select the setup provided by the optimization solution whenever a decision is about to be made on which setup to choose subsequent to the initial setup. Rule_Hotlot is also created to prioritize the processing of the hot lots which are defined as those lots containing critical or key devices associated with production targets.

The customization feature in ASAP was used to implement the new rules. The specific motivation for combining the best features of the two approaches was several fold. First, the standard rules in ASAP are inherently myopic in that set up and dispatch decisions reflect the best choice for each machine at the current point in time. In contrast, GRASP makes decisions in full view of system capacity and prioritized demand for the entire planning horizon. Second, ASAP provides one solution while GRASP makes repeated runs to explore a large portion of the feasible region. Third, ASAP handles the multiple-pass (reentrant flow) requirements of a lot easily and more efficiently because it updates the unassigned lot list when the first pass of a lot finishes. In contrast, the enhanced GRASP only starts to process subsequent passes when all the first passes of lots that require the same setup are completed.

With this in mind, the third purpose and contribution of this paper are to (1) further customize ASAP rules by taking advantage of the type and frequency of machine setups recommended by GRASP results, (2) evaluate and compare the effectiveness of the various dispatch rules for machine setup and scheduling within ASAP, and more generally, (3) to demonstrate how to combine the logic of intelligent heuristics with discrete event simulation.

Jia et al. (2015) as discussed in Chapter 4 developed three more advanced dispatch rules for configuring machines and assigning lots to machines. The first gives priority to hot lots containing key devices while using the setup frequency table obtained from GRASP output. The second embeds the more robust selection features of GRASP in the ASAP model through customization. This allows ASAP to explore a larger portion of the feasible region at each decision point by randomizing machine setups using adaptive probability distributions that are a function of solution quality. The third rule, which is a

simplified version of the second, always picks the setup for a particular machine that gives the greatest marginal improvement in the objective function among all candidates. With all these attempts to solve the AT scheduling problem, we select six dispatch rules and compare their performance using 30 real and simulated data sets. Specific goals are to (1) discover the relative performance of each rule, (2) study the impact of having setups installed on machines at time zero on rule performance, (3) investigate how restrictions on the maximum number of changeovers during the planning horizon affects rule performance, (4) undertake an experimental design to study multiple factors that might affect the rule performance when taken together. In the analysis, the weighted sum key device shortages and the weighted throughput of lots are the two measure used to order rule performance.

In the next chapter, semiconductor manufacturing operations are outlined mainly with respect to back-end (assembly and test) operations. Chapter 3 explains how to combine simulation and optimization to create the new dispatch rules. Chapter 4 focuses on more advanced dispatch rules that exploit our intelligent heuristic to configure machines and assign lots, and Chapter 5 presents the performance analysis of our dispatch rules.

Chapter 2: Semiconductor Manufacturing Operations

2.1 AT Operations

An AT facility can be viewed as a flexible job shop with parallel, nonhomogeneous machines in each workcenter. For a typical planning horizon of 1 to 5 days, hundreds or even thousands of lots must be scheduled, where each lot may consist of hundreds of devices. Figure 1 depicts the major back-end operations, and may include anywhere from 10 to 30 steps (Van Zant 2000). The devices progress through some or all of these steps before being turned out as finished goods and either shipped to customers or placed in inventory. Because end-products differ in terms of dimensions, consumables, and process specifications, the process flows differ by lot.

Figure 2.1 shows in part that the test component collectively includes burn-in, electrical testing, marking/branding, baking, programming, mechanical scanning, quality check and packaging, in this order (Freed et al. 2006). In testing, each lot requires a unique subset of operations (burn-in, marking, baking, and programming may or may not be required), and several different machines may be eligible for each operation. In some cases, these machines may not be identical with respect to processing rates or output quality, so there may be lot assignment preferences among the set of eligible machines. Yield and lead-time variability in previous (front-end) stages of the manufacturing process results in variable lot sizes and lot priorities at the AT steps. Lot priorities range from low when ample inventory exists, to ‘hot’ or critical when promise dates are near or orders are past due.

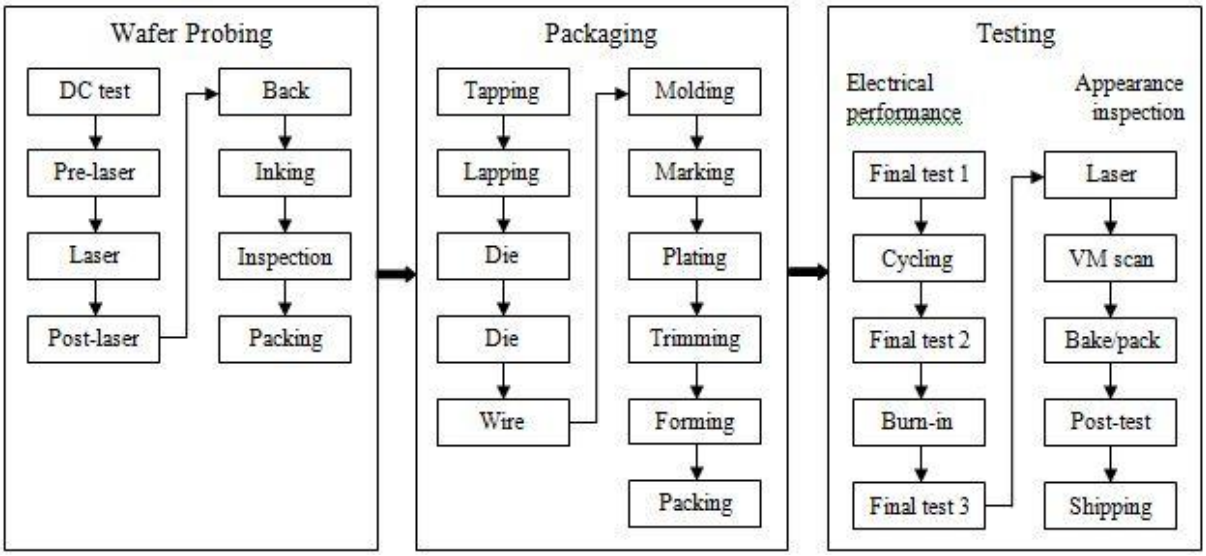


Figure 2.1 High-level back-end process flow

At back-end facilities, finished wafers go through an extensive regimen of inspection and testing that can take up to 3 hours at each step. Over a planning horizon of anywhere from 8 hours to several days, hundreds of thousands of wafers, grouped into thousands of lots must be assembled and tested. Each wafer goes through up to 32 sequential operations before it enters finished goods inventory. At each operation, a queued lot must be assigned to one of a subset of appropriate machines, which can number in the hundreds, and when two successive lots consist of different devices, a setup is incurred between lots. Setups or changeovers are performed by a crew of technicians and typically take two hours, although fewer hours may be needed, depending on the tooling. If the current device on a machine must be tested at a high temperature while its successor requires testing at room temperature, and both use the same fixtures, then the setup time is equal to the amount of time it takes for cool down, usually an hour. Labor may or may not be a constraining factor.

2.2 EXPLANATION OF TERMS

In specific terms, a lot is a collection of identical chips (also called devices or products) that follows a unique route through the facility consisting of a dozen or more steps. Each step corresponds to a machine operation that requires specific tooling, and must be performed at a predetermined temperature. Two lots may contain the same device but a different number of chips. A lot remains in the facility until it undergoes all of its prescribed operations. All lots are associated with customers and have delivery due dates. When a delivery is late, a penalty is incurred which is a function of lateness and volume. Because setups are so time consuming, it is critical for the planners to assign lots to machines and tooling to machines in such a way that as few setups as possible are required and due dates are taken into account.

The age of a lot is the current time minus the time it entered the facility. For each operation, each lot is assigned to a particular machine. To be eligible, the machine must be set up with the appropriate tooling pieces, as specified by the lot's *routing table,* and must be able to operate at the required temperature. Machines are divided into families. In most cases, two machines from the same family are identical; however, it is possible that "identical" machines operate under different temperatures and hence are not interchangeable. The limiting resource at most operations is the number of tooling pieces. As with machines, tooling pieces are divided into tooling families and only operate at a limited number of temperatures.

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. The step number in the route also affects the weight, with later steps having larger values or higher priorities. Two lots may consist of the same device but

differ in chip count, age, and upcoming step, and so will have different weights. Hot lots contain what are called *key* devices and have the highest priority. They are typically singled out for special treatment. It is thus desirable to ensure that as many of these "hot" lots as possible are processed over the planning horizon to avoid or reduce penalties. Regular 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.

There is a one-to-one relationship between a route and a device, but a route may really be a collection of subroutes that differ by machine, tooling, or temperature. At each step a different subroute can be selected, depending on the availability of machines and tooling. For example, route LTR-T3 might be associated with device QPWPRG4, and for step 1 (call it final test 1), there might be three different machine-tooling combinations (subroutes) that can be selected. Of the three, the first in the system's route table is typically the primary choice and the remainder, secondary or alternatives choices.

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, some lots (called *initial lots*) are likely to be in process so machines running them cannot be altered until the lots are completed.

Machines and tooling are divided into families with up to ten instances in each machine family and one to three instances in each tooling family. Each machine is allowed to operate only under a predetermined set of temperatures, so two machines from the same family, though identical, may not be able to process the same lots due to temperature considerations. The same can be said of the tooling pieces. 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 l_1, then after finishing lots l_1, it can run with another tooling setup λ_2 under another temperature τ_2 to process another lot set l_2 later in the planning horizon, when τ_2 is feasible for configuration λ_1.

2.3 PROBLEM STATEMENT

At the beginning of each planning horizon, typically a shift or a day, a finite number of lots are available for processing. A subset of these lots may contain what are called *key and package devices*. Any demand that cannot be satisfied for these two types of devices results in a large penalty. It is thus desirable to ensure that as many of these "hot" lots as possible are processed over the planning horizon to avoid or reduce penalties. Regular lots are assigned a value that depends on their age and remaining cycle time in the facility.

For a given planning horizon, available machines and tooling, and set of lots, the problem is to determine how to set up each machine with tooling to operate at a specified temperature so that the number of key devices falling short of their demand is minimized and the weighted sum of the lots processed is maximized without violating the system's capacity. These are the top two priorities; secondary objectives include minimizing the number of machines used and the makespan, in that order. In constructing schedules, it is

necessary to consider machine-tooling compatibility, machine changeovers when queues are empty, and the sequence of steps defined for a lot.

If a lot must undergo additional processing after completing its current operation, it is considered a *multipass lot*. Depending on the priority of such a lot and its current processing step, it may be necessary to include several subsequent operations in the upcoming schedule to ensure that its planned cycle time is not exceeded. As mentioned, each pass of a lot is associated with a step in the routing table and is called *lot-operation.*

2.4 BASIC MODEL

In this section, we present the basic model for the AT parallel machine scheduling problem with resource constraints which considers at most one setup for each machine. It is assumed that all machines are idle, all tooling pieces are detached, operating temperatures are predetermined (and hence, are omitted), and that setup and unloading times are negligible. Nevertheless, even with these simplifications the corresponding MIP requires a large amount of notation, and from a practical point of view, is intractable.

The notation and basic AT model follow.

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); $i \in M$

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

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

Parameters and data

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

$n_t^{tooling}$ number of tooling pieces in family t

$n_l^{devices}$ 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

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

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

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

$$\sum_{i \in M} \sum_{\lambda \in \Lambda(i,t)} b_{\lambda t} y_{i\lambda} \le n_t^{tooling} \text{ , } \forall\ t \in T, n \in N \tag{1d}$$

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

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

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

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

$$x_{ilr} \in \{0,1\}, \forall\ i \in M, l \in L(i), r \in R(i,l), y_{i\lambda} \in \{0,1\}, t_{i\lambda} \ge 0,$$
$$\forall\ i \in M, \lambda \in \Lambda(i),\ \Delta_k^{short} \ge 0, \forall\ k \in K, t^{max} \ge 0 \tag{1i}$$

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 makespan. 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.

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 $\Lambda(i)$. (At this point a more complete model would actually include tooling-temperature combinations, but as mentioned, temperature has been omitted.) 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 λ is selected for a particular machine, only lots compatible with that configuration 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. 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 and corresponding tooling setups. The right-hand side represents the number of tooling pieces in family t.

Constraints (1e) compute the 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, $C^{\max}$. 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 $C\Delta_k^{short}$ 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 $C = \max\{w_l : l \in L\} + 0.1\sum_{l\in L} w_l$ 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 full version of the AT model includes lot sequencing, changeovers, an accounting of lots running at time zero, multiple passes in a route, and a range of temperature options for machine setups (see Bard et al. 2013). Solutions are obtained with a GRASP, which, for convenience, is referred to as the "optimization" approach in the remaining sections. Of course, those solutions are not necessarily optimal since GRASP is not an exact algorithm. Ying and Lin (2014) studied a much simplified version of the AT problem with the single objective of minimizing the total setup time, a single route for each lot, and no hot lots in the mix. They proposed a hybrid artificial immune system algorithm to find solutions and tested it on a large number of randomly generated instances.

Solution methodology. To deal with initial lots, machine changeovers, and reentrant requirements, we have developed an enhanced GRASP whose various components can be adjusted to account for the current scheduling environment. For example, when a subset of the machines is set up at time zero, we want to be able to fix the corresponding y variables to 1 and bypass them in ULP. Also, provision must be

made to update the set of available tooling pieces when the queue of the machines they are assigned to empties. The general algorithm consists of the following eight steps.

Initialization

Step 1. (Initial lots) Identify the lots $\bar{L}$ that are currently in process and the machines $\bar{M}$ on which they are running. Identify the tooling setups $\bar{\Lambda}$ on those machines and the operating temperature. For each initial lot $\bar{l}(i) \in \bar{L}$, calculate its remaining time $\bar{C}_{\bar{l}(i)}$ on machine $i \in \bar{M}$ and let its remaining capacity be $\bar{H}_i = H_i - \bar{C}_{\bar{l}(i)}$.

Step 2. (Multi-pass lots) Identify the set of lots *LM* that require multiple passes through the work area being scheduled, noting that some of these might be initial lots. Replace each such lot $l \in LM$ with multiple copies – one for each pass – to get l^1, l^2, . . . Include only the first pass lots in *L*.

Step 3. (Solve basic MIP). Set $y_{i\lambda} = 1$ for all $i \in \bar{M}$ and $\lambda \in \bar{\Lambda}$. For the updated set of lots *L*, call GRASP to find a solution to model (1) with the objective limited to minimizing the shortages of the key-type devices, maximizing the weighted throughout, and minimizing the number or machines; that is, without consideration of the makespan. Let the solution be $\left(x^1, y^1\right)$, define $L^1 = \{\, l : x^1_{ilr} = 1 \;\forall\; i \in M,\, r \in R(i,l) \,\}$, and set $H_i^1 = \bar{H}_i + t_{i,\lambda(i)}$ for all $i \in M$, where $\lambda(i)$ is the tooling configuration assigned to machine *i* in the solution and $t_{i,\lambda(i)}$ is determined by Eq. (1e). Also, let t^1 be the earliest time that one of the scheduled machines becomes free.

Step 4. Call Minimize_Makespan with L^1 to get $C^{\max}$ and a new set of lot assignments, x^1.

General iteration γ

Step 5. (Add next multi-pass lot) For all $l \in LM$, if $l^m \in L^\gamma$, put $L \leftarrow L \cup \{l^{m+1}\}$; that is, add the next copy of multi-pass lot $l \in LM$ to *L*, where l^m is the m^{th} copy of *l*.

Step 6. (Changeovers)

6a. For the current solution (x^γ, y^γ), identify the first machine that becomes free when the lots assigned to it are completed and release the tooling pieces attached to it so they can be reallocated. Let t^γ be the time that this event occurs.

6b. Update the remaining time available with respect to t^γ; that is, set
$H_i^{\gamma+1} = \min\{ H_i^\gamma - t_{i\lambda^\gamma}, H_i - t^\gamma \}$ for all $i \in M$, where $\lambda^\gamma \in \text{argmax}\left\{ y_{i\lambda}^\gamma : \lambda \in \Lambda(i) \right\}$

6c. Considering the subsets of machines and tooling pieces available at t^γ, try to schedule as many of the remaining lots in L as possible with the GRASP. Allow the last lot on a machine to extend beyond the planning horizon. Call the solution $(x^{\gamma+1}, y^{\gamma+1})$.

Step 7. (Update WIP) Set $L^{\gamma+1} = \{ l : x_{ilr}^{\gamma+1} = 1 \ \forall \ i \in M, r \in R(i,l) \}$ and update WIP by putting $L \leftarrow L \setminus L^{\gamma+1}$.

Step 8. (Termination check) If $L = \varnothing$ or $H_i^{\gamma+1} \geq$ planning horizon – setup time(i,λ) for all $i \in M$ and $\lambda \in \Lambda(i)$, stop; otherwise put $\gamma \leftarrow \gamma + 1$ and go to Step 5.

Complexity of GRASP. Let $|D|$ be the number of devices, $|L|$ be the number of lots in the WIP, $|\Lambda|$ be the number of setups, $|M|$ be the number of machines, $|R|$ be the number of routes and $|T|$ be the number of tooling families. For the initialization process, the complexity of the four steps is $O(|M|+|L|+|\Lambda|) + O(|L|) + O(|M|\cdot|L|\cdot|\Lambda|+|M|\cdot\log(|M|)+|L|\cdot\log(|L|)+|L|\cdot|R|) + O(|M|\cdot|L|\cdot|R|)$. Given that $\log(|M|) \leq |\Lambda|\cdot|L|$ and $\log(|L|) \leq |M|\cdot|\Lambda|$, this summation reduces to $O(|M|\cdot|L|\cdot|\Lambda|) + O(|M|\cdot|L|\cdot|R|)$.

For a general iteration, the total complexity of Steps 5 – 8 is $O(|L|) + O(|M|\cdot|\Lambda|\cdot|L|+|M|\cdot\log(|M|)+|L|\cdot\log(|L|)+|L|\cdot|R|) + O(|M|\cdot|R|) + O(|M|\cdot|\Lambda|)$. Two simplifications are possible. The first is that $|M|\cdot|R|$ is less than $|L|\cdot|R|$ because the number of machine $|M|$ is less than number of lots $|L|$; the second is that the $\log(\cdot)$ terms are dominated by $|M|\cdot|\Lambda|\cdot|L|$. Removing all dominated terms, we get $O(|M|\cdot|L|\cdot|\Lambda| + |L|\cdot|R|)$ for a general iteration, so the complexity of one full GRASP iteration is $O(|M|\cdot|L|\cdot|R|) +$

$O(|M|\cdot|L|\cdot|\Lambda|)$. This value has to be multiplied by the total number of GRASP iterations denoted by N^{GRASP}. A more detailed complexity analysis of each step in the algorithm is in the appendix.

2.5 INTRODUCTION TO THE SIMULATION MODEL

In the context of AT operations, the full version of model (1a) – (1i) provides a prescription of what should be done over the planning horizon to best achieve the four objectives defined in (1a). Under ideal conditions, it might be possible to achieve these objectives, but uncertainty in machine and tooling availability, crew shortages and other disruptions might thwart the course of action proposed by the optimization approach. In order to give shift supervisors more visibility into the operations on the shop floor at any point in time, we developed a discrete event simulation model to support real-time decisions. The two models are intended to complement each other by improving both near-term and long-term system performance. In either case, the same inputs are required and include (i) the lots waiting for processing; that is, WIP, (ii) the preferred and alternative subroutes for each device, (iii) the machines and tooling available at time zero, (iv) lot weights and priorities, (v) targets for key devices, (vi) lots running at time zero, and all the values of the parameters that define model (1a) – (1i).

The simulation was written in AutoSched AP (ASAP), a product of Applied Materials, and is considered to be the standard analytic tool in the semiconductor industry. The configuration of an AT facility is fairly straightforward consisting of a set of machine families M and a set of tooling families F. The machines in each family are grouped together on the shop floor and the inactive tooling pieces are stored on racks by family. Members of the same machine family i, $m \in M_i \in M$, and tooling family j, $f \in F_j \in F$, are interchangeable.

In the current business environment, AT facilities generally runs 24 hours a day so when developing schedules for the next shift or the next week, it is necessary to take into consideration which lots are being processed on which machine at time zero, as well as how those machines are set up with respect to tooling and temperature. Once begun, each lot goes through a series of passes or processing steps on one or more machines determined by the (sub)route selected for it. As an example, consider lot 4030838 which contains the device TPS65856ZQZR. Referring to Table 2.1, the route for this lot is labeled "LJBG-T1" and consists of three steps 7100, 2110, and 7121. There are two options for 7100 and 7121and three options for 7110 as shown in the column "Alt". The primary or preferred subroute is blank and the secondary subroutes are identified by "alt". The column "Stnfam" indicates the station family, PPH or parts per hour is the processing rate on a machine in that family, "Genresfam" is the name of the generic resource family referred to in the industry as tooling, and the "Setup" column is simply a concatenation of the machine and tooling families. Note that the words *steps* and *passes* are used interchangeably from hereon out. For device TPS65856ZQZR, all three steps can be executed on machines in either of the families ETS-0-64 or ETS-1-128, but only step 7110 can use a machine in family ETS-1M-64.

Table 2.1 Example of a route

Route	Part	Step	Stnfam	PPH	Genresfam	Alt	Setup
LJBG-T1	TPS65856ZQZR	7100	ETS-0-64	570	6490924B		ETS-0-64_6490924B
LJBG-T1	TPS65856ZQZR	7100	ETS-1-128	570	6490924B	alt	ETS-1-128_6490924B
LJBG-T1	TPS65856ZQZR	7110	ETS-0-64	570	6490924B		ETS-0-64_6490924B
LJBG-T1	TPS65856ZQZR	7110	ETS-1-128	570	6490924B	alt	ETS-1-128_6490924B
LJBG-T1	TPS65856ZQZR	7110	ETS-1M-64	570	6490924B	alt	ETS-1M-64_6490924B
LJBG-T1	TPS65856ZQZR	7121	ETS-0-64	570	6490924B		ETS-0-64_6490924B
LJBG-T1	TPS65856ZQZR	7121	ETS-1M-128	570	6490924B	alt	ETS-1M-128_6490924B

Each lot in WIP has a unique identification number and contains one type of device or part. As mentioned, for example, lot 4030838 contains TPS65856ZQZR and follows route LJBG-T1. Now, suppose that machine AMAT16-1 is a member of machine family ETS-1-128 and that AMAT17-1 is a member of ETS-1M-64. One possible schedule for this lot is shown in Figure 2.2, where the first step uses setup ETS-1-128_6490924B and the next two steps use setup ETS-1M-64_6490924B.

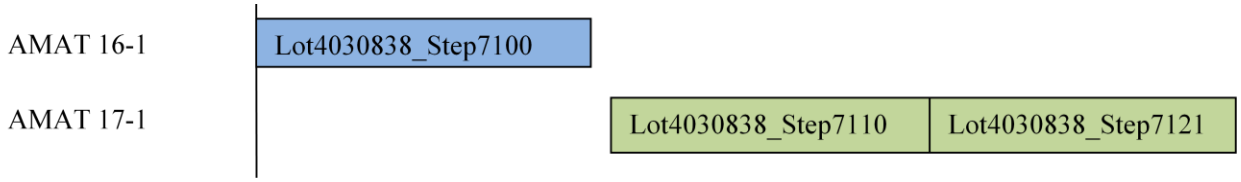


Figure 2.2 Example of a lot assigned to different machines for its passes

An AT facility falls somewhere between a reentrant job shop and flow shop since each lot follows a fixed path but may require the same machine with different tooling at successive steps. In other cases, a lot may require a different machine at each step. Unlike the optimization model (1a) – (1i) which assigns tooling to machines and lots to machines for the entire planning horizon with an objective function to guide the decisions, the simulation makes incremental, shortsighted decisions for each machine as soon as it finishes processing its current lot. In ASAP, machines use rules to select the next lot to work on from their family work list (FWL), which contains all the available lots that can possibly be processed. Each lot on the list is ranked by a predetermined priority measure. Rules and ranks work together to decide which lot a machine selects, so picking the appropriate combination is one way to try to optimize factory performance. ASAP has several built-in rules and ranking schemes. To guide these decisions, ASAP has a

number of built in rules and ranks that it can be applied. For a given work list (i.e., WIP), these include:

Rule_FIRST (first): Select the first job or lot on the work list.

Rule_FIRST_A (first and all available): Same as Rule_FIRST, except that the machine does not choose a lot from the list until all subparts and other resources needed to process the lot are available.

Rule_SSU (same setup): Continue processing lots that require the same setup that exists on the machine under consideration. This rule selects the first lot on the list that matches the current setup of the machine.

Rule_SSU_A (same setup all available): Same as Rule_SSU except that the machine does not choose a job from the list until all subparts and other resources needed to process the job are available.

Rank_FIFO (first-in, first-out): Lots are ranked based on when they become available, with the earliest lot given the highest priority. This is the default rule in AutoSched AP when no other priority scheme is specified.

Rank_HP (highest priority): Lots are ranked in order of their priority as specified in the input file, with the highest priority (integer value) chosen first.

Rank_EDD (earliest due date): Lots are ranked in order of their due date. The lot with the earliest due date is ranked first.

For example, the rule_SSU_A (same setup all available) lets a machine continue processing lots that require the current setup, assuming that all the required resources are available. The rank_HP (highest weight) orders the lots on the FWL according to their weight. Figure 2.3 presents an example of how machine AMAT 16-1 (belonging to station family ETS-0-64) with setup A (colored green) installed at time 0 makes lot selecting decisions at four different points in time, 0 to 3, using rule_SSU_A and

rank_HP. Assume that Lot 1, Lot 2 and Lot 3 all contain part TPS65856ZQZR and have to start from Step 7100, and the weight of each lot is specified in parentheses. The three lots are ranked from the highest weight to the lowest on the AMAT 16-1 FWL according to rank_HP. The list contains only the lots that are about to undergo their first pass at time 0. Suppose that the required tooling for setup A and setup B are both available at time 0. AMAT 16-1 will select lot2_7100 with 90 devices, which is the first ranked lot requiring the same setup as the machine is currently using. At time 1 when machine AMAT 16-1 finishes step 7100 of lot 2, it looks at the updated FWL (which contains the lot just finished) and selects lot2_7110 since it is the first ranked lot on its FWL requiring setup A. Similarly, lot2_7121 will be selected at time 2 when the second pass of lot 2 is finished. At time 3 when the third pass of lot 2 is finished, there are no more lots on the FWL that can be processed with setup A so a changeover is required. The two remaining lots both require setup B so there is no choice. Given that we are using rank_HP, the first ranked lot, Lot1_7100, is selected for processing.

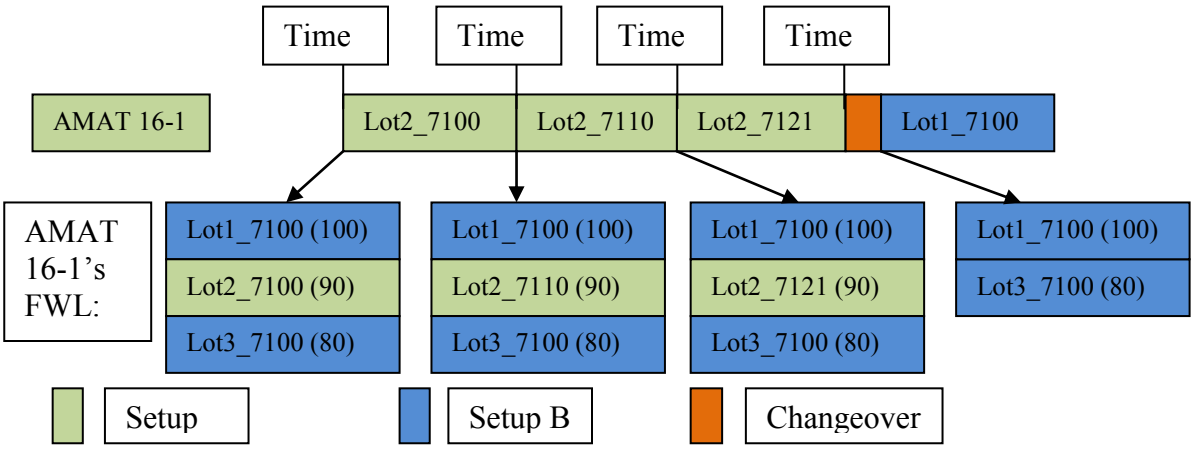


Figure 2.3 Example of lot selection with Rule_SSU_A and rank_HP

Rules and rankings work together to select the next lot to assign to a machine and to decide when changeovers are required. Of course, processing cannot start on a machine until the necessary tooling is available. During the first phase of the project, we ran the simulation using Rule_SSU_A and Rank_HP and obtained the results in Table 2.2 for the first of 6 data sets provided by two Asian AT facilities of a major manufacturer. The optimization results obtained from our GRASP are also included. For the simulation, lots that contained key devices were assigned higher priorities than regular lots for which no targets were given.

Table 2.2 Optimization and simulation comparisons

Prob. no.	Avg. lot processing time†	Simulation model			Optimization model (GRASP)		
		Total steps finished	Total lots finished	No. of machines used	Total steps finished	Total lots finished	No. of machine used
1	8.02 (9.33)	576	392	36	688	419	36
2	8.02 (9.33)	570	379	36	619	336	36
3	11.35 (16.84)	483	352	36	647	378	36
4	11.35 (16.84)	457	326	36	667	420	36
5	117.55 (110.48)	119	116	131	123	121	48
6	117.55 (110.48)	126	119	135	122	120	71

†time in hours; standard deviation in parentheses.

