

Evaluation of heuristics for a class-constrained lot-to-order matching problem in semiconductor manufacturing

Thomas G. Boushell† (thomas.boushell@asu.edu), John W. Fowler*† (john.fowler@asu.edu), Ahmet B. Keha† (ahmet.keha@asu.edu), Kraig R. Knutson‡ (kraig.knutson@asu.edu) and Douglas C. Montgomery† (doug.montgomery@asu.edu)

†Department of Industrial Engineering, Arizona State University, Tempe, AZ 85287, USA

‡Del E. Webb School of Construction, Arizona State University, Tempe, AZ 85287, USA

*Fax: 480.965.8692, Phone: 480.965.3727

Abstract

Semiconductor industry experts project sales to exceed \$250 billion by the year 2007.

While semiconductors are being used in an increasing number of products, semiconductor manufacturers continually look for ways to make their processes more efficient. This paper will focus on an issue in the manufacturing process called the class-constrained lot-to-order matching problem (CLOMP), where individual lots of microprocessors are matched to customer orders, while seeking to optimize multiple objectives. Due to its complexity, the problem is decomposed into two stages—the first identifies which customer orders to fill while the second assigns specific lots to the chosen orders. We design an experiment with four first-stage sorting rules, four second-stage heuristics and two production cases. Based on our simulation results, this paper will recommend the first-stage sorting rule and second-stage heuristic which attain the best results with regards to our measures of effectiveness.

Author keywords: Packing; bin covering; heuristics; class-constrained lot-to-order matching problem; semiconductors; design of experiments

1. Introduction

According to the Semiconductor Industry Association, world semiconductor sales reached an all-time high in 2004, exceeding \$213 billion (SIA 2004). While semiconductors are being used in an increasing number of products, semiconductor manufacturers continually look for ways to make their processes more efficient. These improvements not only sustain an already flourishing industry, but also result in substantial profit margin increases.

A general explanation of the basic product flow in a semiconductor production chain can be found in Frederix (1996) and Knutson *et al.* (1999). This paper will address cases involving the following specific circumstances. First, while companies' manufacturing strategy can be either 'make-to-stock' or 'make-to-order', this paper only looks at the 'make-to-order' scenario where unassembled wafers are stored in a lot warehouse awaiting their order assignment. Second, we will look at the two-class version, where each end-product is designated as either a high- or low-speed integrated circuit. It is assumed that customer orders can be for high-speed, low-speed or a combination of both classes. Whenever a customer order is assigned to the factory, all classes must be filled completely by choosing from the selection of lots in the lot warehouse. When a lot is chosen to fill an order, all die of the same class within the lot must be used to fill the same order. However, each of the classes within the lot is allowed to be assigned to different orders. Last, we assume that a single class can be taken from the lot, leaving the other classes unassembled at the lot warehouse and ready to be assigned to another order.

The class-constrained lot-to-order matching problem has four main objectives. The first two main objectives are due-date related performance metrics used to analyse how well customers are taken care of. In this research, the first of two due-date metrics used is total weighted tardiness (TWT), because it combines the customer's priority, the number of dies and the tardiness of delivery for each order into one value. The other metric, on-time delivery (OTD), calculates the percentage of orders delivered to the customer on or before their due-dates.

The third objective is to ensure that the assembly and test (A&T) facility is being effectively utilized. This metric can be measured by ascertaining how many dies are being used towards customer orders. This metric is referred to as the number of dies to order or DTO. The last objective is to minimize the surplus of unused die that get sent to the warehouse (DTW). It should be obvious that all integrated circuits from an assigned lot will either become DTO or DTW.

1.1. Literature review

While the class-constrained lot-to-order matching problem (CLOMP) exhibits multiple characteristics of various packing problems, it does not conform to any one in particular. In fact, no literature could be found regarding its specific formulation or solution methodology, despite the extensive collection of packing problem variations. Three papers and one Ph.D. dissertation of the same lineage, however, do look at the lot-to-order matching problem (LOMP), where only one class of product is allowed (Knutson 1998, Knutson *et al.* 1999, Fowler *et al.* 2000, Carlyle *et al.* 2001). The problem we solve in this paper, the CLOMP, is a generalized version of Knutson's LOMP, because multiple classes of product are present in the customer orders and factory

lots. Like the LOMP papers, our approach is to formulate it as a mathematical programming problem, decompose it into two subproblems and then solve it heuristically. The need to break the problem into two parts arises from the computational difficulty in solving the original problem. The first phase, which selects the orders, is solved as a knapsack problem and provides input to the second phase, which is a modified bin covering problem that matches lots to the chosen orders.

The cutting and packing family of problems, which includes cutting stock, bin packing, dual bin packing, bin covering and knapsack problems to name a few, is well-established and too expansive to be reviewed in great detail in this paper. Dowsland and Dowsland (1992) observed how the close relationship between packing problems and the classical stock cutting, loading and knapsack problems make it very difficult to define clear classification boundaries. Because of this cloudy delineation of problems, Dyckhoff (1990) developed a consistent and systematic approach to classifying the various types of cutting and packing problems. He underscored the importance of these problems in the literature by presenting a list of cutting and packing surveys that were published about once per year in the 1970s and 1980s. Refer to Boushell *et al.* (2005) for a detailed discussion of the cutting and packing family of problems.

As stated above, the most relevant set of papers in our review specifically deal with the Lot-to-Order Matching problem (LOMP) (Knutson 1998, Knutson *et al.* 1999, Fowler *et al.* 2000, Carlyle *et al.* 2001). The LOMP can be described as the assignment of orders into a capacity-constrained factory, while simultaneously assigning factory lots of different sizes into those customer orders. After formulating the problem as a single integer program, Knutson *et al.* (1999) decomposed the problem into two distinct stages.

The first stage used a knapsack algorithm to identify which orders to send to the factory to be filled. Bin covering algorithms were used in the second stage to identify which lots would be used to fill the orders selected during the first stage. The first stage attempted to maximize the utilization of the capacity-constrained factory—the first of three objectives. The second objective was to minimize the number of orders delivered late by mapping the due date into a penalty function. The final objective, which was directly accounted for in the Stage 2 objective function, attempted to minimize the production of excess inventory (Knutson *et al.* 1999).

In the LOMP, a First-Fit-Decreasing (FFD) policy based on order sizes modified by due dates was initially compared to the existing First-In-First-Out (FIFO) first stage policy (Knutson *et al.* 1999). Subsequent evaluations utilized density decreasing greedy (DDG) and value decreasing greedy (VDG) algorithms (Fowler *et al.* 2000). The second stage compared the same initial policies (FFD and FIFO), but eventually evolved to a first-fit-decreasing (FFD1(S_i)) algorithm that combines a search routine with a priority rule system for matching the lots to orders (Knutson *et al.* 1999, Fowler *et al.* 2000).

1.2. Organisation of the paper

This paper contains five sections, with the introduction and literature review included in the first section. The rest of the paper is organized as follows: Section 2 describes the optimization model focusing on the specific details of the first stage sorting functions and second stage matching algorithms. In Section 3, we describe the experimental design and provide an in-depth description of the four response variables. The modeling details and assumptions are explained in Section 4, while Section 5 contains the results and conclusions.

2. Optimization model descriptions

In order to make smart, timely decisions, as we discussed earlier, the CLOMP is decomposed into two distinct stages or phases. Each of these phases is responsible for making one of the two sets of decisions which affect the four CLOMP objectives. While the mathematical formulation is shown in (Boushell *et al.* 2005), a modified diagram from (Fowler *et al.* 2000) of the relationship between the two stages is depicted in Figure 1. The first phase selects which customer orders to fill on a given day, while the second phase identifies the lots which should be used to fill those selected orders. These decisions are made in an attempt to maximize the on-time delivery of the customer orders (OTD), minimize the total weighted tardiness of the customer orders (TWT), minimize the number of wasted die that exceed the selected orders (DTW) and maximize the number of die satisfying customer orders (DTO).