The data for problem nos. 1 – 4 were provided by section of the first facility which contains 36 machines while the data for the remaining instances come from the second facility which contains 136 machines. The details associated with each instance are given in Section 2.7. From the Table 2.2 we see that the GRASP generally outperforms the simulation with respect to the two performance measures reported: total processing steps finished and total lots finished. Two exceptions are problem no. 2 with respect to lots finished and problem 6 with respect to steps finished. In the latter case, it is worth noting that the optimization model used less than half the number of the machines than the simulation.

The difference in machine usage is due to the difference in logic used by the two methodologies. The simulation is myopic and tries to process each lot as soon as possible even if that means setting up every machine in the facility. As a consequence some machines may be assigned as few as one lot. In contrast, the optimization model takes a longer view and tries to determine the "best" machine-tooling combinations with respect to all the lots in WIP. Lots are then assigned to machines in groups rather than one at a time. Also, the simulation model does not consider the processing time of a lot and so makes assignments based only on the dispatch rule in use. The optimization model,

however, will only assign a lot to a machine if the current step can be completed within the time remaining on the machine. From Table 2.2, it can be seen that the average processing time of the lots in problem nos. 5 and 6 is almost as long as the planning horizon which is 120 hours. As a result the simulation model begins processing many more lots than the optimization model but doesn't finish as many and doesn't complete as many steps. Extending the planning horizon another few days would better even things out.

The overall results are not surprising since the optimization looks over the entire planning horizon when making decisions while the simulation makes local, greedy decisions using dispatch rules and prioritized rankings. In all cases, the GRASP set up a smaller or equal number of machines and came closer to meeting the key device targets (not shown in Table 1) than did the simulation. This is a direct result of hierarchical objective function in (1a) that drives the GRASP. In contrast, the simulation model does not look ahead; it always sets up an available machine if there is a lot on the work list that could be processed by the machine.

In an effort to improve the performance of the simulation we tried to individually and collectively adapt the dispatching rules built into AutoSched AP so that they approximated the logic used by the GRASP to set up machines and assign lots to them. The effort failed, however, on several counts, mainly due to the inherent rigidity of the rules. As a consequence, we decided to develop our own rules that more closely mimicked the steps in the optimization code. The approach taken is described in the next chapter.

Chapter 3: Design of new Scheduling Rules

To go beyond the standard functions and commands available in AutoSched AP it is necessary to embed user-written C++ code in the simulation model at each decision point where modifications to existing rules are desired. The process is called customization and offers a powerful means of increasing system performance.

3.1 INITIALIZATION: RULE_FIRST_SETUP

The solution to the optimization model provides the "best" machine-tooling setups, lot assignments and changeovers for the given planning horizon. The general goal is to maximize the use of the available resources in light of the four hierarchical objectives. The first setup used by each machine plays an important role in achieving those objectives because it assures that lots with the greatest benefits are given top priority even though they may not be at the beginning of the optimal sequence. In fact, the GRASP does not sequence the lots at first but simply assigns them to machines. Feasible sequences are only determined when second and higher number passes are being considered. This is in contrast to the simulation which by design sequences each lot as it is assigned to a machine so solutions are always feasible.

Our first modification is to initialize the simulation with the setups obtained with the GRASP. This is achieved by including the appropriate values in the "Cursetup" field in ASAP's machine file "stn.txt" which represents which setup the machine uses at time 0. The approach is termed Rule_First_setup. To select lots, we still use Rule_SSU_A and Rank_HP. For those machines that are not included in the GRASP solution, we again rely on the two latter rules to perform the initial setups and lot selections.

Figure 3.1 illustrates the potential advantage of using Rule_First_setup. In the upper panel, if the machine starts with the setup specified by AutoSched AP (setup A in the example), then according to the assumed work list, lots 1, 2 and 3 can be processed

after a 30-minute setup. Next, setup B is selected which allows lots 4 and 5 to be processed before the end of the planning horizon is reached. In the lower panel, when the "Cursetup" value (setup C in the example) is specified for the machine, it is possible to process all five lots plus lot 6. By looking ahead, the optimization model determines that setup C will allow more lots to be processed with fewer changeovers than then starting with setup A.

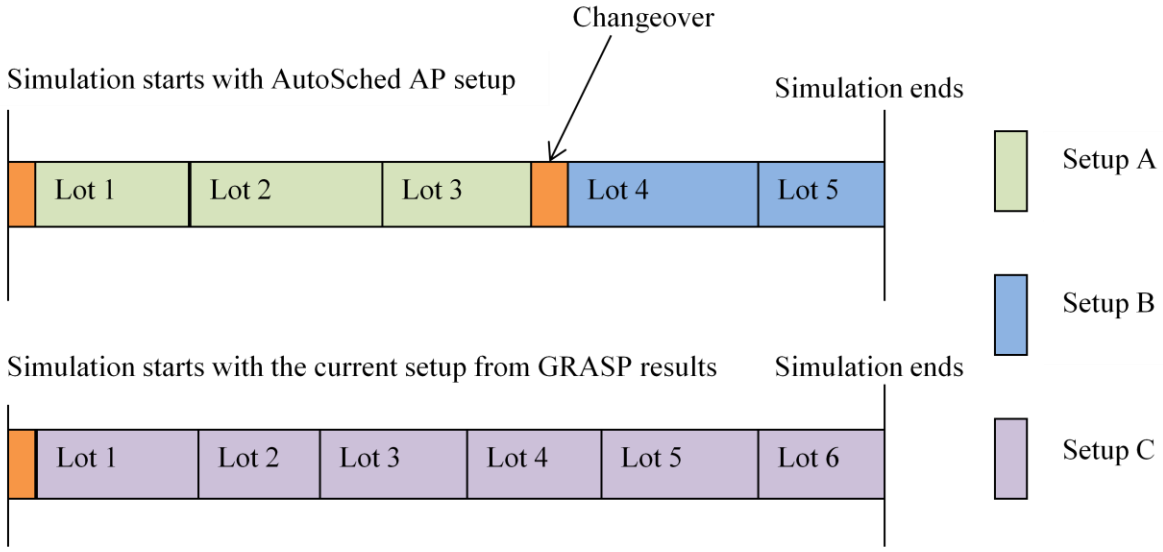


Figure 3.1 Gantt charts with and without exploiting the optimization result

3.2 RULE_ALL_SETUPS

When Rule_First_setup is used, the simulation tries to process all lots on the work list that can be accommodated under "Cursetup," and when no lots remain that are feasible, it looks to Rule_SSU_A and rank_HP to reconfigure the machines. As an enhancement, we constructed Rule_All_setups, which uses all the setups provided by the GRASP and not just the first setup. In other words, whenever a decision is about to be made on which setup to choose subsequent to the initial setup, those provided by the

optimization solution are selected. However, because the logic used by the simulation and GRASP to pick lots is different, using Rule_All_setups rarely produces the same solution as the GRASP.

The upper and lower panels in Figure 3.2 complement each other by illustrating the cases in which Rule_First_setup and Rule_All_setups are used, respectively. Both start out with setup A as specified by "Cursetup" and are able to process lots 1, 2, and 3. Upon completion, the machines must be reset to make use of their remaining capacity. For the first machine, Rank_HP identifies the highest priority lot on the work list and, depending on the available tooling, chooses a specific setup. In this case, it is setup B although setup D was also a possibility. After lots 4 and 5 are finished, a second setup is called for. This time it is setup C. For the second machine, setup D is chosen because that's what was indicated by the GRASP solution. As a result, it is possible to process lots 4 – 6 plus lot 7 in the remaining time. Although the example is somewhat contrived, it demonstrates what might happen with and without Rule_All_setups.

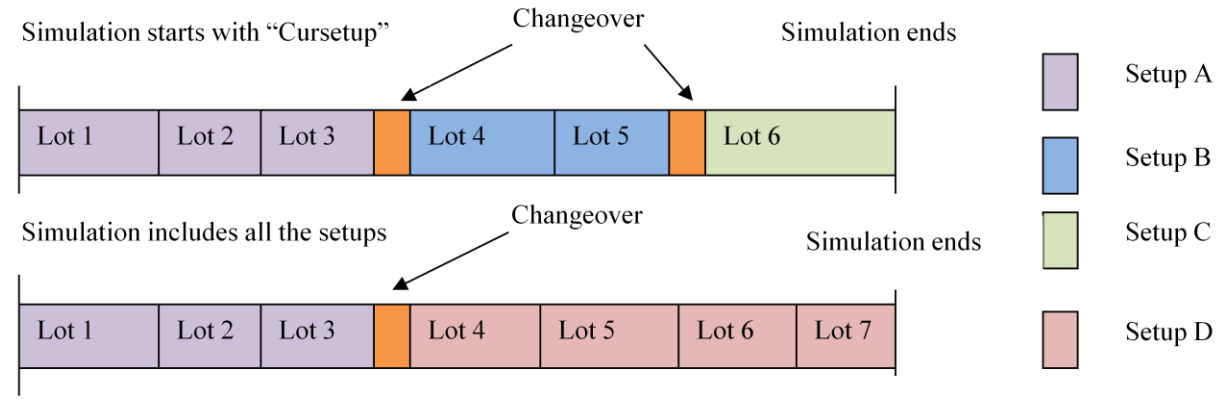


Figure 3.2 Gantt charts for assigning the first setup only verses all setups

3.3 RULE_HOTLOT

Critical devices or hot lots are given the highest priority in model (1) as long as their target is not met. When a sufficient number of lots containing a specific key device are processed through all their steps, the weights of the remaining lots that contain that device revert to their regular values. Rule_HotLot is designed to improve the due date performance of the simulation in a similar manner. The goal of minimizing the shortage of key devices in the optimization model is equivalent to maximizing the throughput of the hot lots in the simulation model. We want to be careful, though, not to tie up resources and continuing processing hot lots after their target is met, unless the hot lot list is empty.

To construct the hot lot list, we do the following: for each key device, first sort those lots that contain this device from the highest weight to the lowest weight, and then sequentially place a sufficient number on the hot lot list until the sum of pieces in the lots selected meets or exceeds the target. For example, assume that part XPS54286PWPR is a key device with a target of 22,910 pieces. Table 3.1 lists the lots that contain XPS54286PWPR and the number of pieces in each. Because they are already sorted in accordance with their priority, we go down the list until the target is met. For the seven lots in the table, the first five are selected and designated as a hot lot (see last column) since the sum of their components is 29,293. This value exceeds the target by as little as possible while preserving the original order.

Table 3.1 An example of hot lot designation

Lot	Part	Current step	Pieces	Original priority	Updated priority	Hotlot
9911008	XPS54286PWPR	7100	4536	4304400	22910	yes
9911014	XPS54286PWPR	7100	6371	4304400	18374	yes
9921414	XPS54286PWPR	7100	1106	3903600	12003	yes
9923471	XPS54286PWPR	7100	8640	3835400	10897	yes
9923472	XPS54286PWPR	7100	8640	3835400	2257	yes
9923473	XPS54286PWPR	7100	8640	3835400	3835400	no
9923474	XPS54286PWPR	7100	8640	3835400	3835400	no

After constructing the list, the priority of each lot is modified to reflect its distance from the target, assuming that lots are processed in nonincreasing order of their original weight. Taking each key device separately, the priority of the first hot lot is set to the target; the priority of the second hot lot is set to the priority of the first lot minus the number of pieces in the first lot, and so on. For example, there are five hot lots in Table 3.1 for key device XPS54286PWPR. We set the priority of lot 9911008 to the target 22,910. Next, we set the priority of lot 9911014 to 22,910 – 4536 = 18,374, and similarly for the remaining three lots. The results are given in the "Updated priority" column in Table 3.1.

After defining hot lots and modifying their corresponding weights, they will be assigned to machines according to the logic in Figure 3.3 at a high level. Lots pass through a series of filters and the ones that emerge are assigned to a machine on which it can be processed. Machines are ranked in nonincreasing order of their capacity with ties being broken arbitrarily. Each station family has associated with it a family work list (FWL) of feasible lots. Hot lots on the FWL are the primary candidates for processing and will be selected in accordance with their priority when the current setup can accommodate them. If not, a decision is made on how to reset the machine so that it can continue processing hot lots. At this point, when there are two or more hot lots on the

FWL, all those that can be processed by a machine in the same family using its current setup are bypassed because a feasible setup already exists for them. The remaining hot lots for which there is no current feasible setup are considered one at a time according to their priority. The highest ranking one for which the required resources are available is selected and the changeover is performed.

When it is not possible to reset the machine to process any of the remaining hot lots on the FWL, regular lots are considered. The one with the highest rank that can be processed with the current setup is selected first. Lower ranking regular lots are selected in turn until a changeover is required. At that point, the same logic used to reset the machine for hot lots is used for the regular lots.

In summary, taking hot lots first, the machine will first try to select the first ranked lot that has all of the required resource available and can be processed with the current setup. If this fails, then the machine will select the first ranked lot that has all of the required resource available that cannot be processed with the current setup used by other machines in the same machine family. If that still fails, the machine will just select the first ranked lot for which all of the required resources are available. This logic is designed to process hot lots first even if changeovers are required. In contrast, the GRASP tries to process all feasible lots before considering a changeover. Therefore, when Rule_Hotlot is used, the simulation may outperform the GRASP with respect to minimizing shortages.

To describe the logic associated with Rule_Hotlot in more detail, we make use of the following additional notation.

Sets and indicies

M set of machines in the factory; $i \in M$

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

Hotlot set of hot lots

$M(i)$ set of machines belonging to the same machine family as machine i

$DES(i)$ set of lots that passes through various filters for $l \in FWL(i)$

$Allavail(i)$ set of lots in $FWL(i)$ for which the required machine and tooling are available; $Allavail(i) \subseteq FWL(i)$

$\lambda(i)$ current setup for machine i

$\lambda(i,l)$ setup that can be used to process lot l with machine i

$\Lambda(M(i))$ setups used by the machines belonging to $M(i)$ except machine i

Alorigthm_Rule_Hotlot

Step 0. Initialization:

For current i, determine the entries in the sets $FWL(i)$, $Allavail(i)$, $DES(i)$, $M(i)$, $\lambda(i)$ and $\Lambda(M(i))$. For each $l \in Allavail(i)$, retrieve the values of $\lambda(i,l)$ and determine whether l is a hot lot.

Step 1. Rank all of hot lots in the set $FWL(i)$ in nonincreasing order of their priority, and then below them rank all regular lots in $FWL(i)$, also in nonincreasing order of their priority.

Step 2. Apply multiple filters to $FWL(i)$ and put the lots that pass through these filters to the $DES(i)$.

If $Allavail(i) \neq \varnothing$, then

FOR each $l \in Allavail(i)$

If $l \in Hotlot$ and $\lambda(i) = \lambda(i,l)$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i)$

If $l \in Hotlot$ and $\lambda(i,l) \notin \Lambda(M(i))$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i)$

If $l \in Hotlot$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i)$

If $\lambda(i) = \lambda(i,l)$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i)$

If $\lambda(i,l) \notin \Lambda(M(i))$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

$DES(i) \leftarrow Allavail(i)$

Go to Step 3.

Endif

Step 3. Select the first lot of the $DES(i)$ to assign to machine i.

Figure 3.3 Logic for Rule_Hotlot

Complexity of Rule_HotLot. Let $|M|$ be the number of machines in the factory, $|L|$ the number of lots in WIP, $|T|$ the number of tooling pieces, and $|\Lambda|$ the number of feasible setups. At Step 0, determining $FWL(i)$ requires an examination of each lot in L so the complexity is $O(|L|)$ for machine i. Similarly, $Allavail(i)$ can be determined in $O(|L|\cdot|T|)$ time, $M(i)$ in $O(|M|)$ time, $\lambda(i)$ in $O(|\Lambda|)$ time, and $\Lambda(M(i))$ in $O(|M|\cdot|\Lambda|)$ time. The complexity of determining whether l is a hot lot is $O(|L|)$ and the complexity of finding the entries in λ (i,l) is $O(|L|\cdot|\Lambda|)$. Recognizing that $|M|\times|\Lambda|$ is dominated by $|L|\times|\Lambda|$ because $|M|$ is small than $|L|$, the complexity of Step 0 reduces to $O(|L|\cdot|T| + |L|\cdot|\Lambda|)$.

Step 1 involves the ranking of all the lots in $FWL(i)$ and can be done $O(|L|\cdot\log(|L|))$ time. Considering Step 2, for any i, there are at most six filters (a filter is

referred to by an "if" statement before placing a lot l in the set $DES(i)$ in Step 2) that will be applied to the $FWL(i)$. The second and the fifth filters take $O(|\Lambda|)$ time while the remainder filters take constant time $O(1)$. Thus Step 2 requires at most $O(|L|\cdot|\Lambda|)$ time. Taking into account that Step 3 can be performed in $O(1)$ itme and that $\log(|L|) \leq |M|\cdot|\Lambda|$, the complexity of Rule_HotLot for a single machine is $O(|L|\cdot|T| + |L|\cdot|\Lambda|)$.

3.4 RULE_FIRST_SETUP_LIMITED

Recall that the third objective in model (1) is to minimize the number of machines used, a component of the GRASP but not a feature of the simulation as originally conceived. Referring to the comparisons in Table 2, we see that for two of the six data sets the simulation using Rule_SSU_A and Rank_HP required many more machines than the GRASP but failed to achieve a noticeable improvement in performance. These rules try to limit the number of machine setups but rarely achieve this goal. To avoid unnecessary setups we developed Rule_ First_setup_limited which restricts the machines that the simulation can use to only those that are included in the GRASP solution. This rule works in conjunction with Rule_SSU_A and Rank_HP.

3.5 DISCUSSION AND CONCLUSION

Backend operations in semiconductor assembly and test facilities give rise to reentrant flow. Each lot needs to be processed using different tooling in a predefined sequence. Scheduling thousands of lots a day to minimize throughput, minimize key device shortage, reducing the number of machines used and minimize the makespan is a complex problem that has not been adequately addressed by the research community. In this paper, the optimization and the simulation procedures were both presented and compared to schedule the back-end operations in the semiconductor manufacturing. The

mathematical model was formulated and high-quality solution for the multi-pass scheduling problem was obtained by using the GRASP heuristics.

The simulation model was developed using the simulation software AutoSched AP 10.0.2, but the software built in rules performed poorly compared to the GRASP results from almost every measurement that was used in this paper. Three new rules were designed by using AutoSched AP customization aims at improving simulation model performance. The Rule_First_setup initializes the simulation with the setups obtained with the GRASP. The Rule_All_setups uses all the setups provided by the GRASP. The Rule_HotLot focuses highly on minimizing the key device shortage by defining the hotlots and developing new algorithm for the machines to do lot selection. The Rule_Setupnum puts higher priority in processing the hotlots and also take account into the setup frequency results from machine optimizer. Extensive computational comparisons were made between different rules and the GRASP results using various metrics with six real datasets provided by the Taiwan and Clark probe AT facilities of the collaborating company. The computational results showed that the Rule_First_setup and the Rule_All_setups have both improved the performance the basic simulation rule in most of the performance measures, and these two rules even outperformed the GRASP when comparing the total lots finished and the key device shortage. These three rules combine the merits of optimization model and the simulation model by using the same setups as the optimization model and grouping the sequential steps in a lot's route. The Rule_HotLot performed the best in minimizing the key device shortage as how it was designed.

Chapter 4: Improving the performance of dispatch rules in A&T operations

Three new dispatch rules with more intelligent algorithms are developed in order to combine the merits of both GRASP and ASAP with customization. The comparison results of all developed dispatch rules on eight data sets are presented in Section 4.5.

4.1 LITERATURE REVIEW

For an overview of dispatch rules typically applied in the semiconductor industry, see Atherton and Atherton (1995). Wu et al. (2008) developed a dispatching algorithm that tries to balance the output rate of each product segment with the goal of improving on-time delivery for a make-to-order semiconductor wafer fab. They showed that the algorithm outperformed the scheduling procedures favored by the company on 10 test scenarios with respect to on-time delivery rates and cycle times. Saito (2007) proposed a pseudo periodical priority dispatching (P3D) rule for dynamic allocation of WIP in mixed products semiconductor manufacturing. The P3D rule evaluated both the amount of WIP and the arrival rate of lots for each quantum, where a quantum is defined as a period during which a single type of product is processed on a machine. Results comparing P3D with first-come, first served logic, and the shortest processing time rule for simulated data with Poisson arrivals showed that P3D uniformly outperformed the other rules in terms of adjustment rate, throughput, response time, and tardiness.

For scheduling semiconductor back-end operations, Chiang (2008) introduced a fuzzy analytical hierarchy process to identify acceptable WIP deviation levels, which were then used to determine job priorities. The approach was shown to balance on-time delivery goals and WIP targets with the help of a simulation model that was calibrated with real data. Fu et al. (2011) presented a MIP model and a deterministic scheduling system (DSS) to minimize prioritized tardiness for the weekly production scheduling of a

semiconductor back-end facility. Depending on customer orders, the DSS uses either a linear programming optimizer or a material-requirements-planning optimizer along with one of two scheduling rules: dynamic lot prioritization or dynamic machine prioritization, for finding schedules. The results were consistent and satisfactory from management's point of view, and required less solution time for randomly generated large problem instances than the MIP formulation. Related research in a job shop environment was undertaken by Sels et al. (2012) who compared 30 rules under two flow time-related and three tardiness-related objectives.

For a single product, Narahari and Khan (1996) proposed an approximation method for predicting the performance of heuristics for scheduling reentrant flows based on mean value analysis (MVA). They modeled reentrant lines with buffers as a non-traditional queuing network and were able to show that MVA was better than simulation-based methods with respect to accuracy and time complexity. One shortcoming of their approach was the need to treat each machine as a unique family, so they couldn't take advantage of situations in which some machines were identical. To address the more general case, Park et al. (2002) considered a facility that processed multiple products using multi-servers, where each server consisted of one or more identical machines. Choi et al. (2011) proposed a decision tree-based real-time scheduling mechanism for the reentrant hybrid flow shop scheduling problem. A decision tree was created using four attributes related to the jobs in the queue; the extremities of the tree contained the proposed dispatching rule of which one was identified as being the best through the roll-up logic. Testing showed that the approach led to higher throughput in less time when compared to discrete event simulation.

Freed et al. (2007) developed a dispatcher within an Excel-VBA decision support system. The dispatcher takes current WIP data and sorts it based on due-date and

processing requirements, and gathers feedback from managers to prioritize the use of resources before providing the operators with the final schedule. Testing showed that on-time delivery increased from 70% to 90% and lot lead time was reduced by 30% due to the dispatcher. Knutson et al. (1999) proposed a method for deciding the lot assignments on a given day with the overall goals of maximizing on-time delivery and minimizing excess product that had to be stored. The problem was formulated as nonlinear integer program with three objectives: maximize the number of die sent to the customers, minimize the number of die sent to the warehouse, and meet due date requirements for orders. A two-stage decomposition approach was used to find solutions. Stage 1 consisted of a knapsack problem whose objective was to maximize a combination of factory utilization and on-time-delivery, while Stage 2 was a modified bin covering problem in which the orders represented variable size bins. A first-fit-decreasing (FFD) heuristic with order sizes modified by their due date was used in Stage 1. The results were used to fill orders one at a time in Stage 2. The results showed that FFD performed significantly better than a FIFO algorithm.

Song et al. (2007) applied ant colony optimization (ACO) to reduce the conversion time of a bottleneck machine during assembly and test. Machine conversion is necessary when the product type switches between time intervals. The authors considered three objectives: minimize total unsupported customer demand, minimize the total number of conversions, and minimize the total conversion time. The optimization problem was mapped to an undirected multipartite network and solved using ACO. It was shown that this approach was able to reduce conversion time by 20% compared to the quantity-per-shift method, which was in use at the time.

4.2 RULE_SETUPNUM

When Rule_Hotlot is used, the simulation tries to process hot lots first, and when there are multiple types of hot lots remain on the FWL that are feasible, the first ranked hotlot with required setup not used by other machines in the same machine family will be selected. As another option, we constructed Rule_Setupnum, which is to first determine how many same setup as the hot lot's required setup that has been already used by other machines in the same machine family named as setup count, and then select the hot lot with setup count less than the frequency number listed in the setup frequency table. Rule_Setupnum uses the same logic of constructing the hot lot list and update the priority, the algorithm requires the setup frequency table from the machine optimizer. Table 4.1 lists the setup results provided by machine optimizer for machine family ETS564.

Table 4.1 An example of setup result from machine optimizer

Machine instance	Setup 1	Setup2
T4	ETS564_6455407A	ETS564_6462741B
T5	ETS564_6442302C	ETS564_6462741B
T6	ETS564_6440109A	ETS564_6462741B
T7	ETS564_6453620A	ETS564_6462741B
T10	ETS564_6430442A	ETS564_6462741B
T13	ETS564_6462741B	
T18	ETS564_6459957B	

According to the machine optimizer result, machine T4 used the setup ETS564_6455407A as the initial setup and then changed to setup ETS564_6462741B to process lots until no more lots left. Machine T13 has used one type of the setup ETS564_6462741B. Table 5 is an example of a setup frequency table that summarized the setup result.

Table 4.2 An example of setup frequency from machine optimizer

Setup	Setup number by optimization
ETS564_6430442A	1
ETS564_6440109A	1
ETS564_6442302C	1
ETS564_6453620A	1
ETS564_6455407A	1
ETS564_6459957B	1
ETS564_6462741B	6

Table 4.2 tells that the setup ETS564_6430442A has been set up once in the whole planning horizon, and similarly the setup ETS564_6462741B has been used for 6 times by the machines according to the optimization result. The following is an example to show the use of the setup frequency table. If machine T4 wants to select a lot, and hot lot A that requires setup ETS564_6430442A and hot lot B that requires setup ETS564_6462741B are both feasible, next is to determine the setup counts for both setups. In the simulation model, machine T5 that belongs to the same machine family as T4 is currently using setup ETS564_6430442A which makes the setup count of ETS564_6430442A as 1, and similarly the setup count of ETS564_6462741B is found as 1. So the hot lot that requires ETS564_6462741B will be selected by T4 since its setup count 1 is less than the frequency number 6 that is listed in the setup frequency table. After defining hot lots, modifying their corresponding priorities and obtaining the setup frequency table, the machine will select lots according to the logic in Figure 6. Overall, lots pass through a series of filters and the ones that emerge are assigned to a machine on which they can be processed.

Taking hot lots first, the machine will first try to select the first ranked lot that has all of the required resource available and can be processed with the current setup. If this fails, then the machine will select the first ranked lot that has all of the required resource available with the required setup count less than the frequency in the setup frequency

table. If that still fails, the same logic used by Rule_Hotlot for regular lots is used to select the regular lots.

To describe the logic associated with Rule_Setupnum in more detail, we make use of the following additional notation.

Sets and indicies

M	set of machines in the factory; $i \in M$
$FWL(i)$	set of lots that can be processed on machine i
Hotlot	set of hot lots
$M(i)$	set of machines belonging to the same machine family as machine i
$DES(i)$	set of lots that passes through various filters for $l \in FWL(i)$
$Allavail(i)$	set of lots in $FWL(i)$ for which the required machine and tooling are available; $Allavail(i) \subseteq FWL(i)$
$\lambda(i)$	current setup for machine i
$\lambda(i,l)$	setup that can be used to process lot l with machine i
$\Lambda(M(i))$	setups used by the machines belonging to $M(i)$ except machine i
$O(\lambda)$	setup frequency used by machine optimizer for setup λ
$CT(i,l)$	count of the setup that required by lot l on machine i used by other machines in the same machine family

Alorigthm_Rule_Setupnum

Step 0. Initialization:

For current i, determine the entries in the sets $FWL(i)$, $Allavail(i)$, $DES(i)$, $M(i)$, $\lambda(i)$ and $\Lambda(M(i))$. For each $l \in Allavail(i)$, retrieve the values of $\lambda(i,l)$ and determine whether l is a hot lot.

Step 1. Rank all of hot lots in the set $FWL(i)$ in nonincreasing order of their priority, and then below them rank all regular lots in $FWL(i)$, also in nonincreasing order of their priority.

Step 2. Apply multiple filters to $FWL(i)$ and put the lots that pass through these filters to the $DES(i)$.

If $Allavail(i) \neq \varnothing$, then

FOR each $l \in Allavail(i)$

If $l \in Hotlot$ and $\lambda(i) = \lambda(i,l)$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i) \cap Hotlot$

FOR each $m \in M(i)$

If $\lambda(m) = \lambda(i,l)$

$CT(i,l) = CT(i,l) + 1$

ENDFOR(m)

If $CT(i,l) < O(\lambda(i,l))$ then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i)$

If $\lambda(i) = \lambda(i,l)$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i)$

If $\lambda(i,l) \notin \Lambda(M(i))$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

$DES(i) \leftarrow Allavail(i)$

Go to Step 3.

Endif

Step 3. Select the first lot in $DES(i)$ to assign to machine i.

Figure 4.1 Logic for Rule_Setupnum

Complexity of Rule_Setupnum. Let $|M|$ be the number of machines in the factory, $|L|$ the number of lots in WIP, $|T|$ the number of tooling pieces, and $|\Lambda|$ the number of

feasible setups. At Step 0, determining $FWL(i)$ requires an examination of each lot in L so the complexity is $O(|L|)$ for machine i. Similarly, $Allavail(i)$ can be determined in $O(|L|\cdot|T|)$ time, $M(i)$ in $O(|M|)$ time, $\lambda(i)$ in $O(|\Lambda|)$ time, and $\Lambda(M(i))$ in $O(|M|\cdot|\Lambda|)$ time. The complexity of determining whether l is a hot lot is $O(|L|)$ and the complexity of finding the entries in $\lambda(i,l)$ is $O(|L|\cdot|\Lambda|)$. Recognizing that $|M|\times|\Lambda|$ is dominated by $|L|\times|\Lambda|$ because $|M|$ is small than $|L|$, the complexity of Step 0 reduces to $O(|L|\cdot|T| + |L|\cdot|\Lambda|)$.

Step 1 involves the ranking of all the lots in $FWL(i)$ and can be done $O(|L|\cdot\log(|L|))$ time. Considering Step 2, for any i, there are at most five filters (a filter is referred to by an "if" statement before placing a lot l in the set $DES(i)$ in Step 2 that will be applied to the lots in $FWL(i)$. The second filter takes $O(|M|\cdot|\Lambda|)$ time, the fifth filters take $O(|\Lambda|)$ time while the remaining filters take constant time $O(1)$. Thus Step 2 requires at most $O(|L|\cdot|M|\cdot|\Lambda|)$ time. Taking into account that Step 3 can be performed in $O(1)$ time and that $\log(|L|) \leq |M|\cdot|\Lambda|$, the complexity of Rule_Setupnum for a single machine is $O(|L|\cdot|T| + |L|\cdot|M|\cdot|\Lambda|)$.

4.3 RULE_GRASP_ASAP

The high-quality solutions provided by GRASP_opt motivatesd us to construct new rules for the simulation that mimic its logic. Referring to Deng et al. (2010), GRASP_opt was designed to uncover a diversity of good feasible solutions by randomly selecting the machine setups at the upper level in accordance with an adaptive greedy measure and then solving the resultant lower level problem to obtain the optimal lot assignments. Since ASAP is somewhat limited in how it can be customized, Rule_GRASP_asap only adopts the GRASP_opt logic for solving the upper level problem and uses a combination of filters already available to determine the lot assignments.

In ASAP, a scheduling rule contains a set of filters that is used to place lots that meet certain criteria on one of several processing lists. The customized filter, filter_GRASP, is used to assign a setup to idle machines at time 0. Essentially, it determines the "best" setup for all machines using logic similar to that used by Rule_First_setup. The major difference between Rule_GRASP_asap and all other rules is that it is run for n = 1000 major iterations and the best solution found is selected. In each inner iteration, a candidate list (CL) is constructed with all possible machine-tooling combinations and sorted in nonincreasing order according to their benefit value. Then, a restricted candidate list (RCL) is obtained by selecting the first l_{RCL} elements on CL. The length l_{RCL} is adjusted dynamically based on the quality of solutions obtained from the previous iterations.

Next, an element on RCL is randomly selected and the corresponding machine is set up with the indicated tooling. To complete an inner iteration the available resources and set of unassigned lots are updated. This process continues until each idle machine has been set up or there are no more lots that can be assigned.

Calculating benefit of a lot. Let L_0 be the set of unassigned lots and let $ben(l)$ be the benefit value associated with each lot $l \in L_0$. The latter is an approximation of the marginal improvement in the objective function if lot l is processed. The following formula is used in the calculations.

$$ben(l) = w_l + \left(w_{d_l}/C\right) \cdot \min\{n_l^{chips}, sh(d_l)\} \cdot I\{sh(d_l) > 0\} \cdot I\{d_l \in \{K \cup P \cup KP\}\} \qquad (1)$$

The first term on the right-hand side of (1), w_l, is the weight of lot l; the second term takes into account the size of the lot and its relative importance with respect to meeting key device targets. Here, the weight of device d_l contained in the lot l is represented by w_{d_l}. The ratio $\left(w_{d_l}/C\right)$ is the unit benefit value associated with the device

in lot l. The term $sh(d_l)$ represents the shortage of device d_l, which is calculated by subtracting the number of completed pieces from its target. The magnitude of the penalty reduction depends on $\min\{n_l^{chips}, sh(d_l)\}$. If $n_l^{chips} < sh(d_l)$, then all n_l^{chips} chips contained in lot l go towards reducing the penalty. Otherwise, only $sh(d_l)$ of them contribute. The term $I\{\alpha\}$ is an indicator function equal to 1 if the phrase α is "true" and 0 otherwise. For the two indicator functions, $I\{ sh(d_l) > 0\} = I\{ d_l \in \{K \cup P \cup KP\}\} = 1$ when $d_l \in \{K \cup P \cup KP \}$ and $sh(d_l) > 0$. The set K contains all the key devices, set P contains all the package devices and set KP contains all the pin package devices. Lots that contain these devices will have a benefit larger than their weight when respective shortages exist.

Building the candidate list (*CL*). The setup selected by each machine greatly affects the objective function value, so it is of interest to evaluate the benefit of a setup to see the potential gain in the objective function value when machine i in machine family j is assigned setup (j, λ). The benefit of a setup $ben(j, \lambda, L_0)$ is defined as the maximum sum of benefits of the unassigned lots in L_0 that can be processed within the planning horizon using setup (j, λ). It is computed by solving the following knapsack problem,

$$ben(j, \lambda, L_0) = \max\left\{ \sum_{l \in L(j,\lambda) \cap L_0} ben(l) z_l \; : \; \sum_{l \in L(j,\lambda) \cap L_0} \left(\sum_{m \in M(i,l,s)} \frac{n_l^{chips}}{r_{ilsm}} \right) z_l \le H_i, z_l \in \{0,1\}, \forall l \in L(j,\lambda) \cap L_0 \right\}$$

where $ben(l)$ is the benefit value for lot l calculated by equation (1), and set $L(j, \lambda) \cap L_0$ contains the lots that haven't been assigned to any machine and can be processed using setup (j, λ). The term n_l^{chips} / r_{ilsm} is the time required to process lot l on machine i with route s at pass m, and the inner summation is the time required to process all passes of lot l on machine i. The decision variables z_l, for all $l \in L(j, \lambda) \cap L_0$, are binary such that $z_l = 1$ when lot l is assigned to the machine $i \in SIM_j$ (where SIM_j is the machine family to which machine i belongs) and 0 otherwise. Instead of solving the knapsack problem exactly, a heuristic is used to find the solution. The lots $l \in L(j, \lambda) \cap L_0$ are first sorted

according to the benefit rate given by $ben(l)/\left(\sum_{m\in M(i,l,s)} n_l^{chips}/r_{ilsm}s\right)$ in nonincreasing order. Then, the lots are assigned to machine i in a greedy way until the end of the planning horizon is reached or there are no more feasible lots. The term $ben(j, \lambda, L_0)$ is the sum of benefits of the assigned lots.

This value is calculated for each setup (j, λ) defined in the route table when there is at least one machine $i \in SIM_j$ and one tooling piece required by setup λ available. Each element in CL is a triplet consisting of some $j \in SIM$, a tooling setup $\lambda \in \square(j)$, and the corresponding benefit $ben(j, \lambda, L_0)$, where *SIM* is an abbreviation for *set of identical machines* The elements in CL are sorted in nonincreasing order of $ben(j, \lambda, L_0)$. Table 4.3 gives an example of CL.

Table 4.3 Example of CL

SIM, j	Tooling setup, λ	ben(j, λ, L0)
2	1	100
2	3	90
3	2	80
1	3	70
2	2	60
1	1	50

Constructing the self-adjusted restricted candidate list. RCL is derived from CL by keeping only the top candidates. The length of RCL, l_{RCL}, has to strike a balance between solution quality and diversity. If l_{RCL} is large then it is likely to produce many inferior solutions; if it is small, many good solutions may be missed. Therefore, instead of setting l_{RCL} to a fixed value, it is restricted within the following range: $l_{RCL} \in \{2, 3,\ldots,11\}$. The value of l_{RCL} is adjusted during the GRASP iterations according to the quality of observed solutions.

Let $A = \{\alpha_1, \alpha_{2, \ldots,} \alpha_{10}\}$ be the set of considered values for l_{RCL} and let p_i be the corresponding probability of selecting α_i in A, i =1,…,10. Initially, p_i is uniformly distributed; that is,

$$p_i = 0.1,\ i = 1,\ldots,10$$

Subsequently, these probabilities are adjusted in the following way. Let ϕ^* be the best solution found in all previous GRASP iterations and let A_i be the average value of the solutions obtained for $l_{RCL} = \alpha_i$. Now, define

$$q_i = \left(\frac{\phi^*}{A_i}\right)^{\delta},\ i = 1,\ldots,10 \qquad (2)$$

as the relative performance of the algorithm when $l_{RCL} = \alpha_i$, where δ is a shape parameter. For higher values of A_i, q_i will be lower since $\phi^* \leq A_i$. Normalizing q_i's gives

$$p_i = q_i \Big/ \sum_{j=1}^{10} q_j,\ i = 1, \ldots,10 \qquad (3)$$

The probability distribution given in (2) and (3) is updated after the execution of each block of n^{block} iterations. When l_{RCL} is set to α_i yields relatively small average solutions, a higher probably p_i will be assigned to α_i. In the next block of n^{block} iterations, α_i will have a higher chance to be selected and lead to a better solution. In the implementation, we set δ = 10, A = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11} and n^{block} = 100. Also, some adjustments are made at each iteration to assure that each p_i has a finite, though small, chance of being selected.

After determining the length of RCL, it is constructed by keeping the top l_{RCL} candidates on CL. For example, if the l_{RCL} = 3, then the RCL in Table 5 is the top 3 elements in CL given in Table 4.4.

Table 4.4 Example of RCL

SIM, j	Tooling setup, λ	ben(j, λ, L0)
2	1	100
2	3	90
3	2	80

Selecting from RCL and updating. In the first inner iteration, one random setup (j, λ) from RCL is selected and an available machine $i \in SIM_j$ is configured with a tooling piece associated with setup λ. Next, the number of available machines and other resources are updated, and the set of unassigned lots L_0 is updated by removing those lots that contributed to the $ben(j, \lambda, L_0)$ calculation. The second inner iteration starts with an updated CL and RCL, and a second setup is randomly selected. These iterations stop when there is either no resources left or the set of unassigned lots $L_0 = \varnothing$.

The pseudocode for the logic contained in filter_GRASP is outlined in Figure 4.2. In the description, the following additional notation is used.

Sets and indices

L_i set of lots that contribute to the benefit calculation for setup (j,λ); $i \in SIM_j$

$\overline{L}$ set of lots are processing at time 0; $\overline{L} \subseteq L$

M_0 set of unassigned machines; $M_0 \subseteq M$

$\overline{M}$ set of machines working at time 0; $\overline{M} \subseteq M$

$\overline{T}$ set of tooling pieces required by the setups on machines $\overline{M}$; $\overline{T} \subseteq T$

T_0 set of available tooling at time 0

SPGRASP set of setups obtained by filter_GRASP

SPGRASP(i) setup obtained by filter_GRASP for machine i; $i \in M$

IFASSIGN(l) binary variable equal to 1 if lot l contributes to the benefit calculation for some setup, 0 otherwise

Setup(i) setup that is assigned to machine i

Lotassign set of lots that contribute to the benefit calculation of the setup being considered

r_{ilsm} speed of processing lot l on machine i with route s at pass m

$M(i,l,s)$ set of remaining passes when lot l is processed on machine i with route s

Procedure: Filter_GRASP (L, M, T, *SPGRASP*)

Input: Set of lots L, set of machine M, and set of tooling families T

Output: Setups *SPGRASP* and value of *IFASSIGN*(l) for $l \in L$

Step 1: //initialization

Put $L_0 \leftarrow L \setminus \overline{L}$; $M_0 \leftarrow M \setminus \overline{M}$; $T_0 \leftarrow T \setminus \overline{T}$; $L_i \leftarrow \emptyset$, $\forall\, i \in M$; *IFASSIGN*(l)=0, for $l \in L_0$;

Step 2: for (k = 1, 2,…, n) {

Compute *ben*(l), $\forall\, l \in L_0$;

Sort the lots in L_0 according to $ben(l) \Big/ \left(\sum_{m \in M(i,l,s)} n_l^{chips} / r_{ilsm} \right)$ in nonincreasing order;

while (some machine $i \in M_0$ is idle and sufficient tooling $t \in T_0$ is available) {

//construct CL

for (all feasible (j, λ) combinations) {

Calculate the benefit of the triplet (j, λ, L_0) and call it b;

Append (j, λ, b) to CL

}

Sort CL according to benefit b in nonincreasing order;

Construct RCL;

Randomly select one (j^*, λ^*) combination from RCL;

//perform the machine tooling setup

Find an available machine $i \in SIM_{j*}$ and configure it with tooling t required by setup λ^* and put $SPGRASP(i) \leftarrow (j^*, \lambda^*)$.

Greedily assign lots to machine i until no time remains; let L^* contain the assigned lots;

$IFASSIGN(l) = 1$ for $l \in L^*$, then

Put $L_i \leftarrow L^*$, $L_0 \leftarrow L_0 \setminus L^*$, and $L^* = \varnothing$

Update machine and tooling usage;

}

If mod(k, n^{block}) = 0

Update the probabilities for selecting the RCL length.

}

Figure 4.2 Pseudocode for filter_GRASP

After applying the filter_GRASP and getting $SPGRASP(i)$ for each machine i, Rule_GRASP_asap is called to determine whether the first ranked lot on $FWL(i)$ can be processed with this setup. If not, machine i will select the first ranked lot l such that all the required resources are available, the required setup has not been used by other machines in the same machine family, and the associated $IFASSIGN\ (l) = 0$. The reason for not using a current setup in the same machine family is to reduce competition for the same tooling among those machines and to avoid setting up an additional machine to process lots that could be assigned to machines already appropriately configured. The condition $IFASSIGN(l) = 0$ is included to prevent machine i from selecting a lot that has contributed to the benefit calculation for some other setup on another machine. If still no assignment is made, then the machine will select the first ranked lot that has all required

resources available. The logic associated with Rule_GRASP_asap is presented in Figure 4.3.

Rule_GRASP_asap

Input: Machine i, set of machines M, set of lots L, set of tooling families T, set of setups $SPGRASP$, value of $IFASSIGN(l)$ for $l \in L$

Output: Lot to process next on machine i

Step 0. Initialization:

At time 0, if $M_0 = \{i \in M: \lambda(i) = \varnothing\} \neq \varnothing$, then run filter_GRASP and determine $SPGRASP(i)$ for machine $i \in M_0$. For machine $i \in \overline{M}$, $SPGRASP(i) = \lambda(i)$.

For current machine i, determine $FWL(i)$, $Allavail(i)$, $DES(i)$, $M(i)$, $\lambda(i)$ and $\Lambda(M(i))$.

For each $l \in Allavail(i)$, retrieve the values of $\lambda(i,l)$.

For each l, if $IFASSIGN(l) = 1$, then Lotassign $\leftarrow$ Lotassign $\cup \{l\}$.

Step 1. Rank all lots in set $FWL(i)$ in nonincreasing order of $ben(l) \Big/ \left(\sum_{m \in M(i,l,s)} n_l^{chips} / r_{ilsm} \right)$.

Step 2. Apply multiple filters to $FWL(i)$ and place the lots that pass through these filters in $DES(i)$.

If $Allavail(i) \neq \varnothing$, then

FOR each $l \in Allavail(i)$

If $l \in \{FWL(i) : SPGRASP(i) = \lambda(i,l)\}$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

FOR each $l \in Allavail(i)$

If $l \in \{\{FWL(i) : \lambda(i,l) \notin \Lambda(M(i))\} \setminus \text{Lotassign}\}$, then

$DES(i) \leftarrow DES(i) \cup \{l\}$

Endif

ENDFOR(l)

If $DES(i) \neq \varnothing$, then

Go to Step 3.

Else

$DES(i) \leftarrow Allavail(i)$

Go to Step 3.

Endif

Step 3. Select the first lot in $DES(i)$ to assign to machine i.

Figure 4.3 Logic for Rule_GRASP_asap

Operationally, the simulation is run for n = 1000 major iterations when Rule_GRASP_asap is used to select lots from the FWL. The configuration that returns the smallest objective function value is reported as the best solution.

Complexity of Rule_GRASP_asap. Let $|M|$ be the number of machines in the factory, $|L|$ be the number of lots in WIP, $|T|$ be the number of tooling pieces, $|\Lambda|$ be the number of feasible setups and $|D|$ be the number of devices. At Step 0, as in the complexity analysis of Rule_SetupNum, the total time to determine $FWL(i)$, $Allavail(i)$, $M(i)$, $\lambda(i)$ and $\Lambda(M(i))$ is $O(|L|\cdot|T|)+ O(|M|\cdot|\Lambda|)$ after simplification. The complexity of constructing the set Lotassign is $O(|L|)$ and the complexity of finding the entries in $\lambda(i,l)$ is $O(|L|\cdot|\Lambda|)$. The complexity of running filter_GRASP is O $(|M|\cdot|T|\cdot(|\Lambda|\cdot|L|+|\Lambda|\cdot\log(|\Lambda|)$

+|M|·|T|+|L|)). Recognizing that |M|×|Λ| is dominated by |L|×|Λ| because |M| is small than |L|, the complexity of Step 0 reduces to O (|M|·|T|· |Λ|·(|L|+log(|Λ|)).

Step 1 involves the calculating the ratio and ranking of all the lots in $FWL(i)$. The ratio can be computed in $O(|L|\cdot|D|)$ and the ranking can be done $O(|L|\cdot\log(|L|))$ time, so the complexity of Step 1 is $O(|L|\cdot(|D|+\log(|L|)))$. Considering Step 2, for any i at most three filters (a filter is referred to by an "if" statement) that will be applied to lots in $FWL(i)$ before placing a lot l in the set $DES(i)$. The second filter takes $O(|L|\cdot|M|\cdot|\Lambda|)$ time, and the remaining filters take constant time $O(1)$. Thus Step 2 requires at most $O(|L|\cdot|M|\cdot|\Lambda|)$ time. Noting that Step 3 can be performed in $O(1)$ time, $O(|L|\cdot|M|\cdot|\Lambda|) \le O(|M|\cdot|T|\cdot|\Lambda|\cdot|L|)$, that $\log(|L|) \le |M|\cdot|\Lambda|$, the complexity of Rule_ *GRASP_asap* for a single machine is $O(|M|\cdot|T|\cdot|\Lambda|\cdot(|L|+\log(|\Lambda|))+|L|\cdot|D|)$.

4.4 RULE_GREEDY

Recall that filter_Greedy determines the setup to assign to each machine i. In contrast, Rule_Greedy decides which lot to select next according to the setup from filter_Greedy along with other logic. In many problems, a greedy strategy does not in general produce an optimal solution but nonetheless may yield a local optimum that may be very close to the global optimum. Although we have no way of verifying global optimality, it is of interest to see how well Rule_GRASP_asap performs when $l_{RCL} = 1$ compared to $l_{RCL} \in \{2, 3,\ldots,11\}$. Setting $l_{RCL} = 1$ greatly reduces the computational burden of the GRASP. To implement this simplification we introduce Rule_Greedy and use filter_Greedy to determine the setup for idle machines at time 0 by always selecting the top candidate on CL, i.e., $l_{RCL} = 1$. The pseudocode for the logic contained in filter_Greedy is presented in Figure 4.4.

Procedure: filter_Greedy (*L*, *M*, *T*, *SPGRASP*)

Input: Set of lots L, set of machines M and set of tooling families T

Output: Set of setups *SPGRASP* and the value of *IFASSIGN*(l) for $l \in L$

Step 1: //initialization

Put $L_0 \leftarrow L \setminus \overline{L}$; $M_0 \leftarrow M \setminus \overline{M}$; $T_0 \leftarrow T \setminus \overline{T}$; $L_i \leftarrow \emptyset$, $\forall\, i \in M$; *IFASSIGN*(l) = 0 for $l \in L_0$.

Step 2: Compute *ben*(l) $\forall\, l \in L_0$;

Sort the lots in L_0 according to $ben(l) \Big/ \left(\sum_{m \in M(i,l,s)} n_l^{chips} / r_{ilsm} \right)$ in nonincreasing order;

while (some machine $i \in M_0$ is idle and sufficient tooling $t \in T_0$ is available){

//construct CL

for (all feasible (j, λ) combinations) {

Calculate the benefit of the triplet (j, λ, L_0) and set it to b;

Append (j, λ, b) to CL

}

Sort CL according to benefit b in nonincreasing order;

Select the first (j^*, λ^*) element from CL;

//perform the machine tooling setup

Find an available machine $i \in SIM_{j^*}$ and configure it with available tooling t associated with setup λ^* and put *SPGRASP*(i) $\leftarrow$ (j^*, λ^*).

Greedily assign lots to machine i until no time remains; let L^* contain the assigned lots;

IFASSIGN(l) = 1 for $l \in L^*$, then

Put $L_i \leftarrow L^*$, $L_0 \leftarrow L_0 \setminus L^*$, and $L^* = \varnothing$

Update machine and tooling availability;

}

Figure 4.4 Pseudocode of the logic in filter_Greedy

After applying filter_Greedy and getting *SPGRASP*(*i*) for each machine *i*, Rule_Greedy begins with some machine *i* and selects the first ranked lot for which all of the required resources are available and can be processed with setup *SPGRASP*(*i*). If this fails, the same logic used for Rule_GRASP_asap is applied. Because $l_{RCL} = 1$ only one major iteration of the procedure is required to obtain a solution.

The complexity of Rule_Greedy is the same as the complexity of Rule_GRASP_asap given that the operation of selecting a random number can be implemented in $O(1)$. As such, the complexity of *Rule_ Greedy* for a single machine is $O(|M|\cdot|T|\cdot|\Lambda|\cdot(|L|+\log(|\Lambda|))+|L|\cdot|D|)$.

4.5 COMPUTATIONAL RESULTS

Testing was done using both real and randomly generated data under Windows 7 on a ThinkPad T440 laptop with a 1.60 GHz Intel core i5 processor and 4 GB of memory. The real data were provided by the Taiwan and Clark Probe AT facilities of the collaborating company. In all, we evaluated eight problem instances with the following characteristics.

Problem no. 1: Real Taiwan data with no initial setups specified for the machines

Problem no. 2: Real Taiwan data with 26 machines having an initial setup

Problem no. 3: Randomly generated data obtained by sampling the number of pieces contained in each lot and no initial setups specified for the machines

Problem no. 4: Randomly generated data obtained by sampling the number of pieces contained in each lot with 9 machines having an initial setup

Problem no. 5: Randomly generated data by keeping 10 initial setups in the Taiwan data that originally with 26 machines having an initial setup

Problem no. 6: Randomly generated data by keeping 18 initial setups in the Taiwan data that originally with 26 machines having an initial setup

Problem no. 7: Real Clark Probe data with no initial setups specified for the machines

Problem no. 8: Real Clark Probe data with some machines having an initial setup

Problem nos. 1 – 6 have 36 machines partitioned amongst 6 machine families, 284 tooling pieces from 160 tooling families, one temperature setting, and 983 lots. Problem nos. 7 and 8 each contain 136 machines from 9 machine families, 233 tooling pieces from 34 tooling families, one temperature setting, and 193 lots. According to the corresponding routes, the average processing rate is 20 parts per minute for problem nos. 1 – 6 and 342 parts per minutes for problem nos. 7 – 8. The planning horizons were 3 days and 5 days, respectively, for the two different facilities.

In the experimental design, we compared the results obtained with the enhanced GRASP (GRASP_opt) with those obtained with seven variants of the simulation: (i) basic Rule_SSU_A and Rank_HP (Sim), (ii) Rule_First_setup (First), (iii) either Rule_All_setups or Rule_First_limited (All/Lim), (iv) Rule_HotLot (Hotlot), (v) Rule_SetupNum (SetupNum), (vi) Rule_Greedy (Greedy) and (vii) Rule_GRASP_asap (GRASP_asap). The abbreviations in parentheses refer to the eight approaches evaluated below. With respect to (iii), Rule_All_setups is used for problem nos. 1 – 6 while Rule_First_limited is used for problem nos. 7 – 8. Because each machine is only set up once for problem nos. 7 – 8 according to GRASP_opt, there is no point in using Rule_All_setups. Instead, we replace it with Rule_First_limited which restricts AutoSched AP to use only the machines in the GRASP_opt solution.

Table 4.5 summarizes the built-in rules, ranking schemes, new dispatch rules, and logic used in our computational experiments. Output was divided into three categories. The first category discusses the values of the first objective function component, i.e., the weighted sum of key device shortages, and the second objective function component, i.e., the weighted throughput and the total objective value. The second category reports the

number of lots finished, the total number of steps finished, and the number of machines used. The third category includes the total number of key device shortages, and the number of changeovers. The results are discussed in the following three subsections. Additional detail is available from the authors.

Table 4.5 Summary of the different rules and rank evaluated

Rules /Rank	Description
GRASP_opt	GRASP_opt solves the AT mixed-integer program with a greedy randomized adaptive search procedure and was implemented in C++.
Rank_HP	Rank_HP (highest weight) orders the lots on the family work list according to their weight in descending order.
Rule_SSU_A	Rule_SSU_A (same setup all available) lets a machine continue processing lots that require the current setup, assuming that all the required resources are available.
Rule_First_setup	Rule_First_setup initializes the simulation with the first setup obtained with GRASP_opt until additional setups are required. At that point, machines select lots using Rule_SSU_A and Rank_HP.
Rule_First_limited	Rule_First_limited restricts the machines that the simulation can use to only those that are included in the GRASP_opt solution. This rule works in conjunction with Rule_SSU_A and Rank_HP.
Rule_All_setups	Rule_All_setups uses all the setups provided by the GRASP_opt solution so that whenever a changeover is called for, the choice is prespecified.
Rule_HotLot	Rule_HotLot is designed to reduce the shortage of key devices in a greedy way. After defining hot lots and dynamically redefining their weights, a machine will try to select hot lots first even if changeovers are required.
Rule_SetupNum	Rule_SetupNum gives priority to hot lots while using the setup frequency table obtained from GRASP_opt output to guide the setup decisions.
Rule_GRASP_asap	Rule_GRASP_asap embeds the more robust selection features of GRASP in the simulation code through customization.
Rule_Greedy	Rule_Greedy, a simplification of Rule_GRASP_asap, always picks the setup for a particular machine that gives the greatest marginal improvement in the objective function among all candidates.

4.5.1 Objective function values

The results for the first two metrics are reported in Tables 4.6 and 4.7 for the eight data sets. For problem no. 1 in Table 4.6, for example, the smallest weighted key device shortage is produced when Rule_SetupNum was used, while in Table 4.7 we see that the largest weighted throughput was achieved with Rule_All_setups. Figure 4.5 depicts the

first objective, that is, the weighted sum of key device shortages for all steps in total. In most cases, the largest shortages are associated with the basic simulation and the smallest shortages are associated with Rule_HotLot and Rule_SetupNum. The smallest number of shortages is only 20% of the largest number of shortages, on average. The good performance of these two rules is due to the fact that they were designed to process as many key device lots as possible and were hence implemented in ASAP with a focus on lot completion rather than on the overall number of setups or machines used. Rule_GRASP_asap outperformed the basic simulation by 38%. Recall that the GRASP logic allows ASAP to explore a larger portion of the feasible region by randomizing machine setups using adaptive probability distributions that are a function of solution quality. After a 1000 runs, the solution that yields the smallest objective function value with a dominant first objective function value is reported.

Also, Rule_Greedy outperformed the basic simulation model by 31% on average in minimizing the first objective. This was to be expected since it always picks the setup for a particular machine that gives the greatest marginal improvement in the objective function among all candidates. A second reason, again, is that it is driven by lot completion.

Rule_First_setup and Rule_All_setups both outperformed the basic simulation by 25% on average, primarily because they both use the setup results from GRASP_opt as input. This helped reduce key devices shortages for lots that contain multiple passes. For problem nos. 7 – 8, GRASP_opt performed better than its competitors by up to 60% in minimizing the first objective. This was due to the fact that GRASP_opt emphasizes step completion rather than lot completion and the fact that all the key devices in these instances have only one step in their route.