Although only two stages are defined, the process of assigning lots to orders requires numerous steps. The same two stages, however, are iteratively applied. At the beginning of each day, the customer orders are arranged in non-increasing order based on the value determined by one of the sorting functions described in Section 2.1. The first-stage knapsack algorithm selects the highest-valued order that can fit within the factory, as long as there is a sufficient number of dies in the lot warehouse to cover the entire order. Before selecting the next highest-valued order, lots are assigned to the previous customer order using one of the second-stage matching algorithms described in Section 2.2. The total number of dies that actually get assigned to the order is used to determine how much capacity remains in the factory for future iterations. The reason is because the lots, not the orders, are the physical items that get sent through the

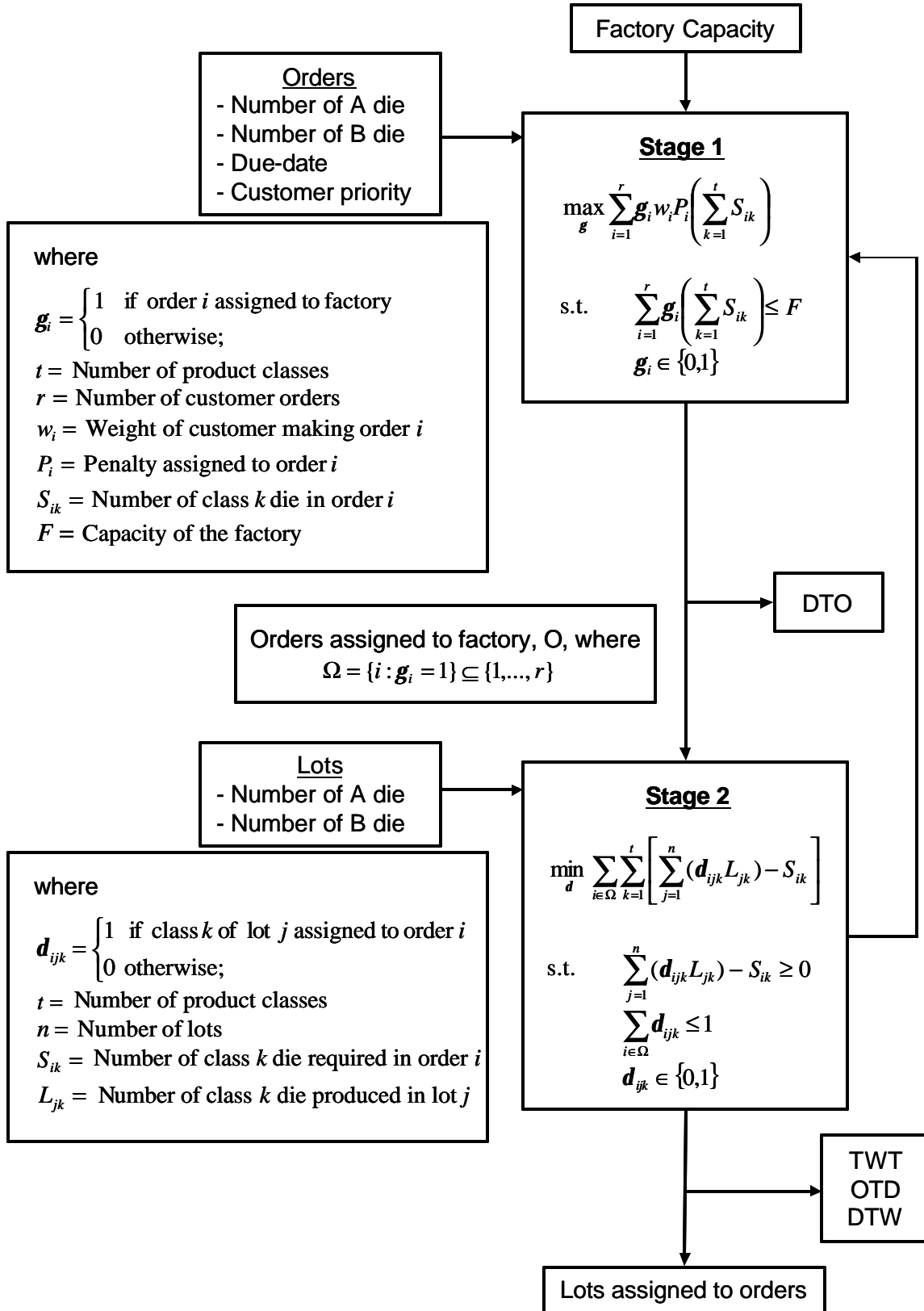


Figure 1: Decomposition of CLOMP into two stages

factory. It should be evident that efficient matching is paramount to optimizing the objectives.

At the conclusion of the lot-to-order matching for the initial order, the first stage is revisited, where the highest-valued remaining order that meets the capacity and covering criteria is chosen. This chosen order, like the first, is matched up with unassigned lots from the lot warehouse using a Stage 2 matching algorithm. The number of factory-assigned dies and warehouse-dies available are updated to reflect the current state before returning to the first stage. This recursive process continues until the last order is checked.

2.1. Stage 1 sorting functions

The sorting functions in Stage 1 are specific guidelines used to identify which orders to choose, allowing for an objective order-selection process. While making selections according to these calculated values attempt to find the best performance with regards to the four main objectives, the highest value does not guarantee the greatest performance. The desire is to find the sorting function that best correlates the calculated values in Stage 1 to the effect on the four objectives. Obviously, the analysis of the results, not the perceived value of the Stage 1 selections, will distinguish which sorting function yields the best performance. In an effort to find the best results, four sorting functions were identified and their results subsequently analysed.

In each of the four cases, the orders are arranged in non-increasing order according to their values. The order-values are calculated each day, based on which sorting function version is being used. The four sorting functions are described in Sections 2.1.1 – 2.1.4. Regardless of which sorting function is applied, the Stage 1

process follows the same steps of the knapsack algorithm shown below. The second stage matching algorithms will be described in Section 2.2.

Stage 1 Order Selection

- Step 1. Sort orders in non-increasing order based on the order value
- Step 2. If all orders have been considered, go to *Step 10*.
- Step 3. Select the next order. Set k to initial class, where k is the class-type (A, B, etc).
- Step 4. If order will not fit with previously assigned orders into factory, go to *Step 2*.
- Step 5. If class k die in lot warehouse can not fill order's class k request, go to *Step 2*.
- Step 6. If k is not last class, set k to next class and go to *Step 5*.
- Step 7. Match lots using appropriate *Stage 2* matching algorithm.
- Step 8. Update number of factory-assigned dies and warehouse-dies available.
- Step 9. Go to *Step 2*.
- Step 10. File lots and orders in appropriate queues. Stop.

2.1.1. First-in, first-out (FIFO)

The baseline sorting function, FIFO, gives priority to the order which was received first and is unaffected by order due-dates or the importance of a particular customer. While the order that was received first is always assigned first, the dies of earlier orders will not necessarily be delivered to the customer first due to the stochastic nature of the assembly and test facility.

2.1.2. Total weighted tardiness (TWT)

In an effort to specifically address the total weighted tardiness objective, the TWT sorting function, $w_i P_i \sum_{k=1}^t S_{ik}$, comes from the product of the customer priority level the projected tardiness and the size of the order. The customer priority level, w_i , is the weight or priority of the customer placing the order. This variable enables the decision maker to bias the assignment of orders in favor of preferred customers or specific orders.