Table 4.6 Comparison of the first objective values

Prob. no.	Weighted key device shortages (1013)							
	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
1	7.168	3.813	3.816	1.040	0.875	4.785	3.759	3.650
2	6.705	6.584	6.801	0.925	4.152	8.542	5.833	5.833
3	5.115	4.368	4.142	1.671	1.445	4.971	4.392	3.056
4	8.214	4.284	3.994	1.580	1.354	4.262	3.833	3.000
5	3.641	3.519	3.407	0.903	0.903	4.824	3.202	3.168
6	6.874	3.519	3.407	0.914	0.903	4.897	3.615	3.197
7	2.326	2.326	2.327	2.375	2.336	2.256	2.352	2.276
8	2.221	2.221	2.264	2.225	2.225	1.250	2.266	2.225

Table 4.7 Comparison of the second objective values

Prob. no.	Weighted throughput (109)							
	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
1	1.025	1.033	1.037	1.008	1.019	0.980	0.869	0.883
2	1.030	1.031	0.923	1.017	1.015	0.966	0.914	0.914
3	0.988	0.983	0.997	0.990	0.991	0.944	0.901	0.883
4	0.908	0.994	0.998	0.988	0.990	0.957	0.921	0.933
5	1.005	1.031	1.034	1.001	1.011	0.978	0.934	0.911
6	1.005	1.025	1.025	1.010	1.009	0.955	0.964	0.958
7	1.679	1.679	1.672	1.617	1.666	1.715	1.566	1.583
8	1.763	1.763	1.733	1.719	1.719	1.799	1.618	1.697

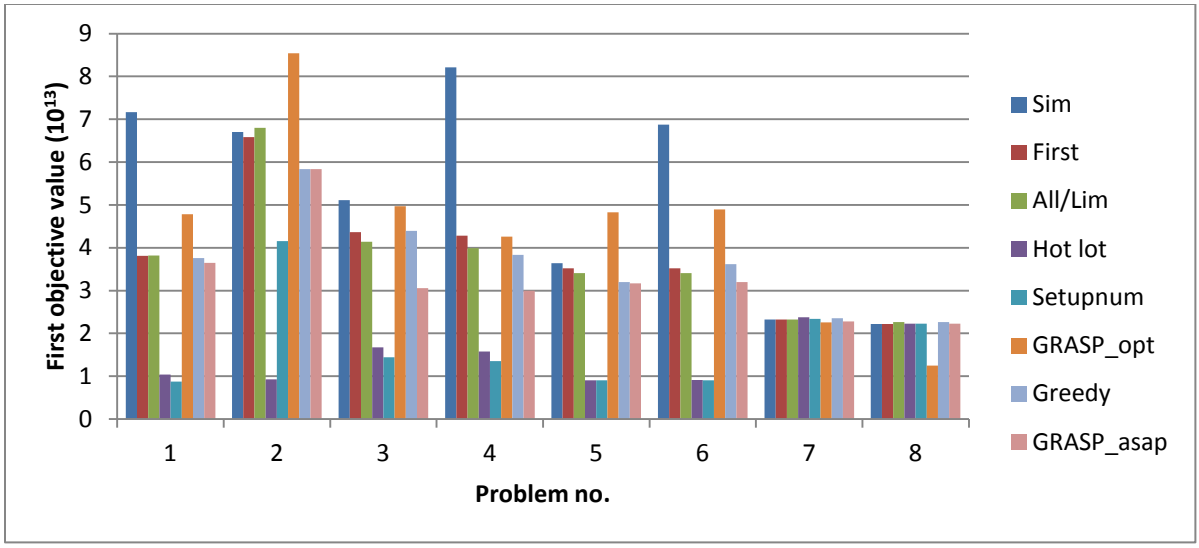


Figure 4.5 Comparison of the first objective value

The second component in the objective function, the weighted sum of all lots processed over all passes, is depicted in Figure 4.6. The basic simulation performed well on this measure since all lots are ranked in descending order by Rank_HP. The results indicate that the lots that were finished were usually those with large weights giving a larger weighted throughput value when compared to the other approaches. GRASP_opt, Rule_GRASP_asap and Rule_Greedy have no apparent relative advantage from the perspective of maximizing the weighted throughput because it is driven by the first term which dominates the second term in an absolute sense.

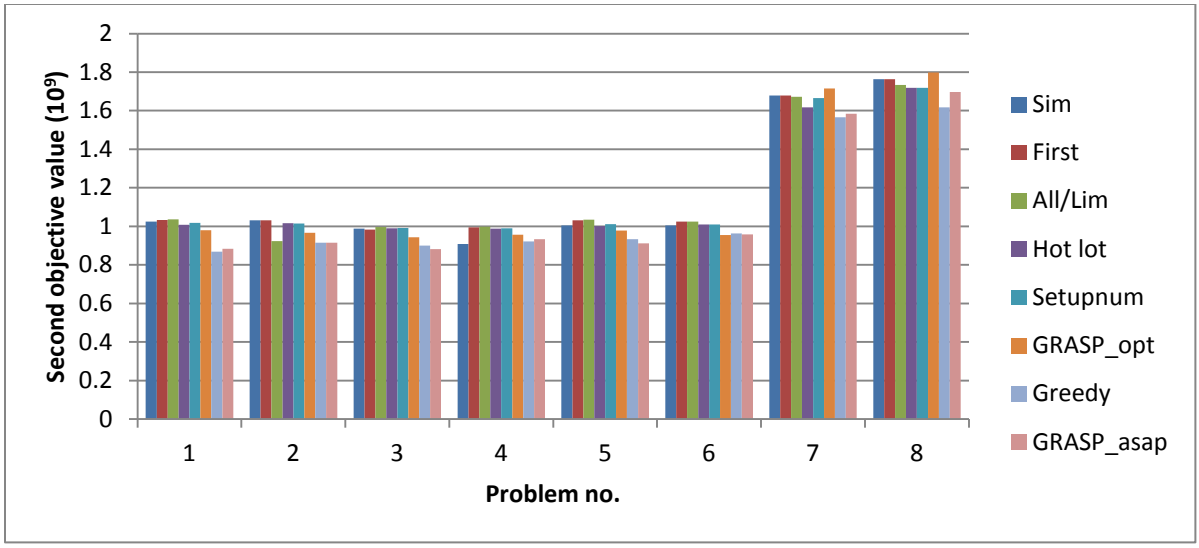


Figure 4.6 Comparison of second objective value

Of course, the total objective function value is the weighted sum of four terms including number of machines used and makespan. However, due to the dominance of the first term – key device shortages, the conclusions drawn with respect to the performance of the different approaches doesn't change. The full sets of results for the four objective function terms are tabularized in the Appendix.

4.5.2 Number of lots and steps finished and number of machines used

The statistics associated with these metrics are reported in Tables 4.9, 4.10 and 4.11, respectively, for the eight data sets. For problem no. 1 in Table 4.9, the greatest number of lots were processed when Rule_All_setups (column 4) was used. Here, a lot is said to be finished when all the remaining steps in its route are finished. Table 4.10 shows that when GRASP_opt was applied, the greatest number of steps were finished for problem no. 1. For the Taiwan data, Table 4.11 indicates that all 36 machines were used for the first six data sets. This suggests that the facility is running at capacity. For the remaining two data sets machine usage varied widely, pointing out the differences in efficiency amongst the eight approaches.

Table 4.8 Comparison of the number of lots finished

Prob. no.	Total lots finished							
	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
1	392	468	459	431	452	419	434	434
2	379	390	390	413	401	419	400	400
3	352	388	371	369	389	378	447	439
4	326	410	401	371	393	420	470	475
5	397	421	407	415	418	363	438	437
6	387	424	423	440	421	370	442	444
7	116	117	115	111	116	121	107	109
8	119	119	116	116	116	120	107	112

Table 4.9 Comparison of the number steps finished

Prob. no.	Total steps finished							
	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
1	576	645	649	620	640	688	640	640
2	570	585	585	607	593	619	619	619
3	483	520	509	507	528	647	633	646
4	457	541	537	509	532	667	681	690
5	583	609	586	607	611	627	658	659
6	574	607	613	632	616	639	657	670
7	119	120	118	114	119	123	109	111
8	126	126	123	123	123	122	114	119

Table 4.10 Comparison of the number of machines used

Prob. no.	Number of machines used							
	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
1	36	36	36	36	36	36	36	36
2	36	36	36	36	36	36	36	36
3	36	36	36	36	36	36	36	36
4	36	36	36	36	36	36	36	36
5	36	36	36	36	36	36	36	36
6	36	36	36	36	36	36	36	36
7	131	130	48	131	128	48	76	77
8	135	135	71	135	135	71	99	98

Considering the lots processed, Figure 4.7 depicts the total number of lots finished for the eight approaches. Although this metric is not one of the four objectives we were aiming to optimize because it doesn't distinguish between regular lots and key device lots, it is a rough measure of factory throughput. The basic simulation proved inferior to the seven other approaches on most of the data sets by as much as 50% down to 5%. In the former approach, machines select the first lot on their FWL with highest weight. GRASP_opt, Rule_GRASP_asap and Rule_Greedy all let the machines select those setups that produce the largest immediate benefit. Recall that Rule_First_setup and Rule_All_setups use the same first setup provided by the optimization results, so transitivity suggests that they should do better than the basic simulation in most cases. For problem nos. 7 – 8, there doesn't appear to be much difference in the number of lots processed.

Considering the steps processed, Figure 4.8 depicts the total number of steps finished by each of the eight alternatives. GRASP_opt, GRASP_asap and Greedy outperformed the five simulation variants on problem nos. 1 – 6 by 8% to 50% even though they used the same number of machines (see Table 4.10). This is not surprising since the GRASP_opt is specifically designed to maximize the weighted throughput

(steps) while minimizing the number of machine used. Both GRASP_asap and Greedy evaluate a diversity of setups before making a selection. This "look ahead" logic results in more steps completed. Considering the five simulation approaches, Rule_First_setup, Rule_First_setup, Rule_All_setups, Rule_HotLot and Rule_SetupNum all completed more steps than the basic simulation by up to 18%.

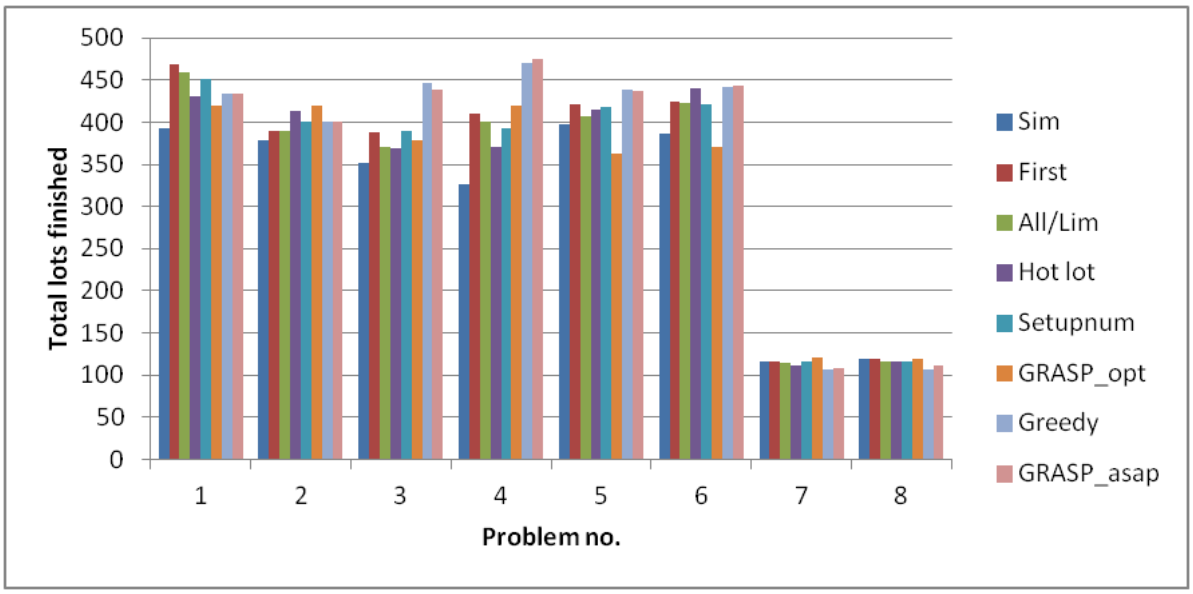


Figure 4.7 Comparison of total lots finished

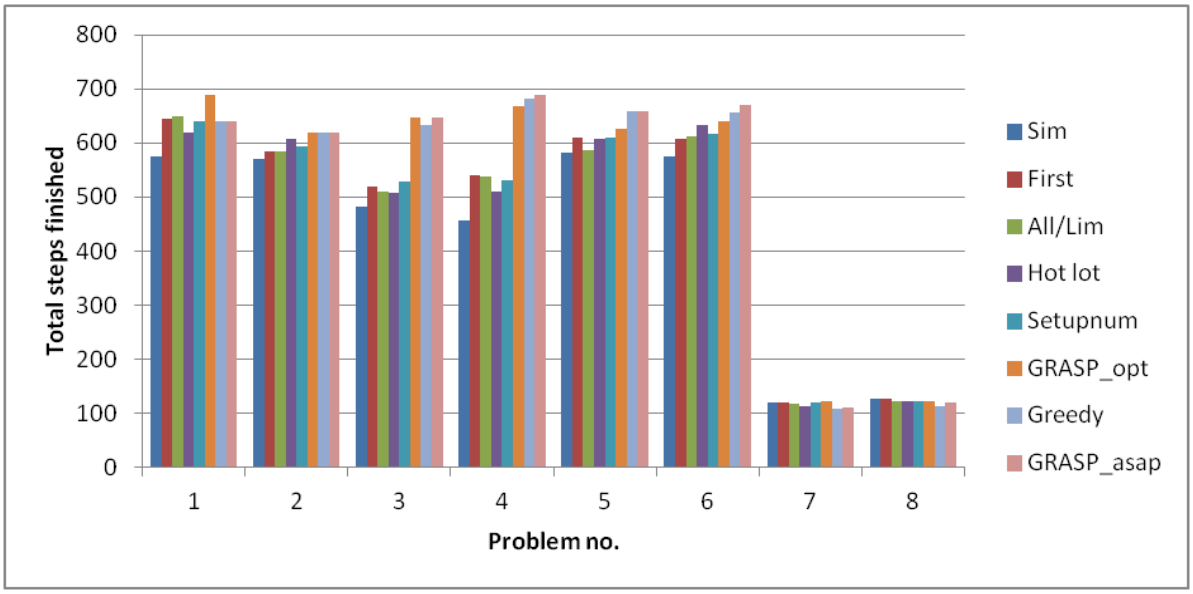


Figure 4.8 Comparison of total steps finished

The number of machines used for each problem instance and each approach is shown in Figure 4.11. For problem nos. 1 – 6, the number of lots in the queue at time 0 in the Taiwan factory is much greater than the number of machines, which are kept constantly busy. The reverse is true of instances 7 and 8. The GRASP_opt solution called for the smallest number of machines that is sometimes a third of the number of machines used in the basic simulation. This is as expected since GRASP_opt was explicitly designed to take this measure into account. Rule_First_setup_limited used the same number of machines as GRASP_opt since it takes as input the setup plan produced by GRASP_opt. Machines not included in that solution remain idle throughout the planning horizon. Rule_GRASP_asap and Rule_Greedy also set up, on average, 70% of numbers of machines that were available. Because the filter_GRASP and filter_Greedy both evaluate the benefits of feasible setups, which depend on the benefit values of the lots that can be processed within the planning horizon, lots that require more time than

available will not be assigned. Also, regular lots that require the same setup on a machine are scheduled immediately after the hot lots and before any changeovers take place.

As opposed to the approaches that exploit results from GRASP, the simulation approaches do not have forward vision and may unnecessarily set up idle machines to process lots in the upcoming time period although other machines are already set up that could process the same lots in the future. The basic simulation, Rule_First_setup, Rule_HotLot and Rule_SetupNum don't control for the number of machines used so setups occur whenever a machine becomes idle and a qualified lot appears on its work list.

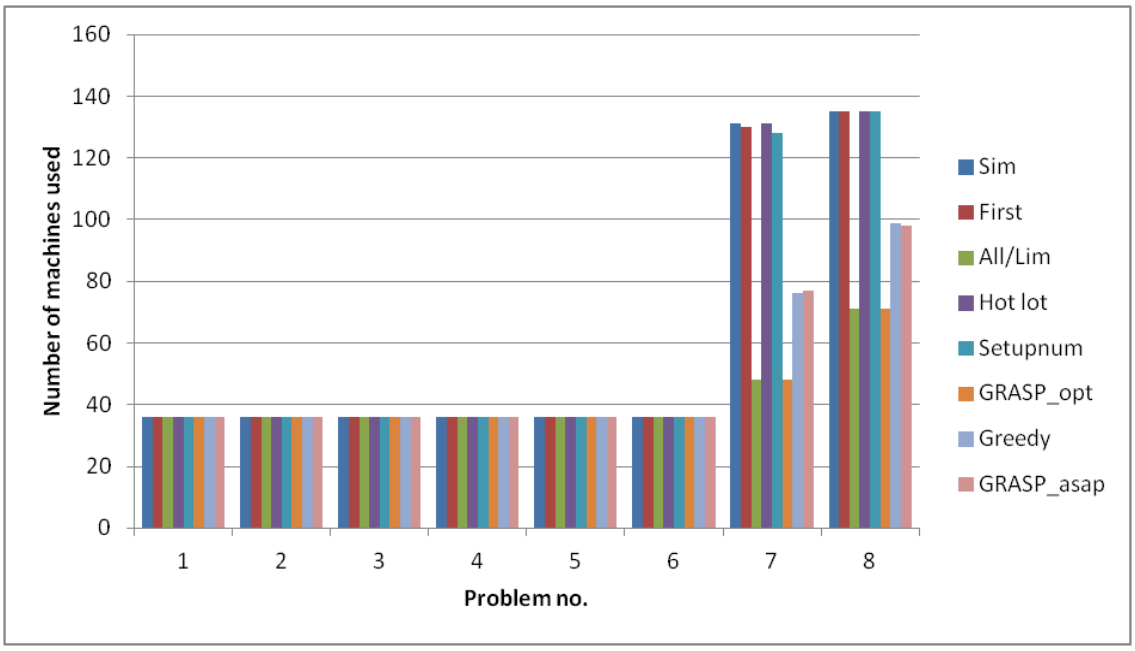


Figure 4.9 Comparison of total number of machines used

4.5.3 Number of key device shortages and number of changeovers

The results for these metrics are reported in Tables 4.11 and 4.12 for the eight data sets. Total key device shortages are calculated by first subtracting the quantity of each key device processed from its target value (only positive shortages are counted) and

then summing the results. For problem no. 1 in Table 4.11, the smallest key device shortage is produced when Rule_SetupNum was used. Table 4.12 shows that when Rule_Greedy was applied, the least number of changeovers were performed for problem no. 1.

As seen in Figure 4.10, for problem nos. 1 – 6, Rule_HotLot and Rule_SetupNum resulted in the smallest number of key device shortages among all approaches. Relative improvement ranged from 25% to 80%. This may have been expected because these two rules are designed to focus on reducing the key device shortage in a greedy manner without considering other goals such as the number of setups that may occur. Although the GRASP_opt model similarly tries to minimize key device shortages it only permits changeovers when no key or regular lot is available for processing. That is, it assigns as many lots as possible to each machine as long as its FWL and capacity are not exhausted. This logic maximizes machine utilization at the expense of the first objective when changeovers are possible. Rule_First_setup and Rule_All_setups exploit the merits of the optimization approach within the simulation by using the same first setups provided by the GRASP solution and processing as many consecutive steps in a lot's route as possible on the same machine before selecting another lot rather than considering only a single step at a time. Rule_GRASP_asap and Rule_Greedy yield, on average, 35% fewer key device shortage than GRASP_opt because they were implemented in ASAP to emphasize lot completion while GRASP_opt emphasizes step completion. Recall that a lot is considered to be finished only when all its remaining steps are finished.

Table 4.11 Comparison of the total key device shortage

Prob. no.	Total key device shortage (105)							
	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
1	2.846	1.176	1.237	0.490	0.409	1.819	1.397	1.342
2	2.765	2.704	2.790	0.459	2.065	3.545	2.591	2.591
3	1.787	1.462	1.353	0.821	0.713	2.077	1.422	1.122
4	3.327	1.421	1.280	0.776	0.667	1.795	1.008	0.941
5	1.179	1.119	1.122	0.447	0.447	1.887	1.272	1.272
6	2.788	1.119	1.122	0.452	0.447	1.907	1.297	1.240
7	0.106	0.106	0.106	0.108	0.107	0.103	0.107	0.104
8	0.101	0.101	0.103	0.101	0.102	0.058	0.103	0.101

Table 4.12 Comparison of the number of changeovers

Prob. no.	Number of changeovers							
	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
1	32	12	18	34	23	20	7	13
2	26	18	18	39	29	20	15	15
3	31	23	17	33	22	24	15	15
4	26	20	18	32	25	22	16	17
5	28	22	14	29	26	21	17	16
6	25	23	18	30	28	18	16	17
7	17	28	39	13	17	0	0	0
8	13	30	39	11	11	0	0	1

Figure 4.11 provides a graphical comparison of the number of changeovers for the eight alternatives. For problem nos. 1 – 6, Rule_HotLot produced the largest number of changeovers, which is reasonable since it calls for a changeover whenever a machine runs out of hot lots. Rule_SetupNum yields a 25% reduction, on average, compared to Rule_HotLot in most of the cases because it limits changeovers to the number in the GRASP_opt schedule. GRASP_opt did better than the basic simulation by an average of 40% with respect to the number of changeovers, but Rule_First_setup and

Rule_All_setups outperformed GRASP_opt by 12% on average for problem nos. 1-6. Rule_GRASP_asap and Rule_Greedy yielded the fewest changeovers compared to the basic simulation model on all data sets. The differences averaged 40%. To a large extent, this is a reflection of the second objective which is aimed at maximizing throughput, which is achieved in part by minimizing changeovers. For problem nos. 7 and 8, GRASP_opt, Greedy and GRASP_asap performed (at most) only one changeover between them. This was due to the relatively long average processing time of the lots with respect to the planning horizon and the fact that no lots are assigned to machines that cannot be finished within the available time.

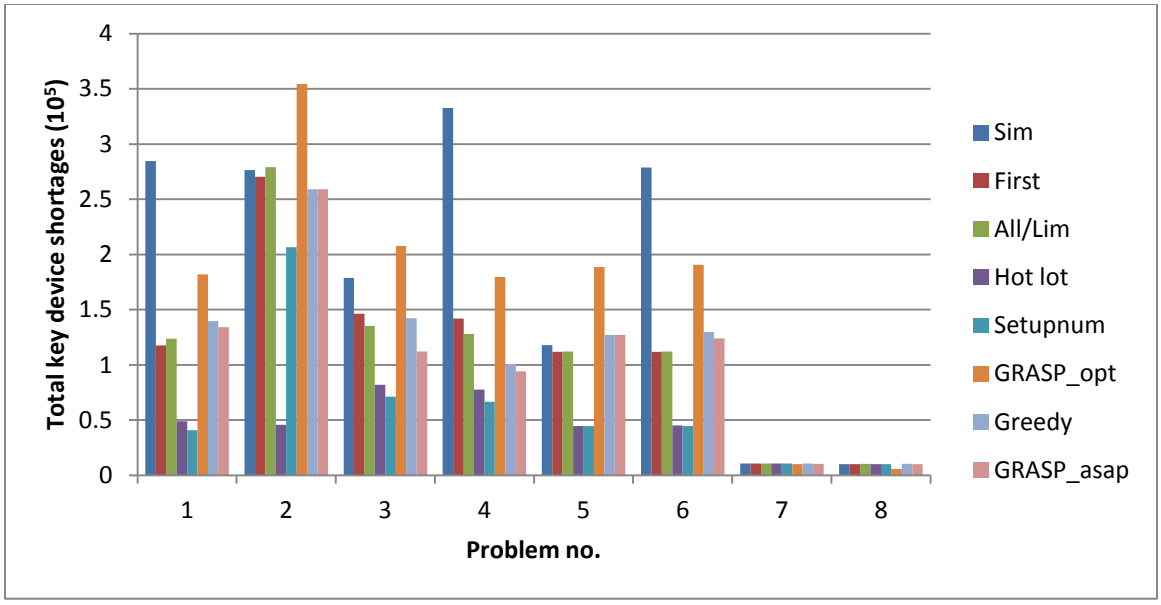


Figure 4.10 Comparison of total key device shortage

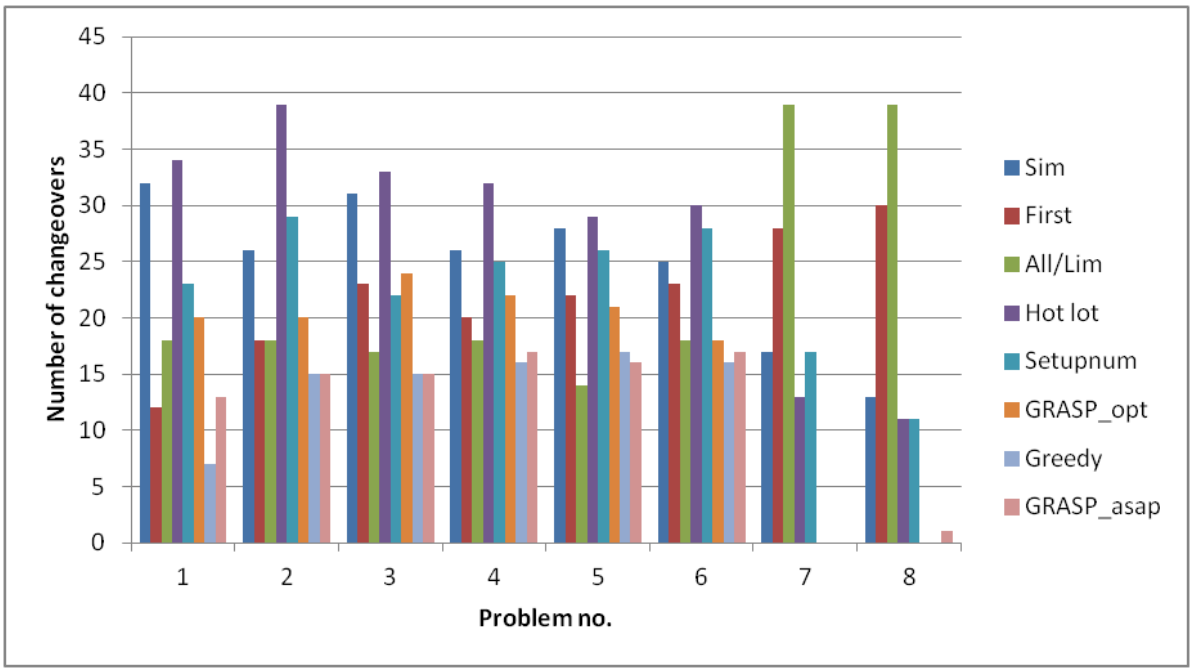


Figure 4.11 Comparison of number of changeovers

4.5.4 Comparison of key device shortages at each pass

For problem nos. 1 – 6, the number of passes for the key devices ranged from 1 to 5. Figures 4.12, 4.13 and 4.14 plot the key device shortage results after the first, second and third passes, respectively. In Figure 4.12, Rule_HotLot and Rule_SetupNum are seen to have achieved the smallest number of shortages after the first pass. This follows because they were designed to greedily process hot lots as a first priority. GRASP_opt performed comparably well since it was also deliberately designed to take this measure into account. Rule_GRASP_asap, Rule_Greedy, Rule_First_setup and Rule_All_setups, listed in the decreasing order of performance, were not as successful with respect to this measure but all outperformed the basic simulation.

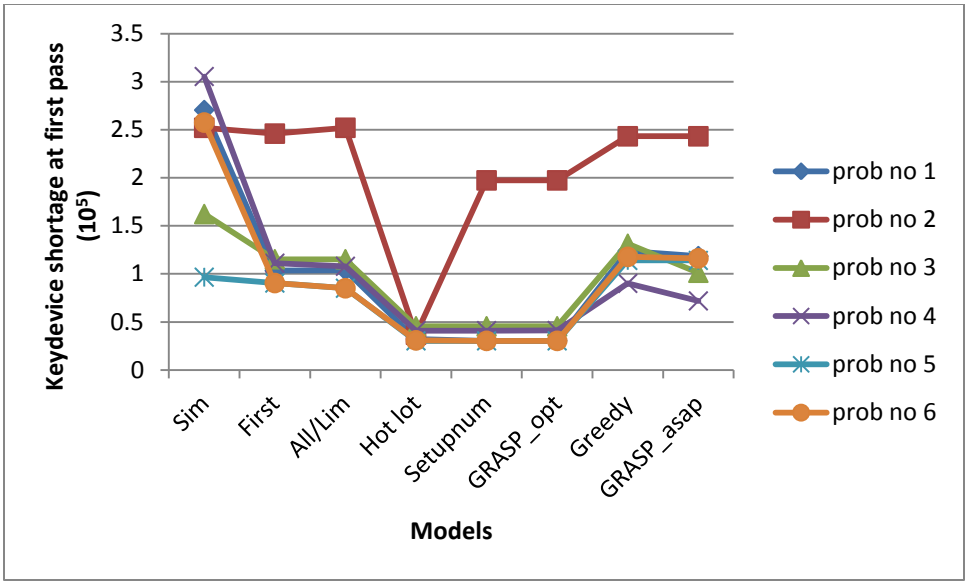


Figure 4.12 Comparison of key device shortages after first pass

As seen in Figures 4.13 and 4.14, Rule_HotLot and Rule_SetupNum still performed the best in reducing key device shortage at subsequent passes, again because these rules avoid setups as long as hot lots remain on their FWL. Rule_Greedy and Rule_GRASP_asap did better than the remaining approaches in reducing the shortages of later passes. As a reminder, if successive steps in a route of a lot can be processed on the same machine, ASAP rules favor this situation. GRASP_opt, however, aims at finishing the first pass of all lots in queue before turning to subsequent passes provided they can be done without new setups. As a consequence, it may not complete all the steps of the key devices when compared to ASAP. As we have seen, this may result in larger shortages when lots have more than one or two passes in their route.

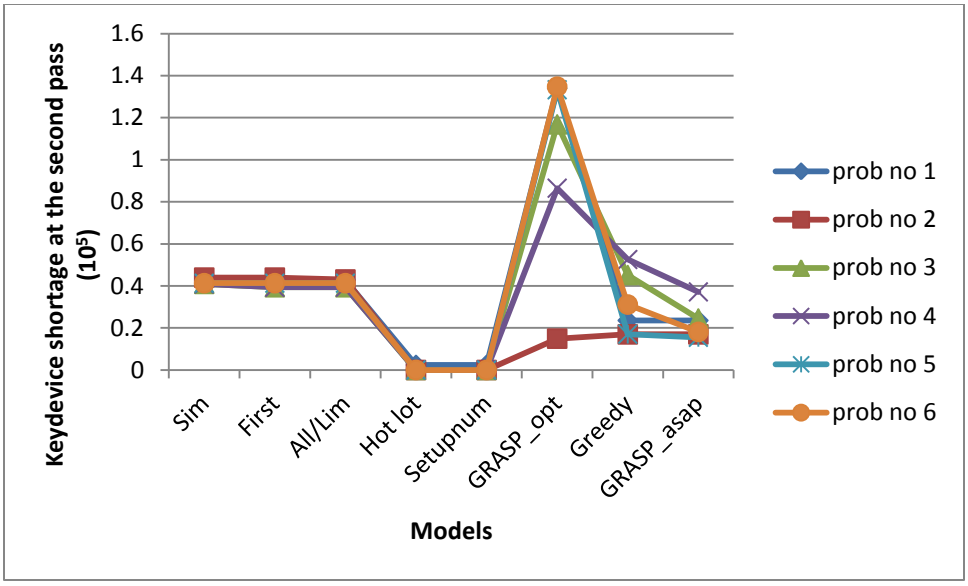


Figure 4.13 Comparison of key device shortage at second pass

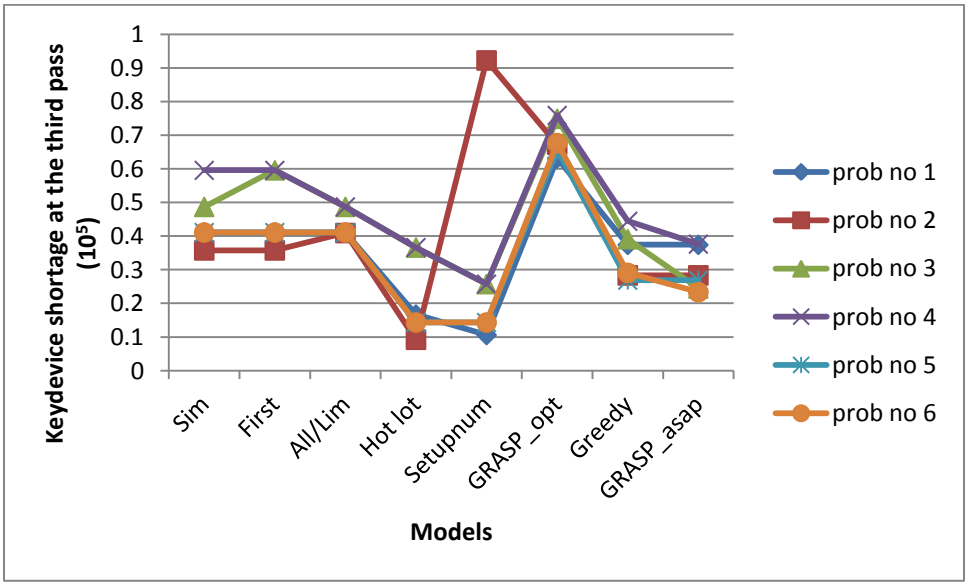


Figure 4.14 Comparison of key device shortage at third pass

4.5.5 Comparison of the results when the number of initial machines varies

In Section 4.4.1, the objective function values for eight different approaches were presented. When there are initial setups on some machines, the number and configuration

of those machines can affect system performance. To study the impact of initial setups, we considered the total objective values with respect the eight alternatives for problem nos. 1, 2, 5 and 6 associated with the Taiwan facility. Table 4.13 lists the number of machines that are already configured with tooling at the beginning of the planning horizon for these four instances. Note that the number of pieces of WIP associated with problem nos. 3 and 4 is different than for the other four data sets so they were omitted from this phase of the analysis. Here, we only want to determine how the results are affected by the number of initial setups with the other factors fixed.

Table 4.13 Parameter values for initial setups

Prob. no.	Number of machines with initial setups
1	0
2	26
5	10
6	18

Figure 4.15 plots the results. The relative performance of the eight approaches is the same when the number of machines that have initial setup varies. Rule HotLot and Rule_SetupNum performed the best, Rule_GRASP_asap and Rule_Greedy were in the second tier, followed by Rule_First_setup and Rule_All_Setups, which outperformed GRASP_opt and the basic simulation. From Figure 4.15, we can also conclude that Rule_HotLot performed consistently well with different numbers of initial machines.

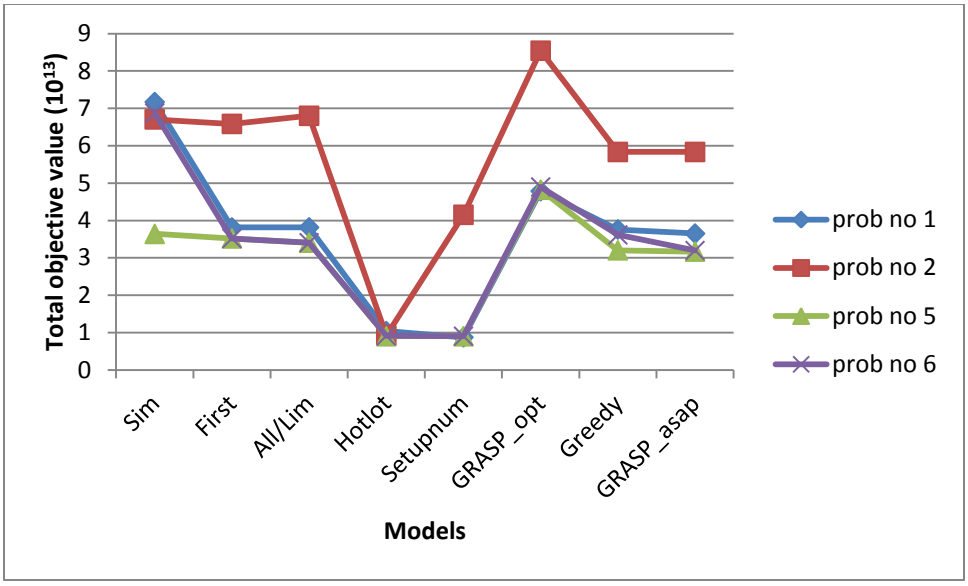


Figure 4.15 Comparison of the total objective with different number of initial setups

4.5.6 Comparison of different rules on an additional data set

For the purpose of testing the robustness of our rules, an additional data set was obtained from the Clark Probe facility of the collaborating company. The scenario included 78 machines from 11 machine families, 1204 tooling pieces from 453 tooling families, one temperature setting, and 1756 lots. From the route file, we calculated the average processing rate to be 35.5 parts per minute. The analysis was conducted for a 3-day planning horizon.

The output associated with the metrics used in Sections 4.4.1 – 4.4.3 is reported in Table 4.14 and indicates that the smallest weighted key device shortage (first objective) is realized when either Rule_HotLot or Rule_Setupnum is used, and the largest weighted throughput (second objective) is achieved with Rule_First_setup. For each of the rules, all 78 machines are used suggesting that the facility is running at capacity. With respect

to changeovers, the fewest were performed with Rule_All_setups but at the expense of the minimizing key device shortages.

Table 4.14 Results for Clark Probe data set with no initial setups

Measures	Sim	First	All /Lim	Hot lot	Setup Num	GRASP _opt	Greedy	GRASP _asap
First objective (weighted key device shortage)(10^9)	1.574	1.574	1.787	1.281	1.281	1.494	1. 315	1.315
Second objective (10^4)	2.483	2.623	2.565	2.519	2.519	2.561	2.347	2.347
Total lots finished	1343	1420	1398	1363	1363	1410	1269	1269
Total steps finished	1346	1421	1402	1367	1367	1411	1270	1270
Number of machines used	78	78	78	78	78	78	78	78
Key device shortage in pieces (10^5)	8.616	8.616	9.785	7.011	7.011	8.180	7.109	7.109
Changeovers	183	165	99	196	194	101	127	127
Third objective	78	78	78	78	78	78	78	78
Fourth objective	720	720	720	720	720	720	720	720
Total objective (10^9)	1.574	1.574	1.787	1.281	1.281	1.494	1.315	1.315

The relative performance of the various rules on the new data set parallels our experience with the original eight data sets. Overall, the new dispatch rules yield better results than the basic simulation model. Rule_Setupnum and Rule_Hotlot consistently performed best in minimizing the first objective, that is, the weighted sum of key device shortages for all steps, on all nine data sets. The ratio of the smallest to the largest first objective function values was 70% for the new data set, which is less significant than the 20% average ratio observed for the other eight data sets.

With respect to changeovers, as we observed previously, GRASP_opt, Rule_GRASP_asap and Rule_Greedy required the least number of changeovers on average, which is similar to the case here with the exception of Rule_All_setups. The same was true for Rule_GRASP_asap and Rule_Greedy with respect to the number of lots finished. For this metric, Rule_First_setup and GRASP_opt outperformed the other rules in maximizing the number of lots finished by 10% in both the new data set and problem nos. 7 - 8. Overall, the results indicate that the rules are robust.

4.6 SUMMARY AND CONCLUSIONS

This chapter presented three new dispatch rules for setting up machines and assigning lots to them in semiconductor assembly and test facilities, each exploiting metaheuristic logic. Rule_SetupNum gives priority to hot lots while using the setup frequency table obtained from GRASP_opt output to guide the setup decisions. Rule_GRASP_asap embeds the more robust selection features of GRASP in the simulation code through customization. This allowed ASAP to explore a larger portion of the feasible region at each decision point by randomizing machine setups using adaptive probability distributions that are a function of solution quality. As the iterations progress, more becomes known about the effectiveness of each machine-tooling combination. This information is used to update the probabilities so the better setups have a higher chance of being selected. After a 1000 runs, the solution that yields the smallest objective function value is reported. Finally, Rule_Greedy, a simplification of Rule_GRASP_asap, always picks the setup for a particular machine that gives the greatest marginal improvement in the objective function among all candidates.

Computational experiments were performed using eight data sets that reflect the average demand at two of the sponsoring company's AT facilities. The following insights were gained from the analysis.

- Rule_HotLot yielded the smallest total objective values but required an excessive number of changeovers.
- Rule_SetupNum achieved relatively good performance with a smaller number of changeovers. If there are sufficient workers in the factory be able to perform the changeover whenever needed, the Rule_SetupNum should be chosen over Rule_HotLot. Using Rule_HotLot, though, avoids the need to run GRASP_opt first.
- If there is only one pass required for most of the key devices, then GRASP_opt should be used due to its superior performance in minimizing key device shortages, as well as reducing the number of machines and changeovers.
- For any other scenario, the advantages of including the logic of GRASP into ASAP makes Rule_GRASP_asap a good approach to schedule AT operations, since it achieves the smallest number of key device shortage with the fewest changeovers, the least number of machines, and no dependence on GRASP_opt output.
- The performance of Rule_Greedy was dominated by Rule_GRASP_asap due to the greedy manner in which it selects setup candidates. Rule_First_setup and Rule_All_setups were outperformed by Rule_GRASP_asap, and have the disadvantage of requiring the extra step of running GRASP_opt.
- All the proposed approaches outperformed the basic simulation.

The relative performance of the different approaches was not affected by the number of machines that have an initial setup.

Chapter 5: A performance analysis of the dispatch rules on semiconductor Assembly & Test operations

This chapter presents a comprehensive performance analysis of the dispatch rules developed to solve the scheduling problem using the statistical tests and factorial design. The goals of the analysis are to discover the general order of rule performance, and investigate the impact of two factors: having initial setups and posing a setup control on the rule performance.

5.1 Literature Review

There has been a significant amount of work in developing dispatch rules for discrete parts manufacturing and semiconductor fabrication but little with regard to assembly and test. Li et.al (2013) proposed an adaptive dispatch rule whose parameters were determined reactively by the real-time scheduling information to deal with the uncertainty in semiconductor fabrication facilities. They showed that the adaptive rule performed better than commonly used rules such as Earliest Due Date (EDD) and Critical Ratio (CR) in terms of movement of WIP and bottleneck utilization. Gowling et al. (2013) developed a framework that initialized an AutoSched AP simulation model to the current state of the facility and used real-time dispatch rules for releasing lots into production within the simulation at Seagate Technology. The benefits of the integrated system included shared data modeling capabilities across products, expanded analytical capabilities, and reduced total cost of ownership. They demonstrated how different parameters of the customized dispatch policy Largest Batch First affected the average WIP and average batch size. A four phase plan was proposed to complete the integrated dispatch framework.

Chen et al. (2013) modeled a typical semiconductor manufacturing environment as multiple parallel queuing systems and formulated a mixed-integer nonlinear

programming (MINLP) model to determine the optimal dispatching policy. Three simplified cases were used to demonstrate the effectiveness of the model. For validation purposes, they also developed a numerical simulation procedure with exponentially distributed inter-arrival and service times. The simulation results were consistent with those predicted by the MINLP.

Wu et al. (2008) developed a dispatching algorithm for a make-to-order fab with machine-dedication features and setup requirements. The algorithm used ideas from line balancing in selecting lots, applied starvation avoidance principles to control the input and output patterns of bottlenecks, and followed a family-based approach for mask dispatching. The simulation experiments showed that the algorithm outperformed pervious methods with respect to on time delivery and cycle time, and did almost as well with respect to throughput on a standard benchmark instances.

Bang and Kim (2011) presented a dispatching algorithm to solve a scheduling problem in a semiconductor wafer probing facility with the goal of minimizing total tardiness of orders. A bottleneck-focused scheduling method was used in the algorithm, which means schedules for other workstations were determined based on the schedule derived for the bottleneck workstation which is determined first. Individual lot tardiness and sequence-dependent setup times were the two major factors considered. A rolling horizon method was proposed to implement the algorithm in a dynamic environment. The computational experiments showed that it outperformed the heuristic currently in use.

Lin et al. (2013) proposed two heuristics and a genetic algorithm (GA) to find nondominated solutions to multiple-objective unrelated parallel machine scheduling problems. Three criteria were of interest, namely, makespan, total weighted completion time, and total weighted tardiness. Each heuristic attempted to simultaneously minimize two of the three criteria while the GA tackled all three at once. The computational results

showed that the proposed heuristics are computationally efficient and yielded good solutions. They were outperformed by the GA, though on all measures. Saito (2007) proposed a pseudo-periodical priority dispatching (P3D) rule for dynamic allocation of WIP in mixed products semiconductor manufacturing. The P3D rule evaluated both the amount of WIP and the arrival rate of lots for each quantum, where a quantum is defined as a period during which a single type of product is processed on a machine. Results comparing P3D with first-come, first served logic, and the shortest processing time rule for simulated data with Poisson arrivals showed that P3D uniformly outperformed the other rules in terms of adjustment rate, throughput, response time, and tardiness.

Perdaen et. al (2008) simulated a reduced model of a reentrant semiconductor fab in their study of process control. A push dispatch policy (i.e., a first-buffer, first-served policy) at the beginning of the line and a pull dispatch policy (i.e., a shortest expected remaining process time policy) at the end of the line were applied. The switch from a push to a pull policy takes place at what is called the push-pull point (PPP). The results showed that (1) only moving the PPP doesn't significantly improve on policies that use a pure push or pull dispatch policy with respect to throughput, and (2) when PPP control is coupled with a CONWIP policy, performance improved fourfold.

Pfund et. al (2006) discussed scheduling and dispatching in wafer fabs, and provided a review of dispatching approaches and deterministic scheduling policies up through the mid-2000s. For the purpose of understanding the tools that are currently being used in a fab, a survey was conducted that asked questions about the types of scheduling methodologies currently used, and the limitations and needs of future generation scheduling systems. The survey results indicated that (1) many dispatching systems were aging, but they still performed well; (2) cycle time and on-time delivery were most affected by a bottleneck machine breakdown requiring that jobs had to be

placed on hold; and (3) most respondents thought that rescheduling, that is, the reevaluation and modification of dispatch decisions, should be performed periodically to better handle system disruptions. With respect to point (3), around 35% of the respondents thought it best to reschedule after every job movement while the remainder favored longer rescheduling intervals within the planning horizon such as every 8 hours.

In their review of dispatching rules, Pfund et al. first examined two commonly-used priority-based rules, and then discussed several advanced strategies including toolgroup-specific dispatching rules and full-wafer fab dispatching rule. Development and implementation efforts at three companies were also reported. In the third section of their paper, they introduced shifting bottleneck heuristic for classical job shops, and the modified shifting bottleneck heuristic for complex job shops. AutoSched was used evaluate scheduling approaches, and the results showed that the shifting bottleneck approach had the potential to improve the on time delivery performance without loss of throughput.

Li et al. (2003) developed a dispatching rule named ODDR to improve on-time delivery for semiconductor wafer fabs. ODDR considers dispatching of bottleneck machines, non-bottleneck machines, batching machines, and hot lots, that is, those lots that need to be processed to guarantee on-time delivery. To avoid starvation at bottleneck machines, WIP is controlled by placing a lower limit on its value based on WIP levels at non-bottleneck machines. Similarly, WIP at non-bottleneck machines is controlled by limiting its value to be no higher than the corresponding values at the bottleneck machines. Using simulated data, the authors showed that ODDR outperformed first-in, first-out, EDD and CR with respect to throughput, cycle time, and especially on-time delivery.

Chen et al. (2012) presented an optimal dispatching rule for a semiconductor assembly production line (ODR-SAP), which takes into account batch processors, multiple reentrant flow, and hybrid processing on the standard measures used to gauge performance. The rule was designed to incorporate policies for batch and non-batch processing with and without setup times, and the need to accommodate emergency jobs. The production process was simulated using ExtendSim, and the results showed that ODR-SAP significantly outperformed traditional dispatching rules with respect to cycle time, production efficiency, WIP, processor utilization, on-time delivery rate, and other key performance indexes.

Similar to our work, Hood and Welch (1992) conducted a statistical analysis of their results obtained from simulating a reentrant semiconductor manufacturing facility that included multiple products and multiple tool groups. They studied four main factors: distribution type and mean for the distribution of the time between interrupts, and distribution type and mean for the duration of interrupts. All possible interaction terms were considered. Response variables included the setup cycle time, preventative maintenance, and the failure-repair process. The four main factor effects along with two-way, three-way and four-way interactions were considered in a full factorial design. One of the major findings was that only the mean duration associated with interruptions had a significant negative impact on the cycle time for the failure-repair process.

Gharbi and Kenne (2000) considered a manufacturing system with multiple identical machines, random breakdowns, and repair and preventive maintenance activities. Their objective was to minimize the total cost of inventory by determining production and preventive maintenance rates for the machines. Solutions were obtained with a heuristic dispatch policy, and an experimental design was conducted to determine which factors, each with three levels, affect overall cost.

Mackulak and Savory (2001) demonstrated how simulation in combination with design of experiments can be used to compare the layout of two automated material handling systems: distributed storage versus center storage. The experimental design was used to study the impacts of the following five factors on the average delivery time: total number of tools used per day, stocker cycle time, speed of the overhead hoist vehicles, tool processing time, and the number of storage locations [each containing a single front opening unified pod per input/output port]. The delivery time was defined as the difference between the time when the lot in a stocker makes a request for a transporter to when it arrives at its destination. The conclusion was the average delivery time associated with the distributed system was strictly less than the value for the centralized system. Additional results included, for example, that increasing vehicle speed improved performance but not as large as one might expect. For a statistical design of experiments used to analyze the critical dimensions of gate poly-silicon, see Park (2004).

5.2 DESCRIPTION OF A&T OPERATIONS AND OVERVIEW OF PREVIOUS RULES

Back-end operations in semiconductor AT facilities give rise to reentrant flow and the combinatorial problem associated with machine setups and lot assignments. Each lot must be processed using different tooling in a predefined sequence of steps on one or more appropriately configured machines. Scheduling thousands of lots a day to minimize key device shortages as well as optimize any number of other objectives is a complex problem for which more advanced analytic techniques are still needed. More details of AT operations can be found in Bard et al. (2015).

The AT scheduling problem under investigation was first formulated as a mixed-integer linear program and solved by an enhanced GRASP referred as GRASP_opt in the later analysis. Preemptive goal programming logic was used to deal with the hierarchical nature of the four objective function components previously mentioned. Solutions

obtained from the enhanced GRASP included a spectrum of machine-tooling combinations and lot assignments over multiple iterations.

As a real-time alternative tool to the GRASP, Bard et al. (2015) developed a deterministic discrete event simulation model using AutoSched AP which relies on dispatch rules to schedule lots over a given planning horizon, typically two to seven days. Rule_SSU_A, is built into ASAP and lets a machine continue processing lots that require the current setup, assuming that all the required resources for that lot are available. The lots on the Family Work List (FWL), a data structure in ASAP, are ordered from the highest weight to the lowest using the feature called Rank_HP. Results obtained from Rule_SSU_A are viewed as the benchmark.

After much experimentation, we found that Rule_SSU_A as well as the other built-in rules proved ineffective relative to the GRASP so we developed several new rules using the customization feature in ASAP. The first was Rule_First_setup which initializes the simulation with the first setup obtained with GRASP_opt. When additional setups are required, machines select lots using Rule_SSU_A and Rank_HP. Rule_HotLot was designed to reduce the shortage of key devices in a greedy way. After defining hot lots and dynamically redefining their weights, a machine will try to select hot lots first even if changeovers are required. Further discussion of these dispatch rules can be found in Bard et al. (2015).

Jia et al. (2015) went a step further by combining the logic of GRASP with discrete event simulation to produce several more customized rules. In particular, Rule_SetupNum tries to limit new setups to the number provided in the GRASP_opt solution. We start by constructing a table of setup results that enumerates all the machine instances and their setup sequence over the planning horizon in the GRASP_opt solution. We then construct a setup frequency table that lists each setup and the number of times it

appears in the solution. Now, when a machine becomes free, we search the frequency table to determine which setup to select by comparing the number in the table with the number of machines on the shop floor that have the same set ups. The first setup associated with a positive difference is chosen.

The same logic embedded in Rule_HotLot is used to construct the hot lot list and to update lot priorities. Continuing, Rule_SetupNum gives priority to hot lots containing key devices while using the setup frequency table obtained from our GRASP optimizer. GRASP_asap embeds the more robust selection features of GRASP in the ASAP model through customization. This allows ASAP to uncover many good feasible solutions by randomly selecting the machine setups probabilistically based on the quality of solutions previously realized.

5.3 EXPERIMENTAL DESIGN AND ANALYSIS

The purpose of this study is to determine the relative performance of the above mentioned dispatch rules with respect to the first two hierarchical objectives; that is, (1) minimizing the weighted sum of key device shortages, and (2) maximizing the weighted throughput when different scheduling policies are used for AT operations. The first objective value is referenced as firstobj and second objective value is called secondobj in the discussion. We also study the impact of different factors on the rule performance. The first factor is binary and relates to whether or not a subset of the machines on the shop floor are configured with tooling at time zero; the second factor limits the number of setups over the planning horizon to a given value.

All computations were performed under Windows 7 on a ThinkPad T440 laptop with a 1.60 GHz Intel core i5 processor and 4 GB of memory. Testing was performed using 30 real and randomly generated data sets, where each data set consists of five separate files: the machine file identifies the machine families and the temperatures at

which each machine instance can operate; the tooling file includes similar information; the route file lists the requirements of processing each product device including the various sequence of steps, the machine family and tooling family requirements, and the temperature at which testing takes place; the WIP file contains information on each lot including its weight, the device name, the quantity of devices in the lot, the upcoming step number, and whether or not is its being processed at time zero.

In an AT facility, the daily operations vary due to the current lots in WIP, the quantity of devices in each lot, machine and tooling availability, and the configuration of each machine at time zero. In the study, the differences between the 30 data sets are: (1) the number of pieces contained in each lot in the WIP file, and (2) the initial setup assigned to each machine. To create lot of varying size, the number of devices in each lot is sampled from a normal distribution with the parameters determined from the sponsoring company's Taiwan AT facilities. More specifically, the lot size was generated from a normal distribution with mean equal to the average size of the lots that contain the same device and standard deviation equal to min{sample standard deviation of the lots with same device, 1500}. Capping the standard deviation at 1500 was done to strike a balance between diversity and too much variation. All of the 30 data sets have 36 machines partitioned among 6 machine families, 284 tooling pieces from 160 tooling families, one temperature setting, and 983 lots.

5.3.1 Scheduling rules

Table 5.1 lists the six dispatch rules included in this study. The corresponding values of firstobj and secondobj were calculated after each scheduling rule was applied to each of the 30 data sets. According to the statistics listed in Table 5.2, the average firstobj value associated with GRASP_opt is 3.4 times as large as of the average firstobj value obtained with Rule_SetupNum, and the average secondobj value associated with

Rule_First differs from the secondobj value obtained with Rule_HotLot by approximately 10^8. Thus, the overall differences among the objective function values can be substantial, depending on the dispatch rules selected.

Table 5.1 Summary of the rules studied

Rules	Description
GRASP_opt	GRASP_opt solves the AT mixed-integer program with a greedy randomized adaptive search procedure and was implemented in C++.
Rule_SSU_A (referred as Sim)	Rule_SSU_A (same setup all available) lets a machine continue processing lots that require the current setup, assuming that all the required resources are available.
Rule_First_setup (referred as Rule_First)	Rule_First_setup initializes the simulation with the first setup obtained with GRASP_opt until additional setups are required. At that point, machines select lots using Rule_SSU_A and Rank_HP.
Rule_HotLot	Rule_HotLot is designed to reduce the shortage of key devices in a greedy way. After defining hot lots and dynamically redefining their weights, a machine will try to select hot lots first even if changeovers are required.
Rule_SetupNum	Rule_SetupNum gives priority to hot lots while using the setup frequency table obtained from GRASP_opt output to guide the setup decisions.
GRASP_asap	GRASP_asap explores a larger portion of the feasible region at each decision point by randomizing machine setups using adaptive probability distributions that are a function of solution quality.

Table 5.2 Average values of firstobj and secondobj values for different scheduling rules

Rule	Average firstobj values	Average secondobj values
Sim	3.915e+13	1.070e+9
Rule_First	3.573e+13	1.077e+9
Rule_HotLot	1.608e+13	0.971e+9
Rule_SetupNum	1.559e+13	1.070e+9
GRASP_opt	5.326e+13	1.027e+9
GRASP_asap	2.860e+13	1.032e+9

Figure 5.1 contains the firstobj and the secondobj values associated with the different rules. The better performing rules should have smaller firstobj values and larger secondobj value, implying that a rule with coordinates at the upper left corner of the graph performs the best and a rule with coordinates at the lower right corner perform the worst. Accordingly, Rule_SetupNum and Rule_HotLot appear to perform best. GRASP_opt looks to be the worst in the group, and, Rule_First appears to do better than Sim. One might also argue that GRASP_asap performs better than the Rule_First because one unit on the horizontal axis is around 3000 times larger than one unit on the vertical axis. An explanation for GRASP_opt's poor showing is given after the results in Table 5.3 are analyzed.

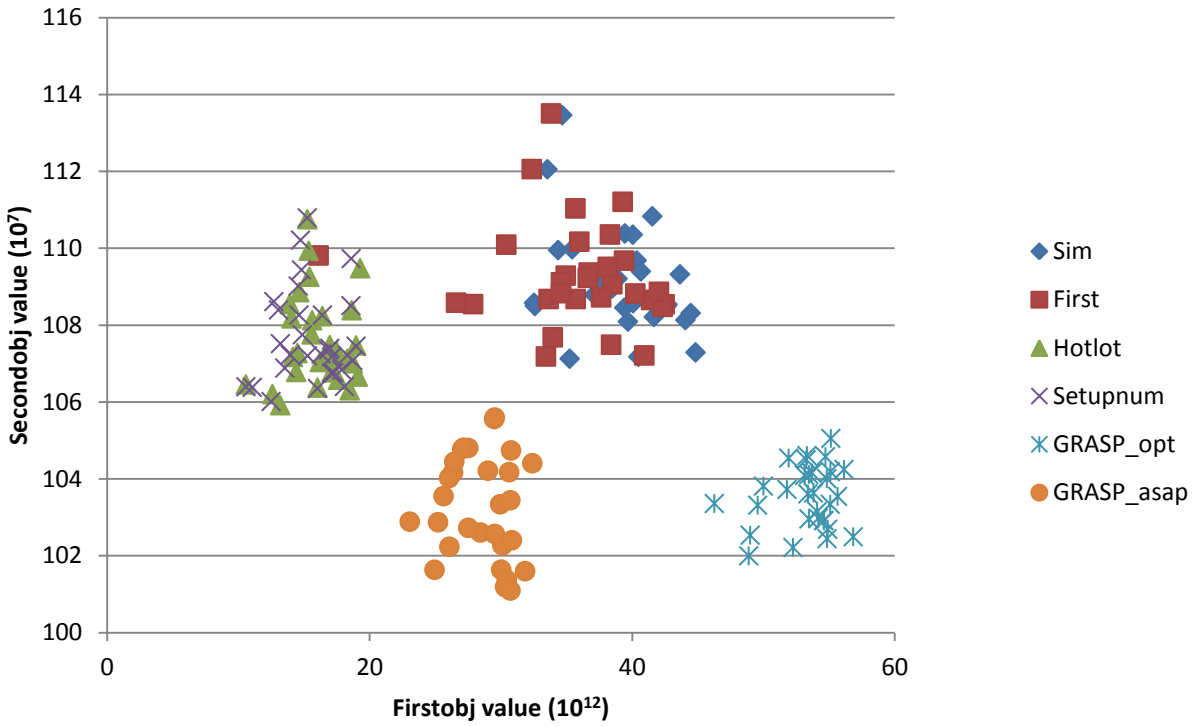


Figure 5.1 Comparison of the firstobj and secondobj values for the different rules

Since there are differences among the firstobj and secondobj values across the six rules, pairwise comparisons were conducted with the R software to gain more insights in their relative effectiveness. Table 5.3 contains the test results. The top number in each cell represents the average difference between the firstobj values for the scheduling rule in each row and each column. For example, for the cell (Sim, Rule_First), 1.67e+13 is the estimated average difference between firstobj with Sim and Rule_First . The results are based on the paired *t*-test which accounts for the Bonferroni experiment-wise error rate (Abdi 2007). The bottom number represents the corresponding p-value, that is, the probability of obtaining a result equal to or more extreme than what was actually observed, assuming that there is no difference between the firstobj values for the two rules. The threshold is selected as 0.05, so a small p-value (≤ 0.05) indicates strong evidence against the null hypothesis (i.e., there is no difference in firstobj values) and it should be rejected. Again for cell (Sim, Rule_First), the bottom entry is less than 0.05 which means that statistically there is significant difference between the firstobj values obtained with Sim and Rule_First.