Because the TWT and OTD metrics do not value early deliveries more than those that arrive exactly on-time, it is important to keep the differences between customer levels from being too great. In other words, high-priority orders should not be completing well in advance of their due-date at the expense of other near-deadline orders.

The second term in the TWT sorting function is a penalty value, P_i , which places more emphasis on orders that have closer due-dates. Although we can not be sure when an order's lots will actually complete the A&T process, we specify a penalty value that is related to the order's due-date. Because most orders complete the A&T processing in less than 22 days, this threshold was established to distinguish between time-sensitive orders and those orders that can safely be delivered on-time without a rush. All orders that have due-dates at least 22 days away receive values of 1.0. For each day closer than 22 days, 0.3 is added. For example, if the current date is day 50, an order due on day 70, would receive a penalty value of 1.6. Examples of other penalty values for various due-dates on day 50 are shown in table 1.

D_i	64	65	66	67	68	69	70	71	72	73	> 73
P_i	3.4	3.1	2.8	2.5	2.2	1.9	1.6	1.3	1.0	1.0	1.0

Table 1: Day 50 – Penalty (P_i) as a function of due-date (D_i)

The last term in the TWT sorting function is the size of the order, SS_{ik} . Because we do not know how many dies will be assigned to the orders until the second stage, we use the actual number of dies required in the order for this value.

2.1.3. Apparent tardiness cost (ATC)

The combination of weighted orders and tardiness concerns led us to the composite dispatching rule known as ATC (Pinedo 1995). The ATC sorting function

leads us to a ranking indexing system where the highest valued order is processed first. Each day, the ATC index (at time t) for all of the orders is calculated using the following equation:

$$I_i(t) = \frac{w_i}{p_i} \exp\left(-\frac{\max(d_i - p_i - t, 0)}{k\bar{p}}\right) \quad (1)$$

In this equation, w_i and d_i are the same customer priority and order due-date as previously discussed. The ranking index function also is dependent on the individual order processing times, p_i , the average processing times of the remaining orders, \bar{p} and the scaling parameter, k .

Because the processing time of an order is calculated from the time the first lot in the order is sent to the A&T factory until the last lot in the order has been processed, there is no way to perfectly predict the values for p_i . However, two concepts assist in estimating the order processing times. First, in the simulation model, larger lots are going to, on average, exhibit longer processing times than smaller lots. Second, because the order processing time is equal to the processing time of its longest lot, the inclusion of additional lots can only increase p_i . Knowing that larger orders will typically have either a larger number of dies per lot or a larger number of individual lots, it is safe to assume that the larger orders will take more time to process than smaller orders. This information, combined with the realization that a significant difference in processing times can dominate the indexing function, drove us to a transformation function that related orders with 45 000, 50 000 and 55 000 dies to processing times of 16, 17 and 18 days, respectively. The value for \bar{p} is recalculated at the beginning of each day.

Knowing the ranges of all the other variables in the indexing function made the selection of k straightforward. Two key attributes were identified as important when

selecting the scaling parameter. First, orders without slack—those jobs with due-dates occurring before their estimated completion time—should be given higher priority than those orders with projected slack. Next, regardless of the size of orders, higher priority customers should have their orders selected first, assuming their projected slacks are nearly identical. After reviewing a number of scaling parameter values, it was evident that $k = 0.1$ was most consistent with our desired attributes.

2.1.4. Earliest due-date (EDD)

The final sorting function, EDD, sorts the orders according to their due-dates, with the earliest due-date selected first. Like FIFO, EDD is unaffected by the importance of a particular customer. By only addressing the due-date of an order, we attempt to limit the tardiness, which affects both the TWT and OTD objectives, by selecting the orders with the least amount of lead-time.

2.2. Stage 2 matching algorithms

As discussed earlier, the daily assignment of lots to orders is an iterative process. Rather than making all of the order selections at once, our heuristic selected orders and assigned lots one order at a time. By doing so, the feasibility of selecting an order could be efficiently determined, without any degradation in solution quality. In other words, we could ensure that the specific order would fit within the factory capacity, and that a sufficient number of dies existed in the lot warehouse to completely fill the order. Once an order is selected, it is completely filled by lots, one class at a time, using one of the second-stage matching algorithms. After covering the order in Stage 2, the heuristic would return to the first stage where the next order is chosen. Unlike the Stage 1 sorting

functions, differences between each of the four Stage 2 matching algorithms are much more substantial. The next four sections will describe each of the matching algorithms.

2.2.1. First-in, first-out (FIFO)

Like FIFO for Stage 1, this baseline algorithm uses the arrival time (of the lots instead of orders) as its differentiator. If an order has a Class A die requirement, it gets filled by the first Class A lot that arrived at the lot warehouse. If the Class A order requirement is not covered, the lot that arrived next would be assigned to that order. This process continues until the number of assigned die meets or exceeds the number of die required in the order. The same process is applied to each of the classes in the order. The steps of the FIFO Stage 2 matching algorithm are shown below.

Stage 2 FIFO Matching Algorithm

- Step 1. Separate lots by class. Set k to initial class.
- Step 2. Sort class k lots in FIFO order.
- Step 3. If order does not require class k dies, set k to next class and go to *Step 2*.
- Step 4. Select the next unassigned class k lot.
- Step 5. Assign the class k portion of lot to order.
- Step 6. If the class k portion of order is not full, go to *Step 4*.
- Step 7. If k is not last class, set k to next class and go to *Step 2*.
- Step 8. Return to *Step 2* of *Stage 1* order selection.

2.2.2. First fit decreasing (FFD)

The origin of the first fit decreasing algorithm (FFD) used in this paper is (Knutson 1998, Knutson *et al.* 1999, Fowler *et al.* 2000, Carlyle *et al.* 2001), where it was referred to as FFD(S_i). The FFD(S_i) heuristic is based on the first fit decreasing algorithm described by Dyckhoff (1990) and was determined to yield the best results among four heuristics tested in (Knutson 1998, Carlyle *et al.* 2001).

For each Stage 2 order covering, the FFD heuristic splits the lots into two classes, with each group of lots arranged in order of non-increasing size, based on the number of class A or B die in the lot, respectively. Because we assume we can split the lots by class-type and send one to the assembly and test facility, leaving the other behind, we can treat each occurrence of stage 2 as two individual sets of bin covering problems, each processed separately. In both, the number of die in each lot represents the item size and the number of die required by the customer order represents the bin size.

By defining LEFT as the sum of the unassigned portion of the order and the smallest available lot, we simply select the largest available lot that does not exceed LEFT. After each lot assignment, LEFT is recalculated and the selection process begins again at the top of the list. This continues until we have covered the class A portion of the order. The process is then repeated for the class B portion of the order. Of course, if an order does not require one of the classes, that portion of stage 2 is omitted. Although not optimal, Knutson *et al.* (1999) deemed this modified FFD algorithm ‘sufficiently reliable’ for the bin covering problems. The second stage FFD algorithm is implemented immediately after an order is chosen during Stage 1. The steps are outlined below.

Stage 2 FFD Matching Algorithm

- Step 1. Separate lots by class. Set k to initial class.
- Step 2. Sort class k lots in non-increasing order by size.
- Step 3. If order does not require class k dies, set k to next class and go to *Step 2*.
- Step 4. Select the next unassigned class k lot.
- Step 5. Determine the size of the smallest available class k lot.
- Step 6. Define LEFT as the sum of the smallest lot and unassigned portion of the order.
- Step 7. If class k portion of lot does not fit in LEFT, go to *Step 4*.
- Step 8. Assign the class k portion of lot to order.
- Step 9. If the class k portion of order is not full, go to *Step 4*.

Step 10. If k is not last class, set k to next class and go to *Step 2*.

Step 11. Return to *Step 2* of *Stage 1* order selection.

2.2.3. First fit decreasing with improved endgame (FFD/IEG)

The FFD/IEG algorithm is a modified version of the FFD algorithm from Section 2.2.2 where the last part of the lot-to-order matching, or endgame, is improved. Rather than completing the second stage by finding the one lot that finishes the covering, the Improved Endgame (IEG) attempts to find the pair of lots that most precisely meet the covering requirement. This modification is designed to improve the selection process once the

As with the FFD algorithm, the order is sequentially filled with the largest lots of a particular class. Before adding subsequent lots, the algorithm checks to see if the two largest lots can cover the order. If they can, the FFD/IEG enters the endgame portion of the algorithm. Otherwise, the algorithm proceeds according to FFD, matching the next largest lot to the order.

Once the procedure reaches the so-called endgame, the two smallest lots are identified. The two smallest lots are both assigned if they can cover the order because there is no better pair, with regards to minimizing the number of wasted dies (DTW). If, on the other hand, they can not cover the order, a more complicated procedure is utilized to identify the final two lots. The next few paragraphs provide a description of this procedure, followed by a list of the specific steps used in the algorithm.

Depending on how the *attempts* and *tolerance* variables are set, the complexity of the procedure can vary from accepting the initial feasible cover-pair to a complete enumeration of all possibilities. The *attempts* variable, which was set at ten for this paper, ensures that at least ten feasible cover-pairs are checked before exiting the

algorithm. A feasible cover-pair is a set of two lots that successfully complete the covering of an order. Of course, a perfect cover, resulting in no excess dies ($DTW = 0$), or less than ten feasible cover-pairs, would result in an exit from the algorithm prior to reaching the prescribed attempt minimum. The *tolerance* variable, which was set to 25 dies, specifies the acceptable number of DTW at the end of the ten attempts. If the best cover-pair has not met the acceptable tolerance at the end of the attempts minimum, the procedure continues until the tolerance is met or complete enumeration has occurred.

The algorithm begins by checking to see if the smallest available lot is a *candidate* lot. Specific lots are considered *candidate* lots if the sum of the specified lot and the largest lot can successfully cover the order. Otherwise, the specified lot is labeled a *non-candidate*. Obviously, any order-pair consisting of a *non-candidate* lot can not be part of a feasible solution and therefore does not need to be checked, thus providing a significant computational time-savings.

If the smallest lot is a *candidate* lot, the algorithm proceeds up the list, searching for the first lot that, when combined with the *candidate* lot, covers the order. Because the lots are ordered by size, the initial feasible cover-pair is also the best cover-pair for that individual *candidate* lot. On the other hand, if the smallest lot is a *non-candidate* lot, the algorithm continues to move to the next-largest available lot until it finds a *candidate* lot. It should be intuitive that once a *candidate* lot is found, all subsequent lots will also be *candidates*. Because each *candidate* lot renders a feasible solution, ten *candidate* lots must be screened before the *attempts* minimum is reached, unless there is a perfect fit.

Stage 2 FFD/IEG Matching Algorithm

Step 1. Separate lots by class. Set k to initial class. Set *Tol* to 25.

Step 2. Sort class k lots in non-increasing order by size.

- Step 3. If order does not require class k dies, set k to next class and go to *Step 2*.
- Step 4. Select the next unassigned class k lot.
- Step 5. If two largest unassigned class k lots can cover the order, go to *Step 7*.
- Step 6. Assign the class k portion of lot to order and go to *Step 4*.
- Step 7. If two smallest unassigned class k lots can not cover the order, go to *Step 9*.
- Step 8. Label two smallest unassigned lots as $Best1$, $Best2$ and go to *Step 22*.
- Step 9. Label two largest unassigned lots as $Best1$, $Best2$; $BestDev$ to resultant DTW.
- Step 10. Select the smallest unassigned class k lot. Label it $Cur1$. Set $Iter = 0$.
- Step 11. If $Cur1$ is the second largest lot, go to *Step 22*.
- Step 12. If sum of $Cur1$ and largest lot can cover order go to *Step 14*. Set $Iter = Iter + 1$.
- Step 13. Select the next largest unassigned class k lot, label it $Cur1$ and go to *Step 11*.
- Step 14. Find smallest unassigned lot that can cover order with $Cur1$. Label it $Cur2$.
- Step 15. Set $CurDev$ to the $Cur1/Cur2$ DTW value.
- Step 16. If $CurDev = 0$, set $Best1 = Cur1$, $Best2 = Cur2$, $BestDev = 0$. Go to *Step 22*.
- Step 17. If $CurDev = BestDev$, go to *Step 19*.
- Step 18. Set $Best1 = Cur1$, $Best2 = Cur2$, $BestDev = CurDev$.
- Step 19. If $Iter < 10$, label next largest unassigned class k lot $Cur1$ and go to *Step 11*.
- Step 20. If $BestDev = Tol$, go to *Step 22*.
- Step 21. Label next largest unassigned class k lot $Cur1$ and go to *Step 11*.
- Step 22. Assign the class k portion of $Best1$ and $Best2$ lots to order.
- Step 23. If k is not last class, set k to next class and go to *Step 2*.
- Step 24. Return to *Step 2* of *Stage 1* order selection.

2.2.4. First-in, first-out with improved endgame (FIFO/IEG)

Realizing that the improved endgame works best when there is a large range of sizes, the fourth Stage 2 algorithm attempts to capitalize on the randomness of the lot arrivals. While improvements in the endgame performance are possible, FIFO/IEG will certainly enable the manufacturers to keep their product from stagnating—purging the oldest lots from the lot warehouse first.

The FIFO/IEG is very similar to both the FIFO and FFD/IEG second stage algorithms. Like the FIFO algorithm, the lot selections are made in FIFO order from the start. Instead of continuing FIFO selections until the order is completely covered, FIFO/IEG modifies its selection process and utilizes the improved endgame steps discussed in Section 2.2.3. A description of FIFO/IEG is highlighted below.

In order to maintain a steady pool of small lots, the algorithm does not allow a section of smaller lots to be selected during the FIFO phase of matching. In this paper, ten lots are considered unavailable as long as there are at least twenty lots available. As the number of available lots decreases from twenty, one less lot is excluded from use. Once there are less than eleven lots available, the default number of banned lots remains at one. To see how the number of hidden lots is a function of available lots, see table 2.

<i>Available Lots</i>	<10	10	11	12	13	14	15	16	17	18	19	20	21	>21
<i>Hidden Lots</i>	1	1	1	2	3	4	5	6	7	8	9	10	10	10

Table 2: Hidden lots as a function of available lots

The FIFO portion of the FIFO/IEG algorithm is employed to assign the lots until the two largest available lots can successfully cover the order. Once this occurs, the lots are rearranged back into non-increasing order and the improved endgame completes the matching just as it did in FFD/IEG. The same process is then applied to each of the remaining classes in the order. The steps of the FIFO/IEG algorithm are shown below.

Stage 2 FIFO/IEG Matching Algorithm

- Step 1. Separate lots by class. Set k to initial class. Set Tol to 25.
- Step 2. Sort class k lots in non-increasing order by size.
- Step 3. If order does not require class k dies, set k to next class and go to *Step 2*.
- Step 4. Identify the *hidden* lots that are not allowed to be used during FIFO.
- Step 5. Sort class k lots in FIFO order.