The pairwise comparisons confirm the results for firstobj implied in Figure 5.1. Rule_HotLot and Rule_SetupNum perform best, but they are statistically indistinguishable. This follows because they were designed to process as many key device lots as possible and were hence implemented in ASAP with a focus on lot completion. The remaining rules can be ordered from best to worst as follows: GRASP_asap, Rule_First, Sim, and GRASP_opt. GRASP_asap performed well because it included the GRASP logic which allows ASAP to explore a larger portion of the feasible region by randomizing machine setups. Rule_First outperformed the Sim because the former included the setup results from GRASP_opt as input which helps reduce key devices shortages for lots that contain multiple passes. GRASP_opt gave the

poorest results because it is designed to maximize step completion rather than the lot completion, its original objective. In addition, the logic embedded in GRASP does not allow for changeovers if any regular lot can still be processed with the current setup, while ASAP may change the setup on a machine to process key device lots even though regular lots are available. Resetting the machine under these circumstances depends on the rule being used.

Table 5.3 Pairwise (from left to right) test results of firstobj value

t-test results	Rule_First	Rule_HotLot	Rule_SetupNum	GRASP_opt	GRASP_asap
Sim	1.67e+13	2.31e+13	2.36e+13	-1.61e+13	1.05e+13
	1.57e-05	< 2.2e-16	< 2.2e-16	0.0008	< 2.2e-16
Rule_First		2.18e+13	1.98e+13	-3.28e+13	5.29e+12
		1.55e-13	< 2.2e-16	1.8e-11	0.0005
Rule_HotLot			-2.03e+12	-3.72 e+13	-1.26e+13
			0.177	< 2.2e-16	< 2.2e-16
Rule_SetupNum				-3.77e+13	-1.31e+13
				< 2.2e-16	< 2.2e-16
GRASP_opt					2.46e+13
					< 2.2e-16

The pairwise comparison results for the secondobj value are reported in Table 5.4. A quick examination shows that the ordering of the rules from the secondobj value perspective from best to worst is also implied in Figure 5.1, namely, Rule_First, Sim, Rule_SetupNum, Rule_HotLot, GRASP_asap and GRASP_opt, with the last two rules being statistically indistinguishable. The good performance of Rule_First and Sim is due to the fact that the lots with larger weights were ranked higher on the family work list and so are processed first. Rule_SetupNum and Rule_HotLot are not as effective as the top two rules due to the fact they focus on processing lots containing key devices but not necessarily with high weights. GRASP_asap and GRASP_opt were the worst performers

since they were designed primarily to minimize key device shortages (firstobj) at each step rather than at lot completion, and not to minimize secondobj.

Table 5.4 Pairwise (from left to right) test result of secondobj value

t-test results	Rule_First	Rule_HotLot	Rule_SetupNum	GRASP_opt	GRASP_asap
Sim	-1.57e+7	1.53e+7	1.44 e+7	5.60e+7	5.92e+7
	0.002856	4.824e-14	1.211e-13	< 2.2e-16	<2.2e-16
Rule_First		1.69e+7	1.60e+7	5.76e+7	6.07e+7
		2.393e-14	5.497e-14	< 2.2e-16	<2.2e-16
Rule_HotLot			-0.91e+6	4.07e+7	4.31e+7
			4.09e-05	< 2.2e-16	< 2.2e-16
Rule_SetupNum				4.16e+7	4.40e+7
				< 2.2e-16	< 2.2e-16
GRASP_opt					2.34e+6
					0.2489

5.3.2 Effect of initial setup

One of the issues that we want to investigate is how the initial setups will affect rule performance. In AT facilities, most machines are configured with tooling at all times although they may not be constantly running lots. Figure 5.2 plots the average firstobj value versus the average secondobj value for the six different rules with and without initial setups. The postfix on the legends represents whether the value is obtained when there was initial setups. For example, the point with legend “first_no” represents the average performance of Rule_First without initial setups, and the point with legend “first_yes” denotes the average performance of Rule_First with initial setups. The point “first_no” is located northwest of the point “first_yes,” indicating that on average Rule_First without initial setups achieves a smaller firstobj value and a larger secondobj value than with initial setups. A similar situation on the graph exists for the paired points associated with Rule_Setupnum, Sim, and GRASP_asap. For GRASP_opt, although a slightly larger secondobj value is achieved, firstobj is much smaller without initial setups.

In general, most of the rules except Rule_HotLot appear to perform better, on average, when there are no initial setups. The order of rule performance appears to be preserved when there are initial setups.

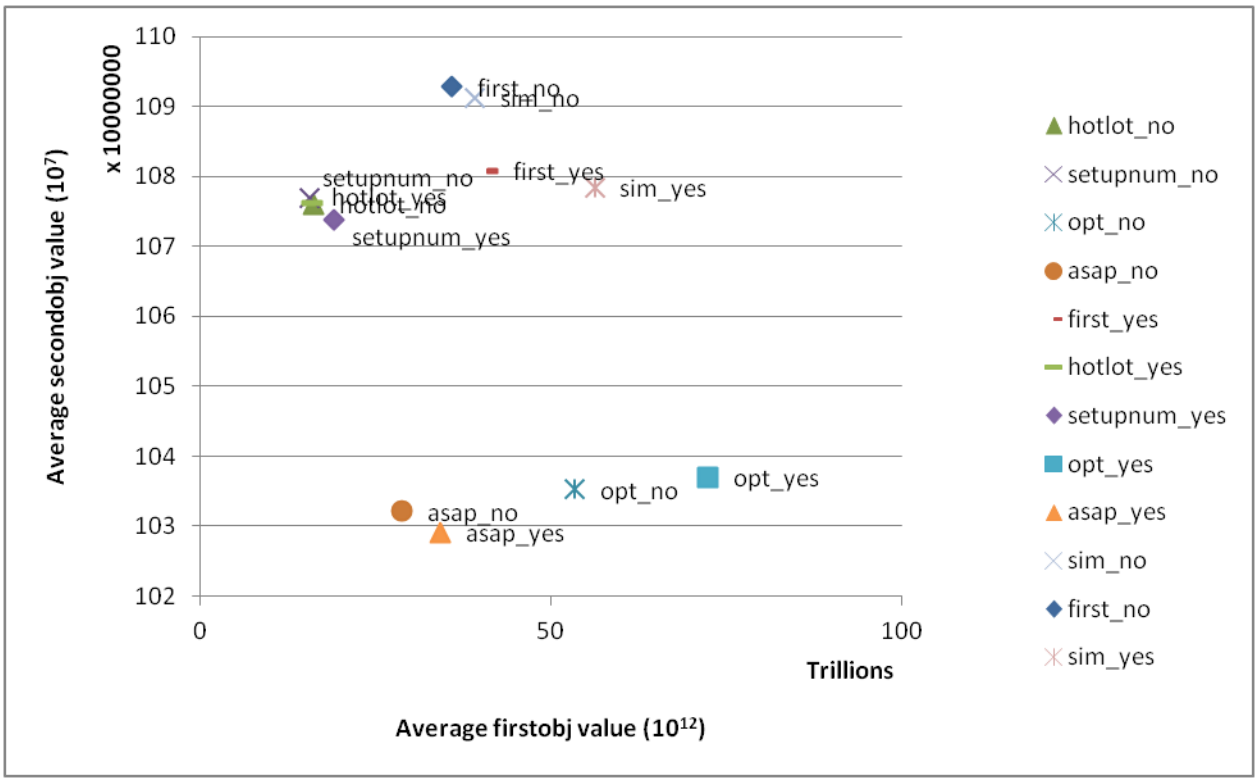


Figure 5.2 Average firstobj and secondobj value for the six rules with and without initial setups

The comparison results for the firstobj value pairs (no initial setup verses with initial setup) are given in Table 5.5. The top numbers represent the mean difference between the firstobj value obtained without initial setups and with initial setups for each schedule rule. For example, when comparing firstobj values calculated after applying Sim for the scenarios no initial setup versus initial setup, the result shows that the average firstobj value with no initial setups is less than the firstobj with initial setup by 1.70e+13. Because the p-value in the bottom cell is (much) less than 0.05 we can conclude that the

difference is statistically significant. Overall, having initial setups significantly increases the firstobj value when Sim, GRASP_opt or GRASP_asap are used as the scheduling policy. Having initial setups hurts the performance of Sim because every machine that is configured with tooling at time zero keeps processing lots that require the same setup regardless of their benefit. A changeover only occurs when the lot list is empty. More often than not, the initial setup is suboptimal with respect to firstobj. Although GRASP is able to look ahead over the planning horizon, it is similarly constrained by the initial tooling on the machines.

Table 5.5 Paired *t*-test result for firstobj: no initial setup vs with initial setup (bottom cell entry is p-value)

Rule	Sim	Rule_First	Rule_ HotLot	Rule_ SetupNum	GRASP_ opt	GRASP_ asap
No initial	-1.70e+13	-3.25e+12	2.10e+11	-3.401e+12	-1.90e+13	-5.48e+12
vs initial	2.00e-6	0.095	0.145	0.074	4.26e-07	0.011

There are no statistical differences in the results for Rule_First, Rule_HotLot and Rule_SetupNum with or without initial setups. While the differences for Rule_First, Rule_HotLot and Rule_SetupNum appear to be large in Table 5.5, statistical significance could not be established due to the large variances associated with the firstobj values.

Table 5.6 reports the comparison results for secondobj values with and without initial setups. Except when Sim is the scheduling rule, the p-values indicate that there is no significant difference between the secondobj values. Our interpretation for the slightly significant differences that occur with Sim is based on the simplicity of its logic. Similar to GRASP, Sim processes all lots available under the current setup so the lots choice

often have smaller weights than other lots in WIP. Limiting the freedom of choosing the setup on each machine after it completes a lot leads to poor performance.

Table 5.6 Paired *t*-test result for secondobj: no initial setup vs with initial setup (bottom cell entry is p-value)

Rule	Sim	Rule_ First	Rule_ HotLot	Rule_ SetupNum	GRASP_opt	GRASP_asap
No initial	1.29e+7	1.18e+7	3.04e+5	3.21e+6	-1.79+7	3.76e+6
vs initial	0.043	0.059	0.7423	0.053	0.521	0.453

5.3.3 Effect of setup control

In the facility, each setup requires a crew of several workers and may take anywhere from 0.5 to 2 hours or more to complete. If too many changeovers are called form in a solution, the workforce may not be able to handle them in a timely manner. From a practical point of view, we want to determine the degree to which imposing an upper limit on the total number of changeovers impacts rule performance. After discussing this issue with several shop floor supervisors we determined that not more than 15 setups (not including those at time zero) over a 2-day period should be allowed. When this limit is reached, only lots that are compatible with one of the existing setups will be processed. Setup control was implemented using the customization feature of AutoSched.

Figure 5.3 plots of average firstobj value versus average secondobj value of the six rules for both cases: with and without setup control. For example, the point with legend "first_no" represents the average performance of Rule_First without setup control, and the point with legend "first_yes" denotes the average performance of Rule_First with setup control. The point "first_no" is located to the northwest of the point "first_yes,"

indicating that Rule_First without setup control on average achieves a smaller firstobj value and a larger secondobj value than with setup control. The performance of the remaining pairs can be similarly found on the graph. As we can see in Figure 5.3, all of the rules appear to perform better, on average, when there is no setup control. This is to be expected and is validated below. The order of rule performance is almost the same with and without setup control.

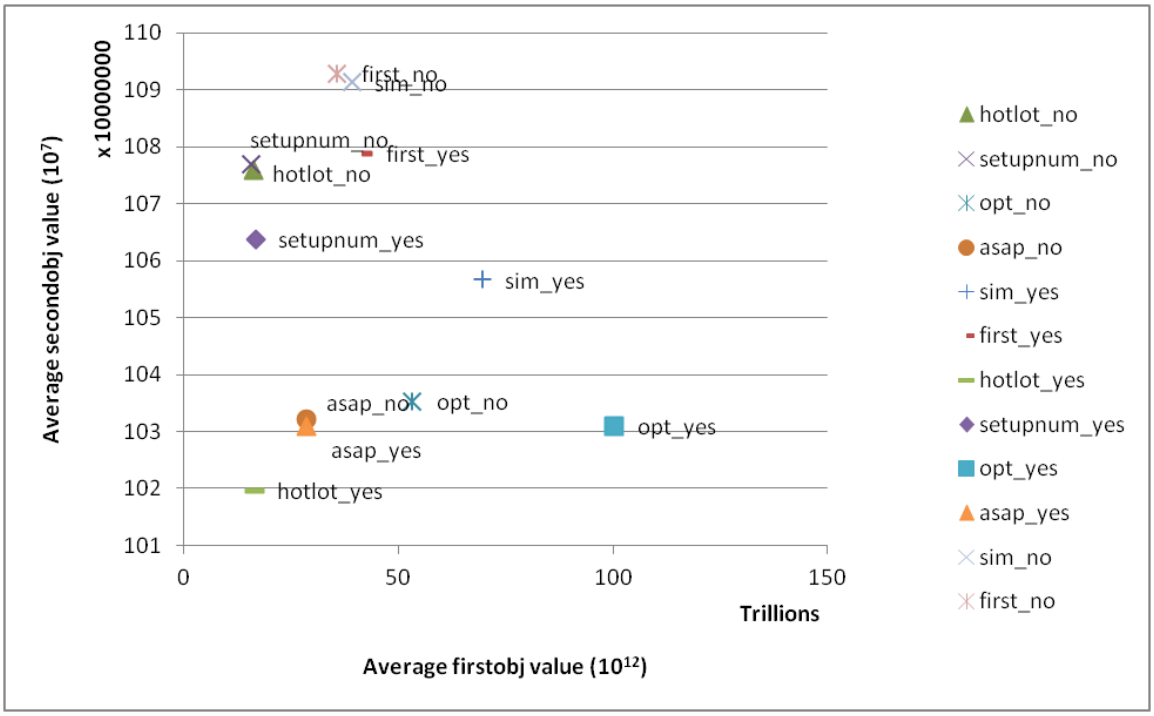


Figure 5.3 Average firstobj and secondobj value of different rules with and without setup controls

The comparison results of the firstobj values pairs (no setup control vs setup control) are reported in Table 5.7. The top number in each comparison represents the mean difference between firstobj values without setup control and with setup control for each dispatch rule. For example, when comparing firstobj values calculated after applying Sim for the two cases, the result shows that the average difference is 3.03e+13.

The p-value 5.304e-12 in the bottom cell indicates statistical significance. Of the six rules, all but GRASP_asap and Rule_HotLot provide firstobj values provide smaller firstobj values without setup control than with setup control. Poorer performance when setup limits are imposed is expected since the machines are not able to process any more lots after the changeover limit was reached. For GRASP_asap, no statistically significant differences between firstobj values were observed because the embedded logic requires each machine to process as many regular and hot lots as possible before a changeover is considered. The limit was rarely reached for the data sets examined. The fact that the difference is positive (2.29e+10) is immaterial, and is simply due to the randomness built into the methodology. For Rule_HotLot, the increase in firstobj values when setup control was imposed appeared to be large, but due to large the variances associated with the firstobj values it was not possible to establish a statistically significant difference.

Table 5.7 Paired *t*-test results for firstobj values with no setup control vs setup control

Rule	Sim	Rule_ First	Rule_ HotLot	Rule_ SetupNum	GRASP_ opt	GRASP_ asap
No setup control	-3.03e+13	-5.92e+12	-3.19e+11	-1.08e+12	-4.69e+13	2.29e+10
vs setup control	5.304e-12	1.08e-05	0.175	1.798e-09	< 2.2e-16	0.190

The comparison results of the secondobj values are given in Table 5.8. The top number in each cell again represents the mean difference between the secondobj values without setup control versus with setup control. For every scheduling rule except GRASP_asap (for reasons stated above), the secondobj value is statistically larger as expected when there is no limit on the number of permissible changeovers. Weighted throughput decreases due to the fact that machines stop processing lots when no WIP exists that can be handled by the current setups and the changeover limit is reached.

Table 5.8 Paired *t*-test results for secondobj values with no setup control vs setup control

Rule	Sim	Rule_ First	Rule_ HotLot	Rule_ SetupNum	GRASP_ opt	GRASP_ asap
No setup control	3.46e+07	1.41e+07	7.56e+07	1.32e+07	4.28e+06	1.21e+06
vs setup control	<2.2e-16	3.986e-09	4.26e-08	7.98e-12	7.60e-04	0.404

5.3.4 Full factorial design with common random numbers

To study the interactive effects that two control settings have on system performance when combined with the different dispatch rules under the two control settings, we conducted a full factorial design. The three factors and their levels are as follows:

1) Scheduling rules (GRASP_opt, Rule_HotLot, Rule_Setupnum and Rule_First). This factor is referred as "rule" in the following analysis.
2) Initial setups (no initial setup versus with initial setups). In the following analysis, this factor is named as "inisetup".
3) Setup control (no setup control versus maximum setup allowed set as 15). For further analysis, this factor is called as "control".

These factors and their levels were selected based on the results from the analysis discussed in the two previous sections. GRASP_opt was chosen because it is very close to the procedure being used by the sponsoring company in their AT facilities; Rule_HotLot and Rule_Setupnum were selected because of their top performance in reducing the firstobj value among all dispatch rules; Rule_First was included because it does the best job in maximizing secondobj.

The regression model for our full factorial design is given in Eq. (1). In the coding, the response variable is either log (firstobj value) or log (secondobj value). The variable "rule" is a categorical variable that gets converted into three dummy variables in the

regression analysis with GRASP_opt as the base case. The variables "initsetup" and "control" are also dummy variables representing initial setups and setup control, respectively. The remaining terms in the Eq. (1) are the two-way interact effects. The coefficient β_0 is the intercept term in the regression model. Since the model contains dummy variables, the intercept corresponds to the base case, which is GRASP_opt, no initial setup, and no controls. The remaining β coefficients represent the effect of each factor on the response variable. For each of the 30 data sets generated in our experimental study, 16 (= levels of "rule" × levels of "initsetup" × levels of "control" = 4×2×2) firstobj or secondobj values were calculated for the different dispatch rules and different options for "initsetup" and "control."

$$\text{Log (firstobj value or secondobj value)} = \beta_0 + \beta_1\text{*rule} + \beta_2\text{*initsetup} + \beta_3\text{*control} + \beta_4\text{*rule*initsetup} + \beta_5\text{*rule*control} + \beta_6\text{*initsetup*control} + \varepsilon \qquad (1)$$

To estimate the coefficients in Eq. (1), Ordinary Least Squares (OLS) can be used. However, OLS requires that the experimental errors ε be normally distributed and statistically independent from one observation to the next. To reduce variation and better isolate the effect of each factor, we used common random numbers (CRN) across the design points [see Kleijnen (1988) for more detail]. However, CRN complicates the analysis of simulation data in an experimental design because it induces correlation. This in turn results in more a complicated variance/covariance structure of the errors. Kleijnen (1988) proposed two possibilities in such situations: (1) continue to use OLS, and (2) switch to estimated generalized least squares (EGLS) to investigate the effects of the factors and interactions. The second possibility is more accurate due to its ability to better cope with more general variance/covariance structures.

For the normality assumption, it was necessary to take the log transformation of the firstobj and secondobj values to more closely satisfy this requirement. The Q-Q plot is a useful graphical tool to help determine whether a set of points plausibly comes from a normal distribution. In our case, when the experimental errors are normally distributed, the points should lie on a straight line. For OLS and EGLS estimators, when using firstobj as the response, the residual Q-Q plots are depicted in Figure 5.4. The upper and lower graphs on the right in the figure indicate that the normality assumption is better met after taking the log transformation, since more points appears to lie closer to the straight line. The residual Q-Q plots for the OLS and EGLS estimators when using secondobj as the response are shown in Figure 5.5. Again, the log transformation does better in meeting the normality assumption, although the differences with and without the transformation are not as pronounced.

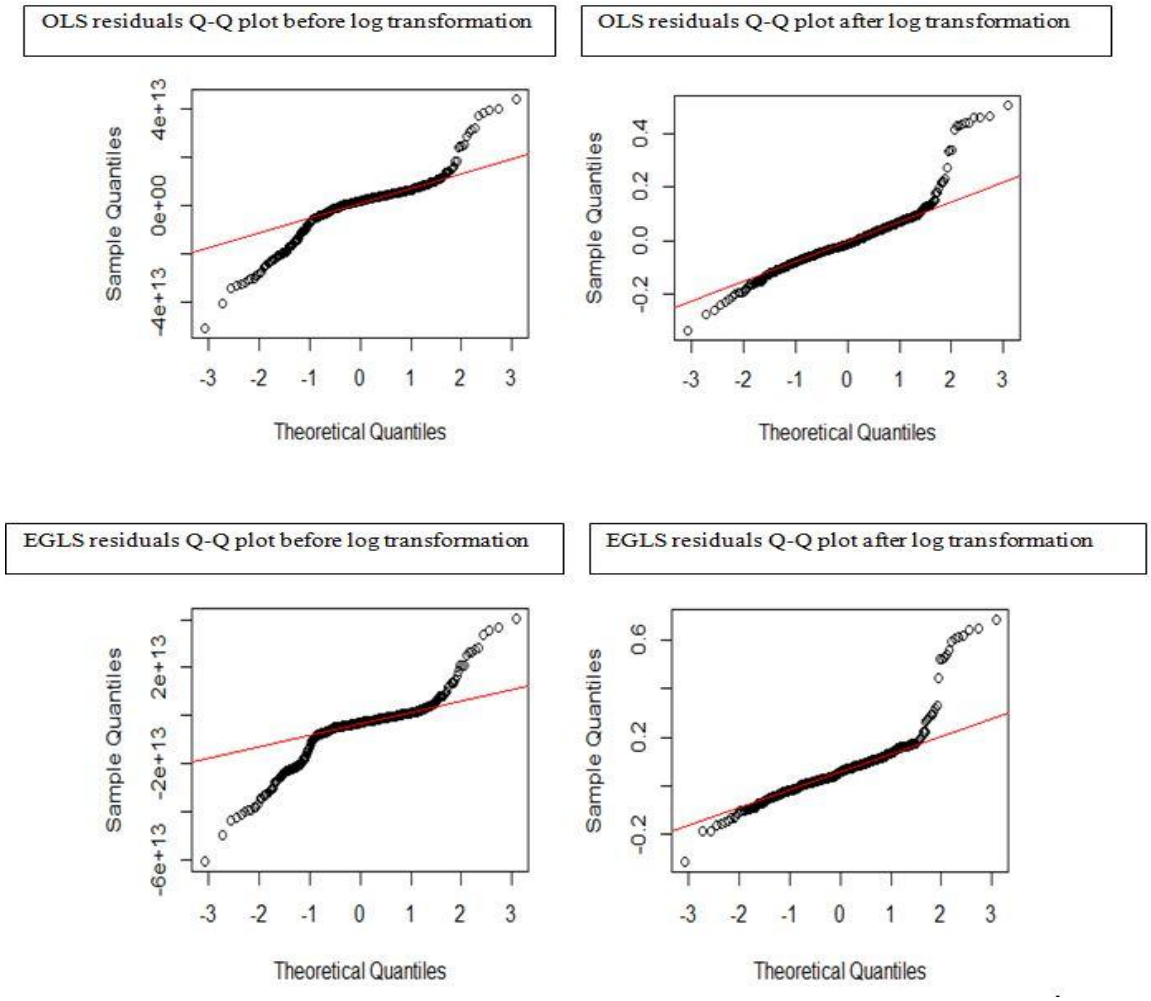


Figure 5.4 Q-Q plots for residuals of OLS and ELGS estimators before and after log transformation with firstobj as the response

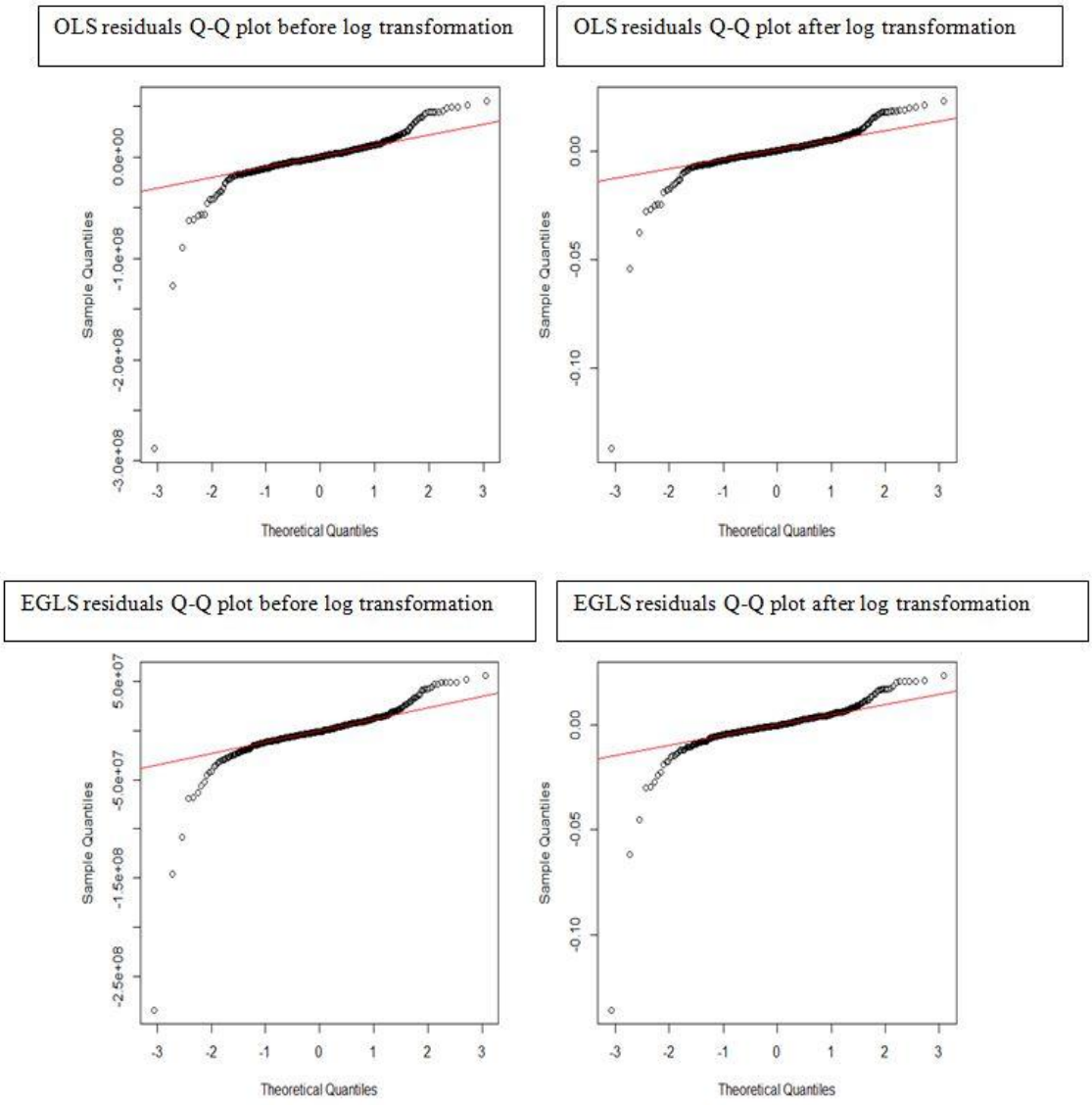
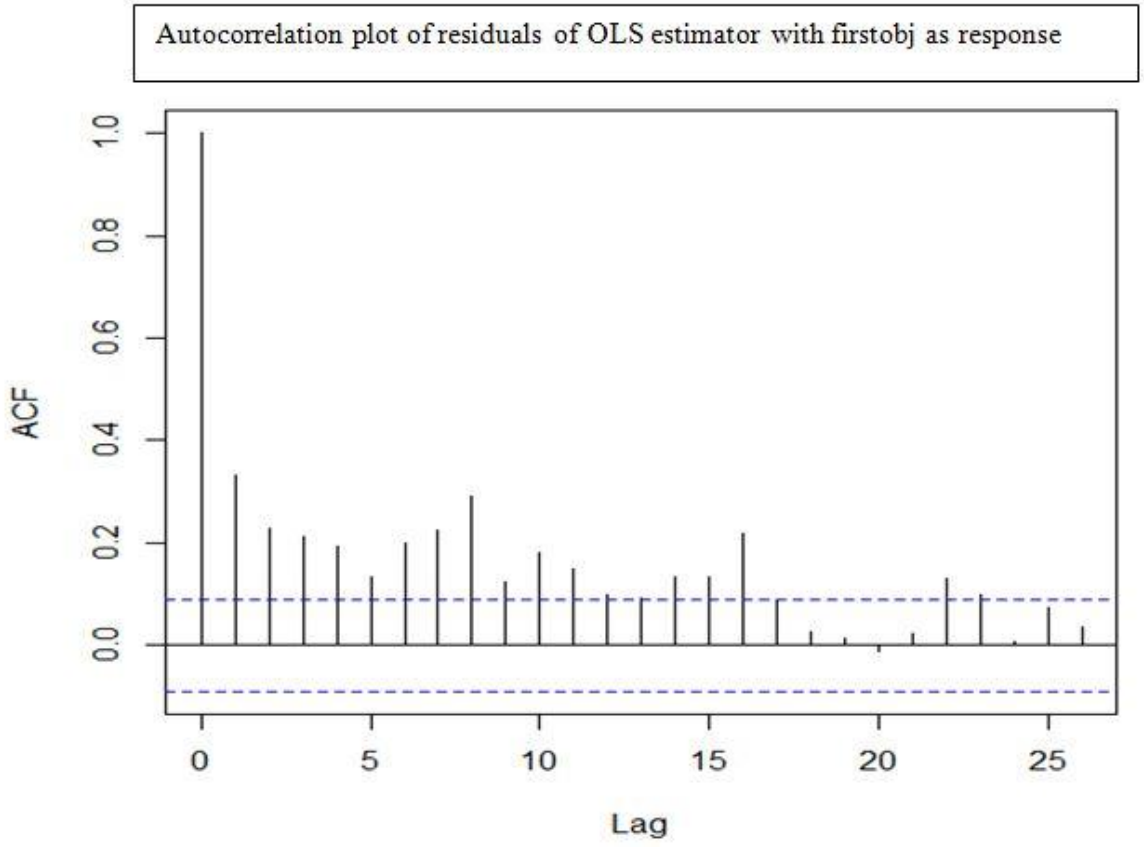


Figure 5.5 Q-Q plots for residuals of OLS and ELGS estimators before and after log transformation with secondobj as the response

Figure 5.6 plots the autocorrelation of the residuals for the OLS estimator with firstobj as response for lags between 0 and 26. The autocorrelation plot is a commonly-used tool for checking randomness for a series of data points. Randomness can be determined by computing autocorrelations for data values at varying time lags. The

autocorrelation at lag h for N observations is the correlation between the first N-h observations and the last N-h observations. If the residual errors random, the autocorrelations should be near zero for any and all time-lag separations. If not random, then one or more of the autocorrelations will be significantly different than zero. The Durbin-Watson test for lag 1 autocorrelation is used to check the assumption that the experimental errors ε in Eq. (1) are independent for the 30 data points. If the p-value of the Durbin-Watson test is less than 0.05, then we will reject the null hypothesis that there is no correlation among the experimental errors. The graph shows high autocorrelation among the residuals since most values are larger than the threshold indicated by the dotted (blue) line. This line represents the approximate 95 % confidence interval for the significant autocorrelations which is computed as $\pm 2/\sqrt{N}$), where N is the number of data points (= 16 × 30 = 480 in our data set). Statistically, the test results lead to the conclusion that the null hypothesis that there is no autocorrelation should be rejected.