- Step 6. Select the next non-hidden unassigned class k lot.
- Step 7. If two largest unassigned class k lots can cover the order, go to *Step 9*.
- Step 8. Assign the class k portion of lot to order and go to *Step 6*.
- Step 9. If two smallest unassigned class k lots can not cover the order, go to *Step 11*.
- Step 10. Label two smallest unassigned lots as $Best1, Best2$ and go to *Step 24*.
- Step 11. Label two largest unassigned lots as $Best1, Best2; BestDev$ to resultant DTW.
- Step 12. Select the smallest unassigned class k lot. Label it $Cur1$. Set $Iter = 0$.
- Step 13. If $Cur1$ is the second largest lot, go to *Step 24*.
- Step 14. If sum of $Cur1$ and largest lot can cover order go to *Step 16*. Set $Iter = Iter + 1$.
- Step 15. Select the next largest unassigned class k lot, label it $Cur1$ and go to *Step 13*.
- Step 16. Find smallest unassigned lot that can cover order with $Cur1$. Label it $Cur2$.
- Step 17. Set $CurDev$ to the $Cur1/Cur2$ DTW value.
- Step 18. If $CurDev = 0$, set $Best1 = Cur1, Best2 = Cur2, BestDev = 0$. Go to *Step 24*.
- Step 19. If $CurDev = BestDev$, go to *Step 21*.
- Step 20. Set $Best1 = Cur1, Best2 = Cur2, BestDev = CurDev$.
- Step 21. If $Iter < 10$, label next largest unassigned class k lot $Cur1$ and go to *Step 13*.
- Step 22. If $BestDev = Tol$, go to *Step 24*.
- Step 23. Label next largest unassigned class k lot $Cur1$ and go to *Step 13*.
- Step 24. Assign the class k portion of $Best1$ and $Best2$ lots to order.
- Step 25. If k is not last class, set k to next class and go to *Step 2*.
- Step 26. Return to *Step 2* of *Stage 1* order selection.

3. Experimental design

The experimental design was created using Design Expert 6.0 (2000) with three factors and four responses. The first two factors have four levels each and are described in detail in Section 2. They are designated ‘Stage 1 Policy’ and ‘Stage 2 Policy’ and are considered categorical variables. The third factor is also a categorical variable and it provides two factory environments in which the simulations were run. Greater details on the ‘Factory Setting’ factor can be found in Section 3.1.3.

At each of the 32 design points, ten replicates were run; the second five using antithetic variables. The same random number seeds were used for each of the 32 first runs. A different set of random number seeds were consistently used for a specific run number across all 32 design points. Because the variables are all categorical in nature, there is no need to include center points to check for curvature in the response. In fact, the categorical variables prevented the attainment of a single equation to describe the differences in the response variables relative to the factor levels. The test matrix is shown in table 3 with descriptions of the factors and responses in Sections 3.1 and 3.2.

	Factor A	Factor B	Factory C
Runs	Stage 1 Policy	Stage 2 Policy	Factory Setting
1-10	1	1	1
11-20	2	1	1
21-30	3	1	1
31-40	4	1	1
41-50	1	2	1
51-60	2	2	1
61-70	3	2	1
71-80	4	2	1
81-90	1	3	1
91-100	2	3	1
101-110	3	3	1
111-120	4	3	1
121-130	1	4	1
131-140	2	4	1
141-150	3	4	1
151-160	4	4	1
161-170	1	1	2
171-180	2	1	2
181-190	3	1	2
191-200	4	1	2
201-210	1	2	2
211-220	2	2	2
221-230	3	2	2
231-240	4	2	2
241-250	1	3	2
251-260	2	3	2
261-270	3	3	2
271-280	4	3	2
281-290	1	4	2
291-300	2	4	2
301-310	3	4	2
311-320	4	4	2

Table 3: Experimental design run matrix

3.1. Factors

A summary of the experimental design factors are shown in table 4. The first two factors each have four different levels, while the third factor only has two.

Factor	Level 1	Level 2	Level 3	Level 4
A - Stage 1 Policy	FIFO	TWT	ATC	EDD
B - Stage 2 Policy	FIFO	FFD	FFD/IEG	FIFO/IEG
C - Factory Settings	EDD, FFD/IEG	EDD, FIFO/IEG	-	-

Table 4: Summary of experimental design factors

3.1.1. Factor A – Stage 1 sorting functions

This factor is a categorical variable and identifies which of the four Stage 1 sorting functions to implement. Described in section 2.1, these sorting functions dictate how the customer orders are valued, so they can be input into the Stage 1 algorithm.

3.1.2. Factor B – Stage 2 matching algorithm

This factor is also a categorical variable and identifies which of the four Stage 2 matching algorithms to implement. Described in section 2.2, these matching algorithms dictate how the lots get assigned to the customer orders.

3.1.3. Factor C – Factory settings

This factor is also a categorical variable and identifies which of the two factory settings to use during the simulations. Each of the factory settings used a specific combination of a Stage 1 sorting function and a Stage 2 matching algorithm to arrive at appropriate lot and order production settings. These settings were tested to make sure the results were realistically good—namely an on-time delivery rate of around 80%. Using trial and error, the first factory setting was designed to get good results using the fourth Stage 1 sorting function and third Stage 2 matching algorithm. The second factory

setting also used the fourth Stage 1 sorting function, but instead of the third, it used the fourth Stage 2 matching algorithm. The only settings that changed were the production rate of the lots and orders. Because of the desire to maintain similar results when implementing the FFD/IEG and FIFO/IEG matching functions, the factory settings for the daily production of lots and orders had to be modified. On average, the daily number of lots produced increased from 46.2 to 46.5 as we moved from the first to the second set of factory settings. The same trend continued with the number of orders produced, where the daily average increased from 9.0 to 9.2 orders as we moved from the first to the second set of factory settings. Histograms of the daily lot and order production distributions used are shown in Figures 2 and 3, respectively.

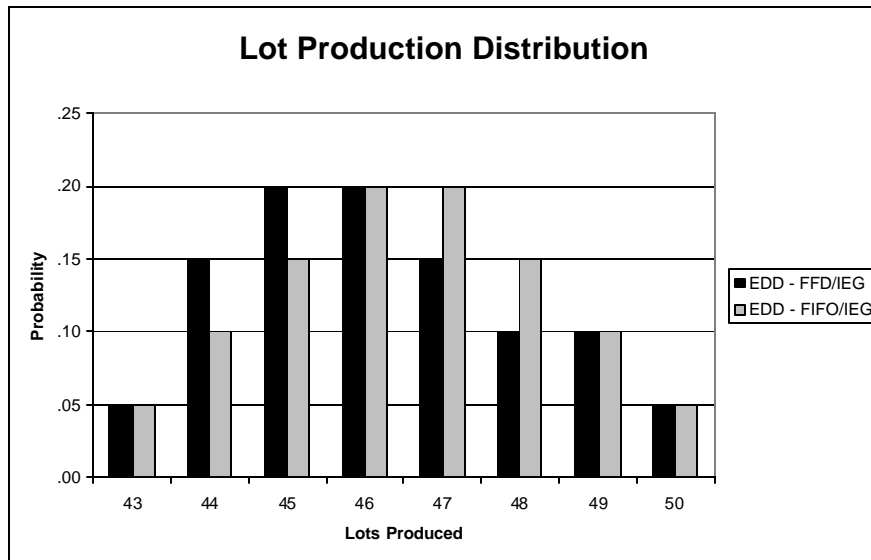


Figure 2: Daily lot production distribution

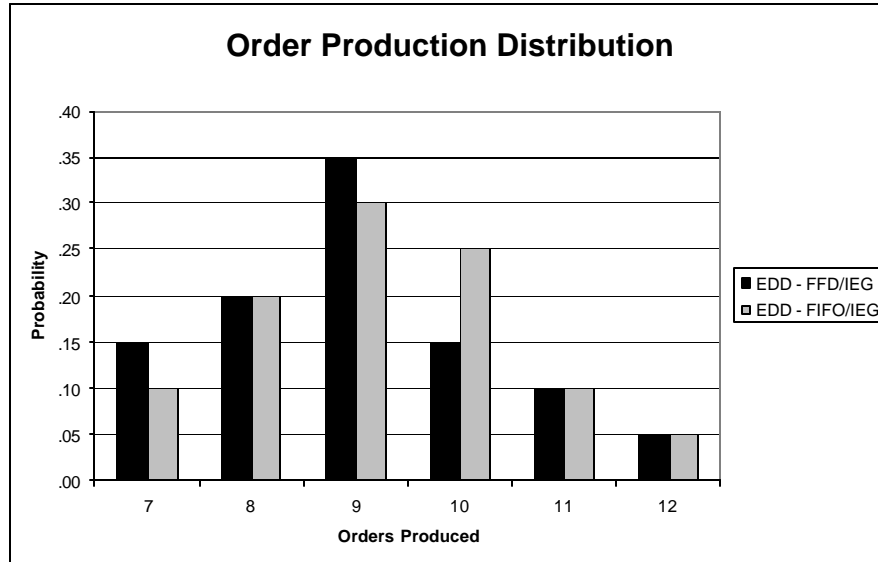


Figure 3: Daily order production distribution

3.2. Responses

In order to evaluate the quality of the first stage sorting functions and the second stage matching algorithms, four measures of effectiveness are analysed from the simulation runs. The four responses that are of interest are total weighted tardiness (TWT), on-time delivery (OTD), dies to order (DTO) and dies to warehouse (DTW). The first three objectives (TWT, OTD and DTO) are all affected by both the first and second stage policies. The final objective (DTW) is only influenced by the second stage matching algorithm. All four objectives are described in the following sections.

3.2.1. Total weighted tardiness (TWT)

The only objective of the four that was not researched in Knutson *et al.* (1999) was total weighted tardiness (TWT). A typical scheduling measure of effectiveness, TWT is calculated using the product of the individual customer's importance, the size of the order and the number of days late that the order is delivered. Consequently, the TWT of an order gets larger each day a late order is not delivered. If, however, the order is

delivered on-time or early, the TWT for the order is zero. Because an early delivery is treated the same as one that arrives on its due-date, it is important not to satisfy an order too early, especially at the expense of one that is nearly late. As one would expect, lower values for TWT are desired.

3.2.2. On-time delivery (OTD)

The second objective, OTD, reports the percentage of orders that are delivered to the customer on or before its due-date. An order is said to be delivered only after all of its assigned lots from each class have completely finished processing in the assembly and test facility. Obviously, higher on-time delivery rates are preferred.

3.2.3. Dies to order (DTO)

The third objective, DTO, measures the number of dies that are delivered to the customer each day, as a percentage of the 500 000 die factory capacity. As with OTD, maximization of DTO is desired, as it measures how efficiently the factory is being utilized.

3.2.4. Dies to warehouse (DTW)

The final objective, DTW, is perhaps the most important measure of the four because its performance has a significant impact on the rest of the objectives. By assessing the number of excess dies that get sent to the warehouse, DTW identifies how good each of the second stage matching algorithms are. If too many dies are assigned to an order, the excess dies are wasted. Additionally, the processing time that the excess dies took away from dies that are required for other orders was also wasted. Minimizing DTW is desired because it not only makes the factory more efficient by processing a

higher percentage of dies that are used towards orders, it also reduces the cost of handling the excess dies.

4. Simulation model description

The modeling details and specific assumptions are reviewed in this section. After making the runs, the data was imported in Design Expert 6.0 where the results were analysed. Details regarding the four responses are discussed in Sections 5.1 – 5.4.

4.1. Modeling details

In order to evaluate the effectiveness of the four Stage 1 sorting functions and four Stage 2 matching algorithms, a simulation model was created using the computer simulation language SLAM II (Pritsker 1995). Each day, the model generated new lots and orders, which served as inputs to the ‘optimization’ model. Coded in FORTRAN, the ‘optimization’ model produced the lot-to-order assignments using the sorting functions and matching algorithms described in Section 2. Once the lots were matched to the orders, the lots were returned to the simulation model where they were processed through a model of the assembly and test facility.

The model of the assembly and test facility sent each lot through a series of nine separate processes. The individual lots had associated load, processing and unload times for each of the nine steps. When an individual lot reached the next process, it waited for one of the process’ limited resources to be freed. Two of the nine processes also required a minimum number of lots to be batched together before processing could occur. Additionally, preventative maintenance and machine breakdowns were modeled to take away resources at various rates. Once all of the lots assigned to a specific order made it through the assembly and test facility, the order was delivered to the customer.

Ten runs for each design point were simulated over the course of 1180 days, with the first 100 days serving as the warm-up period. Measures of effectiveness were recorded for the last 1080 days (~ 3 years) of each run. In order to reduce unwanted variability in our performance metrics, antithetic variates were used in conjunction with common random number streams for each of the two policies (Pritsker 1995).

4.2. Assumptions

We assume that the long-term percentage of high-speed (class A) integrated circuits in a lot, π_1 , is equal to 60%. Accordingly, the long-term percentage of low-speed (class B) die is $(1-\pi_1)$ or 40%. In this paper, the mean number of dies produced is 10 000 per lot, which results in high- and low-speed ICs production means of 6000 and 4000 dies per lot, respectively. The total lot size is modeled with a triangular distribution having lower, mode and upper values equal to 8500, 10 500 and 11 000, respectively. The number of class A dies in the lot is calculated by multiplying the total size of the lot by a random variate attained from a uniform distribution of $60\% \pm 10\%$. The remaining dies are considered class B or low-speed dies.

It is assumed that the company will price the different classes to ensure that the lots produced are sold. Because the average number of class A dies in lots are larger, the average DTW for class A dies turns out to be larger than the DTW for class B dies. Consequently, maintaining equilibrium between dies produced and sold requires a demand of slightly less than 60% class A dies. In this research, it is assumed that the companies' pricing policies maintain a constant class A demand of 59.5% of the dies. This is accomplished by dividing the customer orders into two types—half that contain a

mix of both classes and half that contain only a single class. Both mixed and homogeneous orders, however, maintain a mean of 50 000 die per order.

The mixed orders are comprised of approximately 59.5% and 40.5% high- and low-speed die. The total size of an order is drawn from a normal distribution with a mean of 50 000 and standard deviation equal to 5% of the mean or 2500, rounded to the closest integer. The determination of the mixed order class-sizes is similar to what is used for determining the mix of dies in the lots. The portion of class A dies in an order comes from the uniform distribution, $59.5\% \pm 10\%$, with the remaining dies labeled class B.

Both types of homogeneous orders are drawn from a normal distribution with a mean of 50 000 and a standard deviation of 2500 rounded to the closest integer. In order to maintain the 59.5/40.5 break, 59.5% of all homogeneous orders are for class A die and 40.5% are for class B die. Additionally, each customer order is given a random due-date from the current date (in days) and a customer priority from the distributions outlined in tables 5 and 6 below.