Figure 5.7 contains the autocorrelation plot of the residuals for OLS estimator with secondobj as the response. No autocorrelation among residuals is evident until lag reaches 4, but according to the Durbin-Watson test for lag 1, we can't reject the null hypothesis that there is autocorrelation. However, the Breusch-Godfrey test (Godfrey 1978), which is able to assess whether autocorrelation exists for lags greater than 1, indicates that there is statistically significant autocorrelation among the residuals when the lag is 4. Taken together, the OLS assumption that the residuals should be independent is violated when either firstobj or second is set as the response. This is not surprising because correlation is being induced by CRN. We will present results for both cases as suggested by Kleijnen (1988), but our analysis will focus on EGLS estimators which are more accurate.



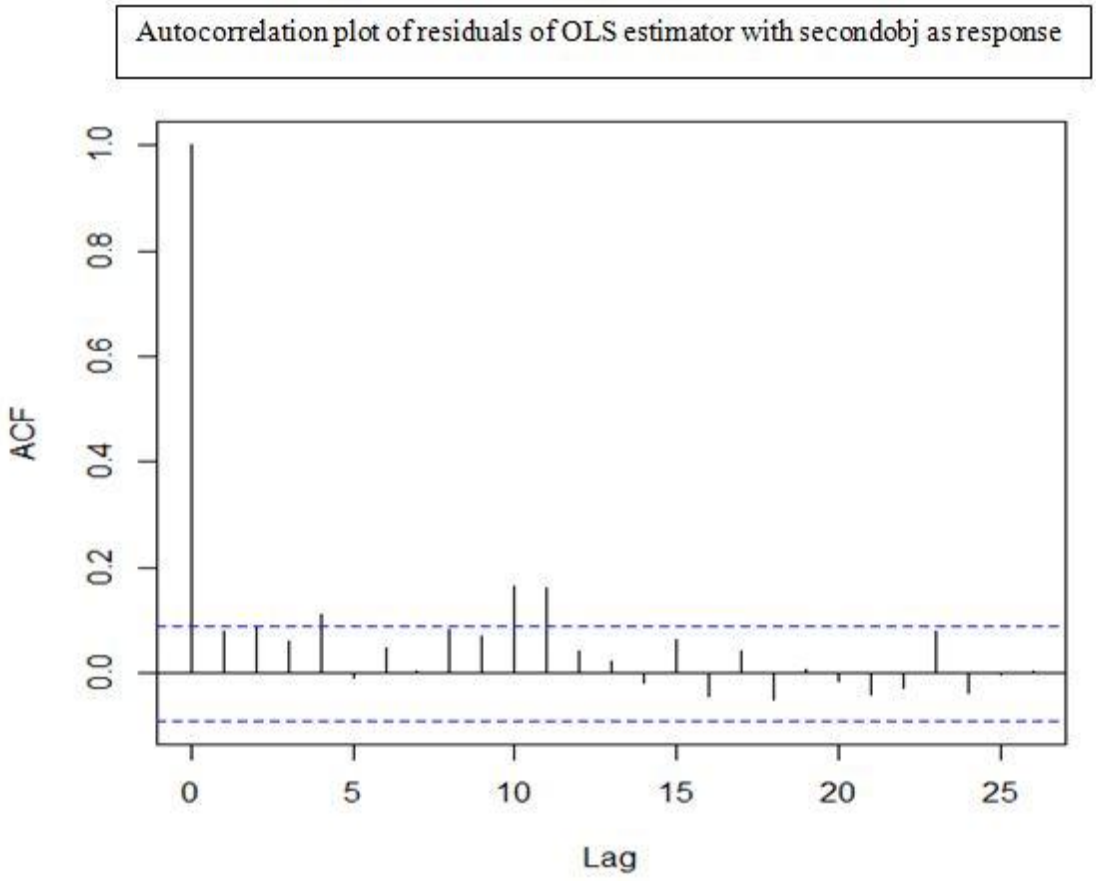
Durbin-Watson test

Data: OLSres_firstobj

DW = 1.338, p-value =3.662e-13

Alternative hypothesis: true autocorrelation is not 0

Figure 5.6 Autocorrelation plot and test of the OLS residuals with firstobj as response



Durbin-Watson test

Data: OLSres_secondobj

DW = 1.338, p-value = 3.662e-13

Alternative hypothesis: true autocorrelation is not 0

Breusch-Godfrey tests for serial correlation of order up to 2

Data: OLSres_secondobj

LM test = 6.0632, df = 2, p-value = 0.0482

Figure 5.7 continues on next page

Breusch-Godfrey tests for serial correlation of order up to 4

Data: OLSres_secondobj

LM test = 11.7588, df = 4, p-value = 0.01924

Figure 5.7 Autocorrelation plot and tests of the OLS residuals with secondobj as response

Table 5.9 contains the regression analysis results for the OLS estimators of the factor effects on the log of the firstobj values; Table 10 reports the same results for the EGLS estimators. Generally speaking, EGLS and OLS lead to the same qualitative conclusion for the estimation of main effects including Rule_First, Rule_HotLot, Rule_SetupNum, initsetup_yes and control_yes. However, the two interaction terms Rule_HotLot*initsetup_yes and Rule_SetupNum*initsetup_yes are significant in EGLS. We know that OLS can yield biased variance estimates if the covariance matrix is more general than the identity matrix (Judge et al. 1985, Section 5.5). In our particular case, OLS appears to yield a positive bias, which is the main differentiator between the two sets of results. More specifically, due to the inflated variance estimates generated by OLS, some of the interaction terms that are insignificant become significant in EGLS.

Table 5.9 Experimental design with CRN results for log of firstobj values using OLS

log(firstobj)	Estimate of β	Std. error	t value	$p > \|t\|$
Intercept	13.7673	0.0059	2316	<2e-16
Rule_First	-0.2236	0.0151	-14.81	<2e-16
Rule_HotLot	-0.5920	0.0129	-45.71	<2e-16
Rule_SetupNum	-0.5907	0.0139	-42.48	<2e-16
inisetup_yes	0.0388	0.0180	2.151	0.0200
control_yes	0.1884	0.0089	21.17	<2e-16
Rule_First*inisetup_yes	0.0189	0.0232	0.8149	0.2109
Rule_HotLot*inisetup_yes	0.0090	0.0229	0.3914	0.3492
Rule_SetupNum*inisetup_yes	0.0343	0.0272	1.2626	0.1084
Rule_First*control_yes	-0.1137	0.0145	-7.8666	5.6e-9
Rule_HotLot*control_yes	-0.1249	0.0168	-7.429	1.7e-8
Rule_SetupNum*control_yes	-0.1359	0.0125	-10.88	<2e-16
Inisetup_yes*control_yes	-0.0328	0.0121	-2.713	0.0055

Table 5.10 Experimental design with CRN results for log of firstobj values using EGLS

log(firstobj)	Estimate of β	Std. error	t value	$p > \|t\|$
Intercept	13.7141	0.0035	3967	<2e-16
Rule_First	-0.1936	0.0135	-14.33	<2e-16
Rule_HotLot	-0.5564	0.0106	-52.58	<2e-16
Rule_SetupNum	-0.5768	0.0119	-48.27	<2e-16
inisetup_yes	0.0553	0.0179	3.086	0.0022
control_yes	0.2184	0.0084	26.05	<2e-16
Rule_First*inisetup_yes	-0.0218	0.0184	-1.185	0.1228
Rule_HotLot*inisetup_yes	-0.0522	0.0180	-2.896	0.0036
Rule_SetupNum*inisetup_yes	-0.0405	0.0217	-1.867	0.0360
Rule_First*control_yes	-0.1287	0.0128	-10.04	<2e-16
Rule_HotLot*control_yes	-0.1972	0.0089	-22.22	<2e-16
Rule_SetupNum*control_yes	-0.1685	0.0097	-17.43	<2e-16
Inisetup_yes*control_yes	-0.1113	0.0096	-11.64	<2e-16

In Table 5.10, the first column lists the names of the factors whose effects are estimated, the second column contains the estimated parameters for each of the factors, the third column is the estimated standard deviation, the fourth column is the t statistic that tests the hypothesis that a population coefficient is zero when the other predictors are in the model, and the last column contains the p-value which is the observed significance

levels for the *t* value. According to the results in Table 5.10, when there is no initial setup and no setup control, the firstobj value with GRASP_opt (the base) is 5.1773e+13 (i.e., $10^{13.7141}$). From the main effects and plugging the coefficient values into Eq. (1), we can see that the firstobj value is significantly smaller when Rule_First, Rule_HotLot or Rule_SetupNum is used, where Rule_SetupNum marginally outperformed Rule_HotLot. The estimated coefficients with Rule_SetupNum and Rule_HotLot are both smaller than the coefficient with Rule_First, which is consistent with the results in Table 5.3.

The main effects for initial setup and setup control noticeably increase the firstobj value, which is as expected due to the fact that these two factors impose additional constraints on the scheduling problem. Moreover, all of the interaction effects considered except Rule_First*initsetup_yes turn out to be significant, and all have negative coefficients indicating that they can help reduce the increase in firstobj. As seen in Table 9, the negative interaction of the factor Rule_HotLot*initsetup_yes nearly cancels out the positive effect associated with the initial setup. This means that if Rule_HotLot is used for dispatching there is not a significant difference in the firstobj value when we add initial setups. This is in line with the results in Table 5.5 which indicate that there is a significant different with and without initial setups for GRASP_opt but not for Rule_HotLot. A similar conclusion can be drawn for Rule_Setupnum and Rule_First, although the difference between the negative main effect of initial setups and the interactions effects is not quite as strong. The analysis illustrates the benefit of conducting a full factorial design, which is able to detect finer distinctions in the factor effects than a pairwise comparison.

The interaction terms associated with control_yes and Rule_First, Rule_HotLot and Rule_SetupNum separately were all estimated to be negative, which matches with the paired *t*-test results in Table 5.7. According to those results, having setup control

hurts the performance of GRASP_opt (the base) the most (i.e., the largest estimated difference). For the other rules, performance is affected less severely relative to the base.

Another finding was that the interaction associated with having an initial setup and a setup control: Initsetup_yes*control_yes was estimated as -0.0023. The negative interaction helps to mitigate the increase in firstobj caused by the two corresponding main effects. One possible explanation is that the number of machines that will require a changeover within a short amount of time is relatively small when there are initial setups. Thus, setup control will be less constraining in this case so when the initial setups are aimed at processing key devices, fewer shortages are likely.

Taken together, if the facility wants to reduce the firstobj value, then Rule_SetupNum should be adopted when there is no initial setup and no setup control. If initial setups must be considered, both rules perform about the same when the interaction effects are taken into account. Rule_HotLot should be used if there is setup controls. The relatively larger interaction effect associated with Rule_HotLot*control_yes reduces part of the increase in firstobj brought about by the setup control for Rule_HotLot.

Regarding the secondobj, Table 5.11 reports the results for the OLS estimators and Table 5.12 for the EGLS estimators. Again with respect to estimators for the main effects, EGLS and OLS qualitatively lead to the same qualitative conclusions. For reasons mentioned in the analysis of the firstobj value, we will focus on the EGLS results in Table 5.12. Accordingly, the intercept means that when there is no initial setup and no setup control, the secondobj value associated with GRASP_opt is 1.035e+9 (i.e., $10^{9.0149}$). When Rule_First, Rule_HotLot or Rule_SetupNum is used for dispatching this value is significantly larger than the intercept. Rule_First achieves the greatest secondobj value which is consistent with the multiple comparison results in Table 5.4. When there is setup control, secondobj is smaller. The interaction terms between control_yes and each of the

latter three rules are estimated to have negative coefficients, and thereby reinforce the decrease of secondobj realized with setup control. The negative coefficients of the interaction terms are in line with the results in Table 5.8, which illustrate that having setup control hurts GRASP_opt (the base) the least. Rule_HotLot, Rule_First and Rule_Setupnum are more critically affected, but the degree is relative to the order in which they are listed. The interactions of these three rules and control_yes are all negative and the order of their coefficients is exactly the same as indicated in Table 5.8.

Table 5.11 Experimental design with CRN results for log of secondobj values using OLS

log(secondobj)	Estimate of β	Std. error	t value	$p > \|t\|$
Intercept	9.0161	0.0008	11407	<2e-16
Rule_First	0.0210	0.0011	19.67	<2e-16
Rule_HotLot	0.0154	0.0011	14.31	<2e-16
Rule_SetupNum	0.0169	0.0015	11.21	<2e-16
inisetup_yes	-0.0013	0.0016	-0.84	0.2044
control_yes	-0.0039	0.0010	-3.85	0.0003
Rule_First*inisetup_yes	-0.0007	0.0016	-0.46	0.3235
Rule_HotLot*inisetup_yes	0.0021	0.0017	1.26	0.1091
Rule_SetupNum*inisetup_yes	-0.0015	0.0029	-0.52	0.3039
Rule_First*control_yes	0.0011	0.0013	0.85	0.2005
Rule_HotLot*control_yes	-0.0189	0.0017	-11.27	<2e-16
Rule_SetupNum*control_yes	-0.0031	0.0024	-1.28	0.1048
Inisetup_yes*control_yes	0.0016	0.0017	0.95	0.174

Table 5.12 Experimental design with CRN results for log of secondobj values using EGLS

log(secondobj)	Estimate of β	Std. error	t value	$p > \|t\|$
Intercept	9.0149	0.0006	15629	<2e-16
Rule_First	0.0230	0.0010	24.26	<2e-16
Rule_HotLot	0.0160	0.0009	17.90	<2e-16
Rule_SetupNum	0.0012	0.0009	17.96	<2e-16
inisetup_yes	0.0012	0.0011	1.099	0.1403
control_yes	-0.0019	0.0005	-3.892	0.0003
Rule_First*inisetup_yes	0.0035	0.0012	2.959	0.0030
Rule_HotLot*inisetup_yes	-0.0009	0.0011	-0.826	0.2076
Rule_SetupNum*inisetup_yes	-0.0019	0.0013	-1.519	0.0698
Rule_First*control_yes	-0.0032	0.0009	-3.683	0.0005
Rule_HotLot*control_yes	-0.0182	0.0016	-11.18	<2e-16
Rule_SetupNum*control_yes	-0.0025	0.0007	-3.758	0.0004
Inisetup_yes*control_yes	-0.0023	0.0005	-4.408	6.5e-5

In summary, Rule_First achieves the highest secondobj value whether or not there are initial setups and setup control. However the top performance of Rule_First with respect to secondobj doesn't make it an attractive dispatch rule in the facility due to its inferior performance with respect to firstobj which is considered to be hierarchically more important.

Tables A15-A18 in the appendix contain the OLS and EGLS statistics for firstobj and secondobj values before the log transformation. Although the results in Tables 5.9 – 5.12 should be more accurate, there is almost no qualitative difference between these estimators before and after the transformation. Even the order of the estimators for the effects is preserved in each table.

5.4 SUMMARY AND CONCLUSIONS

This chapter investigated six dispatch rules developed in our previous work that can be used by shop floor personnel to configure their machines and to assign lots to each in semiconductor assembly and test facilities. Two common functions were used to

measure performance: the weighted sum of key device shortages denoted by firstobj, and weighted throughput. Multiple comparisons were undertaken by analyzing the results obtained for 30 real and randomly generated instances. Output statistics were evaluated using paired *t*-tests for scenarios with and without machine setups at time zero, and when limits were placed on the maximum number of changeovers permitted over the planning horizon. For the more comprehensive case, interactive effects were evaluated using an experimental design that applied the common random number technique to better isolate factor effects. Since minimizing firstobj was given a much higher priority than maximizing secondobj, the former is a better measure of system performance.

Accordingly, we gained the following insights from the analysis with respect firstobj.

- When there were no initial setups and no setup control, the rules from best to worst are Rule_HotLot, Rule_SetupNum, GRASP_asap, Rule_First, Sim and GRASP_opt.
- Having initial setups significantly impacts rule performance negatively when Sim and GRASP_asap are used for dispatching.
- Most of the dispatch rules perform substantially worse when there is setup control, except for GRASP_asap and Rule_HotLot.
- The full factorial design results with common random numbers show that
    (1) Rule_HotLot and Rule_SetupNum achieve the smallest firstobj values compared to GRASP_opt and Rule_First which is consistent with the paired *t*-test results.
    (2) Rule_SetupNum should be used when there is no setup control and Rule_HotLot should be selected when there is setup control.

(3) Rule_First performs the best in maximizing secondobj with and without initial setups and setup control.

(4) Both initial setups and setup control hurt system performance for all rules; however, statistically significant interaction terms associated with setup control and Rule_First, Rule_HotLot and Rule_SetupNum separately are able to mitigate part of the negative resulting from setup control.

(5) The interaction between initial setup and setup control turns out to be negative and statistically significant. One possible reason is that the setup control limit does not come into play when there are initial setups with respect to the case when there no initial setups for the given data sets. This interaction terms reduce the increase in firstobj related to the corresponding main effects.

Currently, the sponsoring company is running GRASP_opt in most of its AT facilities and has developed an plan to integrate it with the ASAP model. The goal is to use the same input files for both approaches, but this has not yet been achieved. The main stumbling block is the inability of their database system to reliably generate accurate WIP and machine setup files. Neither the GRASP or ASAP is sufficiently robust at this time to handle contradictory data elements in these files.

Appendix A1: Flow diagram of GRASP_OPT

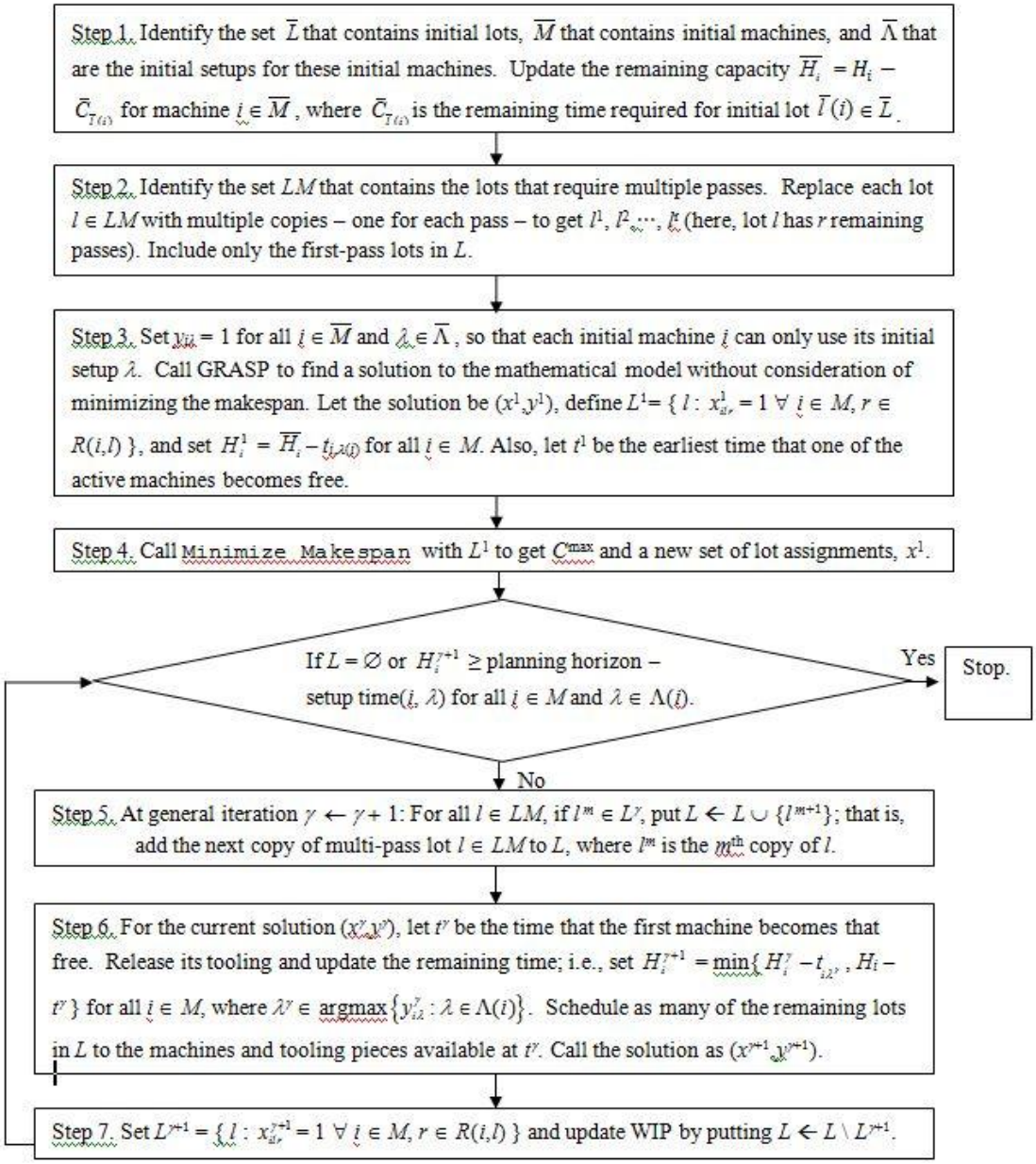


Figure A1. Flow diagram of the enhanced GRASP_opt

Appendix A2: Algorithm for Basic Simulation

The logic followed by AutoSched AP is to rank all the lots on the family work list for each machine and then try to assign them sequentially in accordance with the specified scheduling rule. The machine families and the machines within each family are arrayed alphabetically and examined in that order. In the presentation of the basic algorithm it is assumed that all necessary tooling is available so there is no waiting required. In the description, lot ranking and machine selection rules are generic. The built-in rules are given in Section 5 and our customized rules are given in Section 6.

Notation

MF	set of machine families in the facility; $i \in MF$
FWL(*i*)	set of lots that can be processed on machine family *i*
Allavail(*i*)	set of lots on *FWL*(*i*) with the required machine and tooling are available; *Allavail*(*i*) $\subseteq$ *FWL*(*i*); $l \in$ *Allavail*(*i*)
Idle(*i*)	set of machines in machine family *i* that are currently idle; $j \in$ *Idle*(*i*)
P(*i*,*l*)	processing time of lot *l* when processed using machine family *i*
T(*i*,*l*)	set of currently available tooling pieces that are required to process lot *l* on machine family *i;* $k \in T(i,l)$
Simbegin	beginning of the planning horizon
Simend	end of the planning horizon

Algorithm_Simulation

Step 0. Initialization at time $t \in$ [*Simbegin*, *Simend*]:
FOR each $i \in MF$, get the values of *FWL*(*i*), *Allavail*(*i*) and *Idle*(*i*).
FOR each $l \in$ *Allavail*(*i*), get the values of *P*(*i*,*l*) and *T*(*i*,*l*).

Step 1. Rank all of the lots in *FWL*(*i*) according to the rule specified in the station file.