D_i	19	21	23
$P(D_i)$	0.33	0.34	0.33

W_i	1.0	1.2	1.4
$P(W_i)$	0.75	0.20	0.05

Table 5: Due-date (D_i) probabilities

Table 6: Customer weight (W_i) probabilities

Each day prior to matching lots to orders, additional lots and orders are generated in order to approximately match the long-term usage of each. The average number of lots and orders created each day depended on the third factor in the experimental design. Recall from Section 3.1.3 that the long-term average number of lots produced were either 46.2 or 46.5 lots per day, depending on which factory setting was used. The average number of orders produced also depended on the factory setting, with the production of either 9.0 or 9.2 orders per day, respectively. At the start of the simulation, 30 orders and

100 lots were created. Finally, we limit the number of dies that could be assigned to the factory to 500 000.

5. Results

As expected, the results from the simulations indicate how effective the fourth Stage 2 matching algorithm—FIFO/IEG is compared to the others. While this matching algorithm was believed to be the best prior to the simulations, the only expectation regarding the Stage 1 sorting functions was that FIFO would not be the best. This turned out to be true; however, the fact that ATC performed *significantly* better than the other three sorting functions was not anticipated. The four responses are analysed in Sections 5.1 – 5.4, each including an interaction graph from the ‘Analysis’ tab in Design Expert 6.0 (2000). Overall conclusions about the research in this paper are found in Section 5.5.

5.1. Total weighted tardiness (TWT) results

Two major features should be visible from the TWT interaction graph in Figure 4. First, a substantial dip in TWT occurs when the first stage uses the apparent tardiness cost (ATC) sorting function. By simply changing the Stage 1 sorting function to ATC, the TWT of the FIFO Stage 2 matching algorithm decreases more than 83% from over 1 500 000, using any of the other three Stage 1 sorting functions, down to fewer than 250 000. While we are 95% confident that ATC improves the TWT performance for each of the first three second stage matching algorithms, there is not enough evidence to suggest that the means for those involving the FIFO/IEG matching function are significantly different at $\alpha = 0.05$.

The second major revelation that is revealed in Figure 4 actually is more important because it spans virtually all of the different combinations of policies. With

just one exception, the Stage 2 FIFO/IEG matching algorithm outperforms every other combination. The only case where we are not 95% confident is when FIFO/IEG is compared to the ATC – FFD/IEG combination. Because of this near-perfect dominance, the decision to implement the FIFO/IEG matching algorithm provides us with the best TWT results, regardless of the first stage sorting function used. That being said, the ideal combination seems to be ATC and FIFO/IEG, although there is not enough evidence to claim a statistically significant difference between the four FIFO/IEG results and the ATC – FFD/IEG combination.

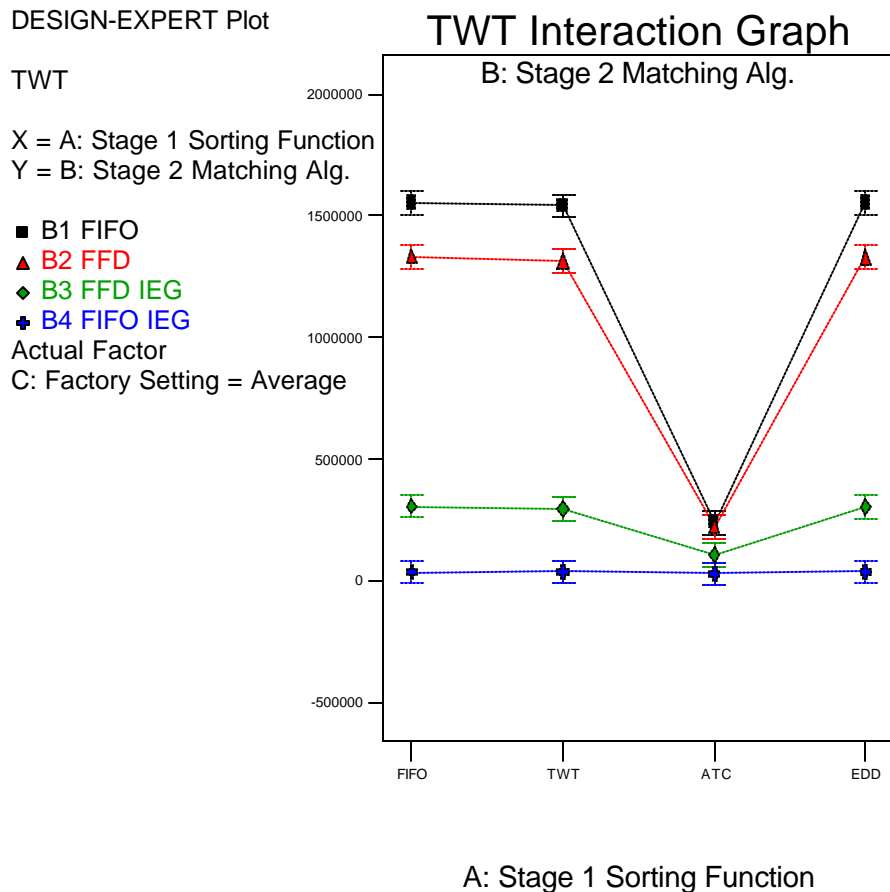


Figure 4: TWT interaction graph

5.2. On-time delivery (OTD) results

Once again, two significant features should jump out from the OTD interaction graph in Figure 5. Fortunately, they both exhibit similar trends from the same policy combinations as did the TWT. First, no matter which Stage 2 matching algorithm is implemented, the utilization of ATC yields the highest on-time delivery percentage of all the sorting functions. The least significant difference (LSD) bars at each of the points show that we are 95% confident that the average OTD values for the two IEG algorithms, when combined with ATC, are at least 85%. While the LSD bars for the other two matching algorithms (FIFO and FFD) never reach 50% OTD with non-ATC sorting functions, their confidence intervals barely dip below 80% when ATC is employed.

If the Stage 1 sorting function, on the other hand, was discounted, it should be obvious that the FIFO/IEG matching algorithm alone provides excellent OTD performance throughout. The LSD bars reveal that while the ATC Stage 1 sorting function does enhance the OTD performance, the employment of FIFO/IEG does most of the heavy lifting. The 95% confidence intervals fall beneath the 80% level for the TWT and EDD sorting functions, but just barely.

While both the ATC sorting function and the FIFO/IEG matching algorithm resulted in very good OTD performance individually, all indications point to the combination of the two being the ideal policy.

DESIGN-EXPERT Plot

OTD

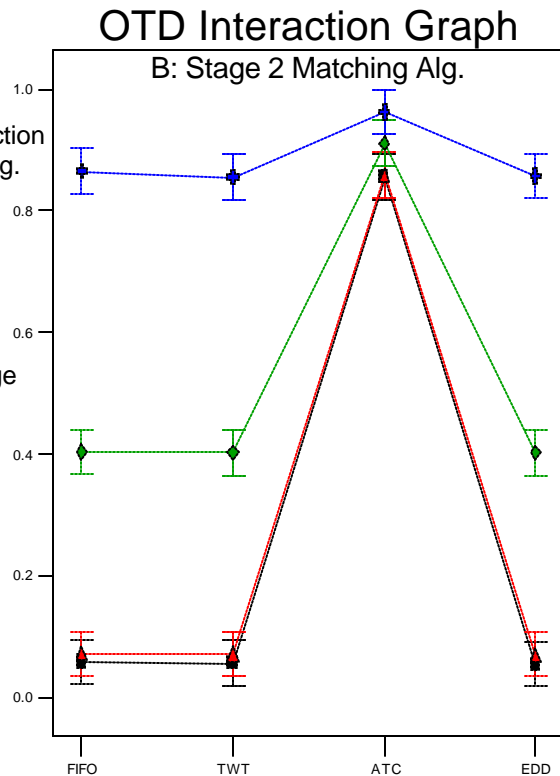
X = A: Stage 1 Sorting Function

Y = B: Stage 2 Matching Alg.