Step 2. Selection at time $t \in$ [*Simbegin*, *Simend*]:
FOR each $i \in MF$ \\ Machine families are selected alphabetically.
WHILE *Idle*(*i*) $\neq \emptyset$
FOR each $j \in$ *Idle*(*i*) \\ Machines are selected alphabetically.
WHILE *Allavail*(*i*) $\neq \emptyset$
Machine *j* selects lot *l* according to the rule specified
Tooling piece $k \in T(i,l)$ is installed on machine *j*
FWL(*i*) $\leftarrow$ *FWL*(*i*) \ {*l*}
Allavail(*i*) $\leftarrow$ *Allavail*(*i*) \ {*l*}
Idle(*i*) $\leftarrow$ *Idle*(*i*) \ {*j*}
T(*i*,*l*) $\leftarrow$ *T*(*i*,*l*) \ {*k*}
If $t + P(i,l) <$ *Simend*
At time $t \leftarrow t + P(i,l)$
Idle(*i*) $\leftarrow$ *Idle*(*i*) $\cup$ {*j*}
T(*i*,*l*) $\leftarrow$ *T*(*i*,*l*) $\cup$ {*k*}
Endif

ENDWHILE
ENDFOR
ENDWHILE
ENDFOR

Figure A2.Algorithm of basic simulation

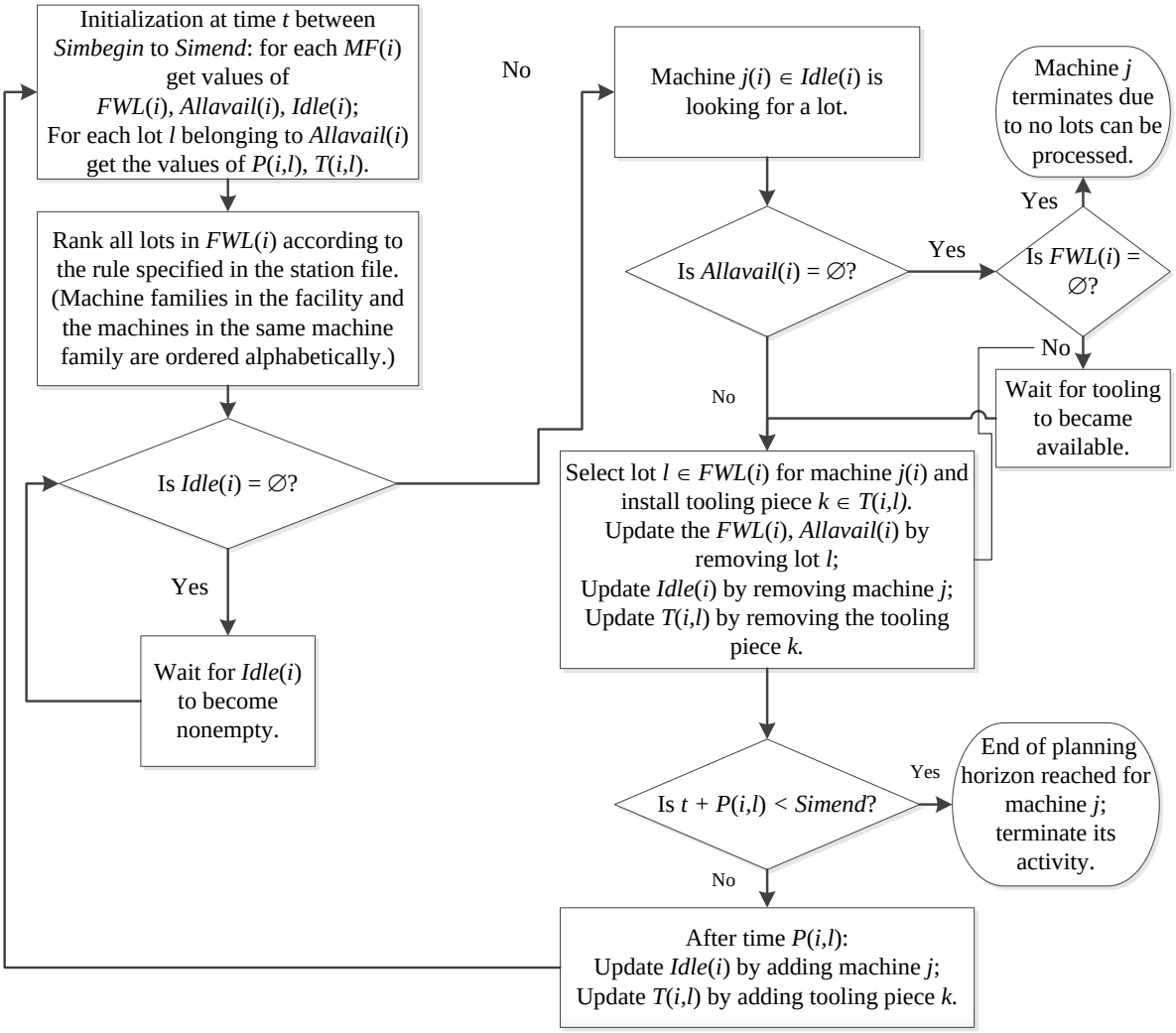


Figure A3. Flow diagram of the AutoSched AP simulation model

Appendix A3: More computation results when comparing different dispatch rules

Table A1. Results for Taiwan 1 data set without initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	392	468	459	431	452	419	434	434
Total steps finished	576	645	649	620	640	688	640	640
Key device shortage in pieces (10^5)	2.846	1.176	1.237	0.489	0.410	1.819	1.397	1.342
First objective (weighted key device shortage) (10^{13})	7.168	3.813	3.816	1.040	0.875	4.786	3.759	3.650
Changeover	32	12	18	34	23	20	7	13
Number of machines used	36	36	36	36	36	36	36	36
Second objective (10^9)	1.025	1.033	1.037	1.008	1.019	0.980	0.869	0.883
Third objective	36	36	36	36	36	36	36	36
Fourth objective	71.2	71.3	71.3	71.3	71.3	71.59	71.5	71.5
Total objective (10^{13})	7.168	3.813	3.816	1.040	0.875	4.785	3.759	3.650

Table A2. Sum of shortages for key devices at each pass for Taiwan 1 data set without initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Shortage after first pass (10^5)	2.704	1.035	1.038	0.323	0.303	0.303	1.239	1.184
Shortage after second pass (10^5)	0.428	0.428	0.428	0.024	0.024	1.340	0.236	0.236
Shortage after third pass (10^5)	0.411	0.411	0.411	0.167	0.106	0.632	0.374	0.374

Table A3. Results for Taiwan 1 data set with initial setups (26/36 of the machines have initial setups)

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	379	390	390	413	401	419	400	400
Total steps finished	570	585	585	607	593	619	619	619
Key device shortage in pieces (10^5)	2.765	2.704	2.790	0.459	2.065	3.545	2.591	2.591
First objective (weighted key device shortage) (10^{13})	6.705	6.584	6.801	0.925	4.152	8.542	5.833	5.833
Changeover	26	18	18	39	29	20	15	15
Number of machines used	36	36	36	36	36	36	36	36
Second objective (10^9)	1.030	1.031	0.923	1.017	1.015	0.966	0.914	0.914
Third objective	36	36	36	36	36	36	36	36
Fourth objective	7.13	7.09	7.09	7.15	7.15	7.20	7.15	7.15
Total objective (10^{13})	6.705	6.584	6.801	0.925	4.152	8.542	5.833	5.833

Table A4. Sum of shortages for key devices at each pass for Taiwan 1 data set with 26 initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Shortage after first pass (10^5)	2.520	2.460	2.520	0.367	1.973	1.973	2.432	2.432
Shortage after second pass (10^5)	0.440	0.440	0.431	0	0	0.149	0.169	0.169
Shortage after third pass (10^5)	0.358	0.358	0.409	0.092	0.923	0.672	0.284	0.284

Table A5. Results for Taiwan 2 data set without initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	352	388	371	369	389	378	447	439
Total steps finished	483	520	509	507	528	647	633	646
Key device shortage in pieces (10^5)	1.787	1.462	1.353	0.821	0.713	2.077	1.422	1.122
First objective (weighted key device shortage) (10^{13})	5.115	4.368	4.142	1.671	1.445	4.971	4.392	3.056
Changeover	31	23	17	33	22	24	15	15
Number of machines used	36	36	36	36	36	36	36	36
Second objective (10^9)	0.988	0.983	0.997	0.990	0.991	0.944	0.901	0.883
Third objective	36	36	36	36	36	36	36	36
Fourth objective	71.1	71.4	71.4	71.1	71.4	71.5	71.5	71.5
Total objective (10^{13})	5.115	4.368	4.142	1.671	1.445	4.971	4.392	3.056

Table A6. Sum of shortage for the device at each pass for Taiwan 2 data set without initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Shortage after first pass (10^5)	1.623	1.153	1.153	0.455	0.455	0.455	1.313	1.012
Shortage after second pass (10^5)	0.410	0.394	0.394	0	0	1.168	0.452	0.245
Shortage after third pass (10^5)	4.872	5.959	4.872	3.663	2.576	7.490	3.912	2.449

Table A7. Results for Taiwan 2 data set with initial setups (9/36 of the machines have initial setups)

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	397	421	407	415	418	363	438	437
Total steps finished	583	609	586	607	611	627	658	659
Key device shortage in pieces (10^5)	1.180	1.119	1.122	0.447	0.447	1.887	1.272	1.272
First objective (weighted key device shortage) (10^{13})	3.641	3.519	3.407	0.903	0.903	4.824	3.202	3.168
Changeover	28	22	14	29	26	21	17	16
Number of machines used	36	36	36	36	36	36	36	36
Second objective (10^9)	1.005	1.031	1.034	1.001	1.012	0.978	0.934	0.911
Third objective	36	36	36	36	36	36	36	36
Fourth objective	71.5	71.5	71.5	71.5	71.5	72	72	72
Total objective (10^{13})	3.641	3.519	3.407	0.903	0.903	4.824	3.202	3.168

Table A8. Sum of shortages for key devices at each pass for Taiwan 2 data set with 9 initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Shortage after first pass (10^5)	0.966	0.905	0.850	0.303	0.303	0.303	1.140	1.140
Shortage after second pass (10^5)	0.413	0.413	0.413	0	0	1.333	0.171	0.155
Shortage after third pass (10^5)	0.411	0.411	0.411	0.144	0.144	0.657	0.269	0.269

Table A9. Results for Clark Probe data set without initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	116	117	115	111	116	121	107	109
Total steps finished	119	120	118	114	119	123	109	111
Key device shortage in pieces (10^5)	1.060	1.060	1.059	1.083	1.065	1.026	1.073	1.036
First objective (weighted key device shortage) (10^{13})	2.326	2.326	2.327	2.375	2.336	2.256	2.352	2.276
Changeovers	17	28	39	13	17	0	0	0
Number of machines used	131	130	48	131	128	48	76	77
Second objective (10^9)	1.679	1.679	1.672	1.617	1.666	1.715	1.566	1.583
Third objective	131	130	48	131	128	48	76	77
Fourth objective	85.1	89.3	111.1	109	104	116	119.5	119.5
Total objective (10^{13})	2.326	2.326	2.327	2.375	2.336	2.256	2.352	2.276

Table A10. Results for Clark Probe data set with initial setups (45/136 machines have initial setups)

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	119	119	116	116	116	120	107	112
Total steps finished	126	126	123	123	123	122	114	119
Key device shortage in pieces (10^5)	0.101	0.101	0.103	0.101	0.102	0.058	0.103	0.101
First objective (weighted key device shortage) (10^{13})	2.221	2.221	2.264	2.225	2.225	1.250	2.266	2.225
Changeover	13	30	39	11	11	0	0	1
Number of machines used	135	135	71	135	135	71	99	98
Second objective (10^9)	1.763	1.763	1.733	1.720	1.720	1.799	1.618	1.697
Third objective	135	135	71	135	135	71	99	98
Fourth objective	112	112	111	112	112	120	120	120
Total objective (10^{13})	2.221	2.221	2.264	2.225	2.225	1.250	2.266	2.225

Table A11. Results for Taiwan 1 data set with initial setups (10/36 machines have initial setups)

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	326	410	401	371	393	420	470	475
Total steps finished	457	541	537	509	532	667	681	690
Key device shortage in pieces (10^5)	3.327	1.421	1.280	0.776	0.667	1.795	1.008	0.941
First objective (weighted key device shortage) (10^{13})	8.214	4.284	3.994	1.580	1.354	4.262	3.833	3.000
Changeover	26	20	18	32	25	22	16	17
Number of machines used	36	36	36	36	36	36	36	36
Second objective (10^9)	0.908	0.994	0.998	0.988	0.990	0.957	0.921	0.933
Third objective	36	36	36	36	36	36	36	36
Fourth objective	7.11	7.06	7.11	7.11	7.14	7.2	7.2	7.2
Total objective (10^{13})	8.214	4.284	3.9940	1.580	1.354	4.262	3.833	3.000

Table A12. Sum of shortage for key devices at each pass for Taiwan 1 data set with 10 initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Shortage after first pass (10^5)	3.054	1.111	1.080	0.410	0.410	0.411	0.901	0.719
Shortage after second pass (10^5)	0.410	0.394	0.394	0	0	0.865	0.526	0.372
Shortage after third pass (10^5)	0.596	0.596	0.487	0.366	0.258	0.759	0.445	0.377

Table A13. Results for Taiwan data set 1 with initial setups (18/36 of the machines have initial setups)

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Total lots finished	387	424	423	440	421	370	442	444
Total steps finished	574	607	613	632	616	639	657	670
Key device shortage in pieces (10^5)	2.788	1.119	1.122	0.452	0.447	1.907	1.297	1.240
First objective (weighted key device shortage) (10^{13})	6.874	3.519	3.407	0.914	0.903	4.897	3.615	3.197
Changeover	25	23	18	30	28	18	16	17
Number of machines used	36	36	36	36	36	36	36	36
Second objective (10^9)	1.005	1.025	1.025	1.010	1.009	0.955	0.964	0.958
Third objective	36	36	36	36	36	36	36	36
Fourth objective	72	72	72	72	72	72	72	72
Total objective (10^{13})	6.874	3.519	3.407	0.914	0.903	4.897	3.615	3.197

Table A14. Sum of shortage for the device at each pass for Taiwan data set 1 with 18 initial setups

Measures	Sim	First	All/Lim	Hot lot	SetupNum	GRASP_opt	Greedy	GRASP_asap
Shortage after first pass (10^5)	2.575	0.905	0.850	0.308	0.303	0.303	1.177	1.162
Shortage after second pass (10^5)	0.413	0.413	0.413	0	0	1.348	0.311	0.181
Shortage after third pass (10^5)	0.411	0.411	0.411	0.144	0.144	0.677	0.291	0.234

Appendix A4: Regression with CRN results before taking log transformation

Table A15. Experimental design with CRN results for firstobj values using OLS

firstobj	Estimate of β	Std. error	t value	$p > \|t\|$
Intercept	5.9871e+13	7.5473e+11	79.21	<2e-16
Rule_First	-2.5575e+13	1.3484e+12	-18.97	<2e-16
Rule_HotLot	-4.6650e+13	1.0401e+12	-44.85	<2e-16
Rule_SetupNum	-4.6200e+13	9.4420e+11	-48.93	<2e-16
inisetup_yes	5.9375e+12	2.9353e+12	2.02	4.42e-5
control_yes	3.3738e+13	1.5795e+12	21.36	<2e-16
Rule_First*inisetup_yes	2.2500e+12	3.2298e+12	0.70	0.2448
Rule_HotLot*inisetup_yes	-2.0000e+11	3.1230e+12	-0.06	0.5237
Rule_SetupNum*inisetup_yes	1.5000e+12	3.0245e+12	0.50	0.3104
Rule_First*control_yes	-2.4850e+13	1.8951e+12	-13.11	<2e-16
Rule_HotLot*control_yes	-2.7500e+13	1.7788e+12	-15.46	<2e-16
Rule_SetupNum*control_yes	-2.8600e+13	1.6951e+12	-16.87	<2e-16
Inisetup_yes*control_yes	-6.8750e+12	1.2356e+12	-5.56	2.61e-6

Table A16. Experimental design with CRN results for firstobj values using EGLS

firstobj	Estimate of β	Std. error	t value	$p > \|t\|$
Intercept	5.1270e+13	4.1020e+11	124.99	<2e-16
Rule_First	-1.6941e+13	1.0587e+12	-16.00	<2e-16
Rule_HotLot	-3.6696e+13	5.5947e+12	-65.59	<2e-16
Rule_SetupNum	-3.7462e+13	5.8214e+11	-64.35	<2e-16
inisetup_yes	1.3117e+13	1.5626e+12	4.55	4.421e-5
control_yes	3.0930e+13	1.5626e+12	19.79	<2e-16
Rule_First*inisetup_yes	-1.2698e+13	2.7094e+12	-4.69	3.00e-5
Rule_HotLot*inisetup_yes	-1.3049e+13	2.8909e+12	-4.51	4.94e-5
Rule_SetupNum*inisetup_yes	-1.2226e+13	2.6591e+12	-4.60	3.85e-5
Rule_First*control_yes	-2.5076e+13	1.7607e+12	-14.24	<2e-16
Rule_HotLot*control_yes	-3.0409e+13	1.5428e+12	-19.71	<2e-16
Rule_SetupNum*control_yes	-2.9238e+13	1.5938e+12	-18.35	<2e-16
Inisetup_yes*control_yes	-6.1714e+12	1.1078e+12	-5.57	7.768e-9

Table A17. Experimental design with CRN results for secondobj values using OLS

secondobj	Estimate of β	Std. error	*t* value	$p > \|t\|$
Intercept	1.0378e+9	1.8362e+6	565.17	<2e-16
Rule_First	5.1567e+7	2.6446e+6	19.50	<2e-16
Rule_HotLot	3.7489e+7	2.6060e+6	14.39	<2e-16
Rule_SetupNum	4.0740e+7	3.3906e+6	12.02	<2e-16
inisetup_yes	-3.2922e+6	3.6201e+6	-0.91	0.1852
control_yes	-9.3614e+6	2.2653e+6	-4.13	0.0001
Rule_First*inisetup_yes	-1.8382e+6	3.8182e+6	-0.48	0.3174
Rule_HotLot*inisetup_yes	4.9023e+6	3.8994e+6	1.26	0.1089
Rule_SetupNum*inisetup_yes	-2.9755e+6	6.2503e+6	-0.48	0.3174
Rule_First*control_yes	2.2660e+6	3.1629e+6	0.72	0.2386
Rule_HotLot*control_yes	-4.5629e+7	3.9768e+6	-11.47	<2e-16
Rule_SetupNum*control_yes	-7.0864e+6	5.1231e+6	-1.38	0.0891
Inisetup_yes*control_yes	4.3184e+6	3.7770e+6	1.14	0.1318

Table A18. Experimental design with CRN results for secondobj values using EGLS

secondobj	Estimate of β	Std. error	*t* value	$p > \|t\|$
Intercept	1.0350e+9	1.3737e+6	753.45	<2e-16
Rule_First	5.6259e+7	2.3754e+6	23.68	<2e-16
Rule_HotLot	3.8933e+7	2.2028e+6	17.67	<2e-16
Rule_SetupNum	3.9901e+7	2.2533e+6	17.71	<2e-16
inisetup_yes	3.0044e+6	2.5941e+6	1.16	0.1278
control_yes	-4.4482e+6	1.1362e+6	-3.91	0.0003
Rule_First*inisetup_yes	8.2783e+6	2.7893e+6	2.97	0.0030
Rule_HotLot*inisetup_yes	-2.3039e+6	2.6797e+6	-0.86	0.1984
Rule_SetupNum*inisetup_yes	-4.6234e+6	2.9904e+6	-1.55	0.0660
Rule_First*control_yes	-7.8713e+6	2.0570e+6	-3.83	0.0003
Rule_HotLot*control_yes	-4.3652e+7	3.8207e+6	-11.43	<2e-16
Rule_SetupNum*control_yes	-6.2394e+6	1.6120e+6	-3.87	0.0003
Inisetup_yes*control_yes	-5.4380e+6	1.2496e+6	-4.35	7.68e-5

References

Abdi, H. (2007). Bonferroni and Šidák corrections for multiple comparisons. *Encyclopedia of Measurement and Statistics*.

Aldakhilallah, K.A. and Ramesh, R. (2001). Cyclic scheduling heuristics for a re-entrant job shop manufacturing environment. *International Journal of Production Research*, 39 (12), 2635-2657.

Allahverdi, A., Gupta, J.N.D. and Aldowaisan, T. (1999). A review of scheduling research involving setup consideration. *OMEGA, The International Journal of Management Science*, 27 (2), 219-239.

Atherton, L. and Atherton, R. (1995). *Wafer Fabrication: Factory Performance and Analysis*. Kluwer, Boston.

Asmundsson, J., Rardin, R.L. and Uzsoy, R. (2006). Tractable nonlinear production planning models for semiconductor wafer fabrication facilities. *IEEE Transactions on Semiconductor Manufacturing*, 19(1), 95-111.

Bang, J.Y. and Kim, Y.D. (2011). Scheduling algorithms for a semiconductor proving facility. *Computers & Operation Research*, 38(3), 666-673.

Bard, J.F., Gao, Z., Chacon, R. and Stuber, J. (2013). Daily scheduling of multi-pass lots at assembly and test facilities. *International Journal of Production Research*. 51(23-24), 7047-7070.

Bard, J.F., Jia, S., Chacon, R. and Stuber, J. (2015). Integrating optimization and simulation approaches for daily scheduling of assembly and test operations. *International Journal of Production Research*, 53(9), 2617-2632.

Bispo, C. F. and Tayur, S. (2001). Managing simple re-entrant flow lines: theoretical foundation and experimental results, *IIE Transactions*, 33(8), 609−623.

Chang,Y.S., Choe, S.H. and Besant, C.B. (1997). Discrete event simulation in a semiconductor assembly area. *Proceedings of the 1997 5th International Conference on Factory 2000 the Technology Exploitation Process*, Cambridge (April 2-4, 1997), pages 183-188, IEE Conference Publication, ISBN: 0537-9989.

Chen, C.F., Wu, K.J., Chang, C.T., Wong, D.S. and Jang, S. S.(2013). Generation and

verification of optimal dispatching policies for multi-product multi-tool semiconductor manufacturing processes. *Computers & Chemical Engineering*, 52(1), 112-121.

Chen, H.A., Chang, L., Yuan, R. and Jun, Z. (2012). Research on dynamic dispatching rule for semiconductor assembly production line. *Proceedings of the 2012 IEEE IEEM*, 493-497.

Chen, J.-S., Pan, J.C.-H. and Wu, C.-K. (2007). Minimizing makespan in reentrant flow-shops using hybrid tabu search. *International Journal of Advanced Manufacturing Technology*, 34(3). 353-361.

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.

Choi, H.S., Kim, J.S., and Lee, D.H. (2011). Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to a TFT-LCD line. *Expert Systems with Applications*. 38(4), 3514-3521.

Danping, L. 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.

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.

Duffie, N.A. and Shi, L. (2009). Maintaining constant WIP-regulation dynamics in production networks with autonomous work systems. *CIRP Annals – Manufacturing Technology*, 58(1), 399–402.

Dugardin, F., Yalaoui, Farouk, F., and Amodeo, L. (2010). New multi-objective method to solve reentrant hybrid flow shop scheduling problem. *European Journal of Operational Research*, 203(1), 22-31.

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

Festa P. and Resende, M.G.C. (2009). An annotated bibliography of GRASP - Part I:

algorithms. *International Transactions in Operational Research* 16(1), 1-24.

Freed, T., Doerr, K.H. and T. 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.

Fu, M.Y., Askin, R., Fowler, J., Haghnevis, M., Keng, N.P., Pettinato, J. and Zhang, M.H. (2011). Batch Production Scheduling for Semiconductor Back-End Operations. *IEEE Transactions on semiconductor manufacturing*. 24(2), 249-260.

Gershwin, S.B., (2000). Design and operation of manufacturing systems: the control-point policy. *IIE Transactions*, 32(10), 891–906.

Gharbi, A. and Kenne, J.P. (2000). Production and preventive maintenance rates control for a manufacturing system: An experimental design approach. *International Journal of Production Economics*, 65(3), 275–287.

Godfrey, L. G. (1978). Testing against general autoregressive and moving average error models when the regressors include lagged dependent variables. *Econometrica*, 46(1), 1293-1301.

Gowling, B., Peterson, J., O'Donnell, T., Kidambi, M. and Muller, D. (2013). An integrated approach to real time dispatching rules analysis at Seagate technology. In R. Rasupathy, S.-H. Kim, A. Tolk, and M.E. Kuhl (eds.), *Proceedings of the 2013 Winter Simulation Conference*, 3766-3776, Institute of Electrical and Electronics Engineers, Piscataway, NJ.

Graves, S.C., Meal, H.C., Stefek, D. and Zeghmi, A.H. (1983). Scheduling of re-entrant flow shops. *Journal of Operations Management*, 3(4), 197-207.

Health, S.K. and Morrice, D.J. (2007). A comparison of scheduling approaches for a make-to-order electronics manufacturer. In S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R.R. Barton (eds.) *Proceedings of the 2007 Winder Simulation Conference*, pp. 1953-1960. Institute of Electrical and Electronics Engineers, Piscataway, NJ.

Hood, S.J. and Welch P.D. (1992). Experimental design issues in simulation with examples from semiconductor manufacturing. In J.J. Swain, D. Goldsman, R.C. Crain,

and J.R. Wilson (eds.), *Proceedings of the 1992 Winter Simulation Conference*, 255-263, Institute of Electrical and Electronics Engineers, Piscataway, NJ.

Hung, Y.F. and Leachman, R.C. (1996). A production planning methodology for semiconductor manufacturing based on iterative simulation and linear programming calculations. *IEEE Transactions on Semiconductor Manufacturing*, 9(2), 257–269.

Jia, S., Bard, J.F., Chacon, R. and Stuber, J. (2015). Improving performance of dispatch rules for daily scheduling of assembly and test operations. *Computers & Industrial Engineering*, 90(1), 86-106.

Judge, G., Hill, C., Griffiths, W. and Lee, T. (1985). *The Theory and Practice of Econometrics*. John Wiley & Sons, New York.

Kleijnen, JPC. (1988). Analyzing simulation experiments with common random numbers. *Management Science*, 34(1), 65-74.

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

Lee, L.H, Chew, E.P, Frazier P.I., Jia, Q.-S. and Chen, C.-H. (2013).Advances in simulation optimization and its applications, *IIE Transactions on Operations Engineering & Analysis*, 45(7), 683-684.

Li, L., Sun, Z.J., Zhou, M.C. and Qiao, F. (2013). Adaptive dispatching rule for semiconductor wafer fabrication facility. *IEEE Transactions on Automation Science and Engineering*, 10(2), 354-364.

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.

Mackulak, G.T. and Savory, P. (2001). A Simulation-Based Experiment for Comparing AMHS Performance in a Semiconductor Fabrication Facility. *IEEE Transactions on Semiconductor Manufacturing*, 14(3), 273-280.

Kaihara, T., Kurose, S. and Fukii, N. (2012). A proposal on optimized scheduling methodology and its application to an actual-scale semiconductor manufacturing problem. *CIRP Annals – Manufacturing Technology*, 61(1), 467-470.

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.

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

Montoya-Torres, J.R. (2006). A literature survey on the design approaches and operational issues of automated wafer-transport systems for wafer fabs. *Production Planning and Control*, 17(7), 648–663.

Narahari, Y. and Khan, K. (1996). Performance analysis of scheduling policies in re-entrant manufacturing systems. *Computers & Operations Research*, 23(1), 37−57.

Park, Y., Kim, S. and Jun, C.H. (2002). Performance evaluation of re-entrant lines with multi-class jobs and multi-server workmachines, *Production Planning and Control*, 13(1), 56−65, 2002.

Pasupathy, R. and Ghosh, S. (2013). Simulation optimization: A concise overview and implementation guide. In H. Topaloglu (ed.), *Tutorials in Operations Research*, Vol. 10, Chapter 7, 122-150, INFORMS, Cantonsville, MD.

Perdaen, D., Armvruster, D., Kempf, K. and Lefeber E. (2008). Controlling a re-enterant manufacturing line via the push-pull point. *International Journal of Production Research*, 46(16), 4521-4536.

Pfund, M.E., Mason, S.J. and Fowler, J.W. (2006), Semiconductor manufacturing scheduling and dispatching. In J.W. Herrmann (ed.), *Handbook of Scheduling*, Chapter 9, pp. 213-241, Springer, Berlin.

Prais, M., Ribeiro, C.C. (1999). Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3), 164-176.

Saito, Kazuyuki (2007). A robust dispatching algorithm for autonomous distributed manufacturing of mixed VLSI products. *IEEE Transactions on Semiconductor Manufacturing*, 20(3), 299-305.

Sarimveis, H., Patrinos, P., Tarantilis, C.D. and Kiranoudis, C.T. (2008). Dynamic modeling and control of supply chain systems: a review. *Computers & Operations Research*, 35(11), 3530-3561.

Sels, V., Gheyse, N. and Vanhoucke, M. (2012). A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. *International Journal of Production Research*, 50(15), 4255–4270.

Sivakumar, A.I. (1999). Optimization of cycle time and utilization in semiconductor test manufacturing using simulation based, on-line, near-real-time scheduling system. In P.A. Farrington, H.B. Nembhard, D.T. Sturrock and G. W. Evans (eds.), *Proceedings of the 1999 Winter Simulation Conference*, Institute of Electrical and Electronics Engineer, Piscataway, NJ:.

Sivakumar, A. I. and Gupta, A.K., (2002). Simulation based multiobjective schedule optimization in semiconductor manufacturing. In M.E. Kuhl, N.M. Steiger, F.B. Armstrong and J.A. Joines (eds.), *Proceedings of the 2005 Winter Simulation Conference*, 83-95, Institute of Electrical and Electronics Engineers, Piscataway, NJ.

Song, Y., Zhang, M.T., Yi, J. 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.

Starkov, K.K., Pogromsky, A.Y., Adan, I.J.B.F and Rooda, J.E. (2013). Performance analysis of re-entrant manufacturing networks under surplus-based production control. *International Journal of Projection Research*, 51(5), 1563-1586.

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.

Werner, S., Horn, S., Weigert, G. and Jahnig, T. (2006). Simulation based scheduling system in a semiconductor backend facility. In L.F. Perrone, F.P. Wieland, J. Liu, B.G. Lawson, D.M. Nicol and R.M. Fujimoto (eds.), *Proceedings of the 2006 Winter Simulation Conference*, 1741-1748, Institute of Electrical and Electronics Engineers,

Piscataway, NJ.

Wu, M.C., Chiou, S.J. and Chen, C.F. (2008). Dispatching for make-to-order wafer fabs with machine-dedication and mask set-up characteristics. *International Journal of Production Research*. 46(14), 3993–4009.

Xu, W.L. and Nelson, B.L. (2013). Empirical stochastic branch-and- bound for optimization via simulation. *IIE Transactions on Operations Engineering & Analysis*, 45(7), 685-698.

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

Van Zon, A.H. and Kommer, G.J. (1999). Patient flows and optimal health-care resource allocation at the macro-level: a dynamic linear programming approach. *Health Care Management Science*, 2, 87-96.

Woodall, J.C., Gosselin, T., Boswell, A., Murr, M. and Denton, B.T. (2013). Improving patient access to chemotherapy treatment at Duke Cancer Institute. *Interfaces*, 43(5), 449-461.

Ying, K.-C. and Lin, S.-W. (2014). Efficient wafer sorting scheduling using a hybrid artificial immune system. Journal of the Operational Research Society, 65(2), 169-179.

Zhang, H., Jiang, Z., and Guo, C. (2009). Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology. *International Journal of Advanced Manufacturing Technology*, 41(1), 110–121.

Zhang, M.T., Niu, S., Deng, S., Zhang, Z., Li, Q. and Zheng, L. (2009). Hierarchical capacity planning with reconfigurable kits in global semiconductor assembly and test manufacturing methodology. *IEEE Transactions on Automation Science and Engineering*, 4(4), 543–552.