- B1 FIFO
- ▲ B2 FFD
- ◆ B3 FFD IEG
- ⊕ B4 FIFO IEG

Actual Factor

C: Factory Setting = Average



A: Stage 1 Sorting Function

Figure 5: OTD interaction graph

5.3. Dies to order (DTO) results

Unlike the previous two figures, only one noteworthy feature emerges from the DTO interaction graph in Figure 6. The percentage of dies to order does not vary greatly—the range from the highest to the lowest LSD bar approaches 6%. However, a 5% improvement in DTO amounts to around 25 000 dies per day, which is half an order. Therefore, significant improvements can be attained by employing the correct combination of policies.

As opposed to the two previous responses, DTO does not exhibit statistically significant differences between the four Stage 1 sorting functions. But once again, the choice of second stage matching algorithms can result in substantial benefits, with FIFO/IEG retaining its status as the preeminent matching algorithm.

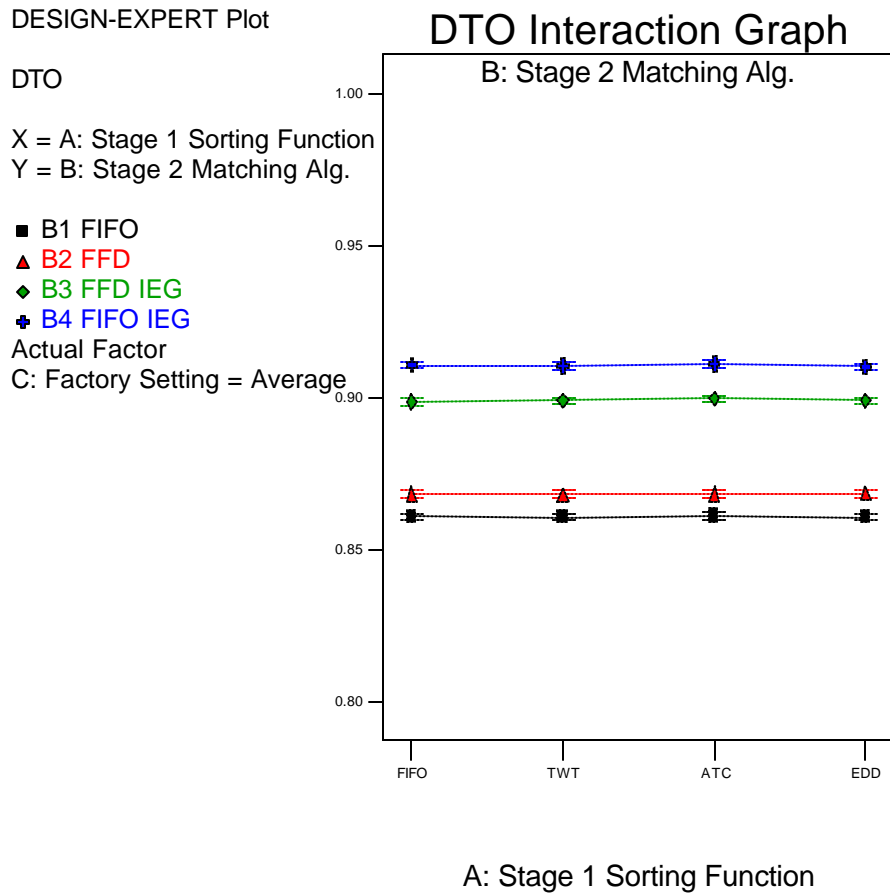


Figure 6: DTO interaction graph

5.4. Dies to warehouse (DTW) results

The final interaction graph depicting DTW also yields just a single notable result. As with the previous three objectives, FIFO/IEG surpassed the other three Stage 2

matching algorithms. With regards to DTW, it outperformed the second best algorithm by nearly a factor of five and the other two by an order of magnitude, regardless of the first stage sorting function used. Indeed, this dominating performance was anticipated, because the improved endgame was designed to make much more precise matches at the end of the matching process. With the inclusion of the random element at the beginning of the matching algorithm, the FIFO/IEG version is able to outdo FFD/IEG because the greater range in the size of available lots is maintained.

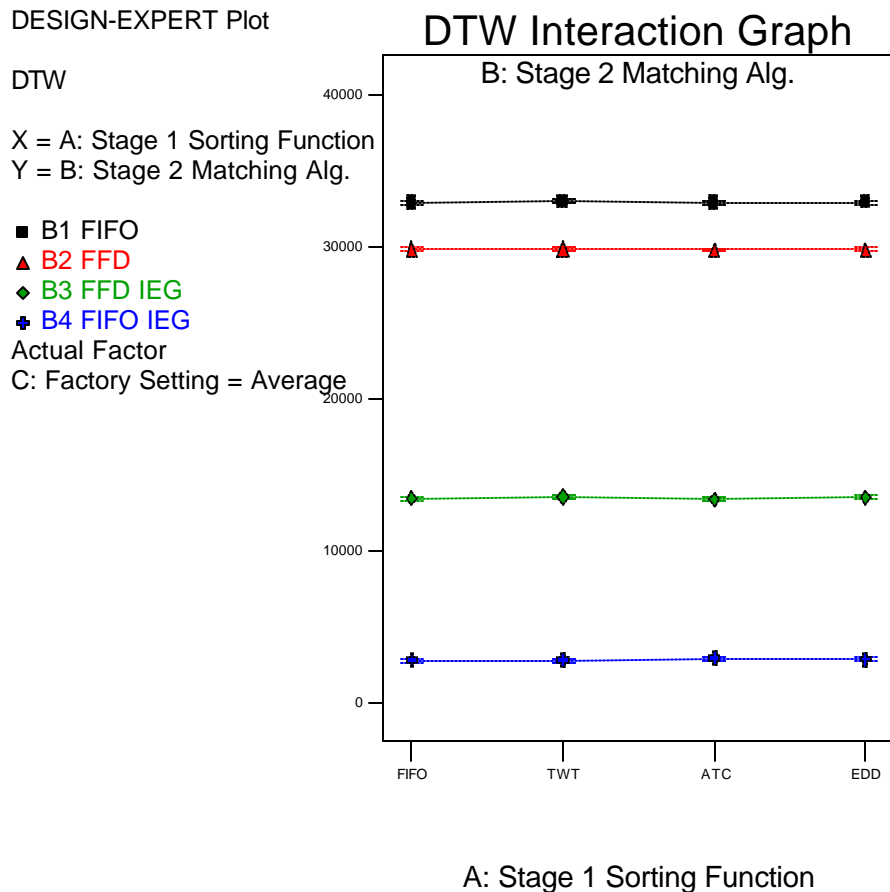


Figure 7: DTW interaction graph

5.5. Conclusions and future work

Results provided in the previous four sections, indicate that the four measures of effectiveness can be greatly influenced by the selection of the proper Stage 1 and Stage 2 policies. The ability to more accurately assign lots to orders provides improvements beyond the obvious reduction in the number of excess dies. The ability to limit the number of wasted dies has positive repercussions that are felt across the other three objectives. Not only do customers receive better service, the semiconductor industry is able to limit their production costs or even reallocate superfluous resources to other endeavors.

It has become apparent that the individual employment of either the ATC sorting function or the FIFO/IEG matching algorithm will provide great improvement to the four objectives. The combination of the two, however, results in unsurpassed performance.

Future research could include the modifications of various input parameters, like customer weighting values, the percentage of mixed versus homogeneous orders, and the distributions of the lots and orders. Additionally, research could be done on how the problem changes if all classes within a selected lot are forced to be used if any part of the lot is assigned. Among other possibilities, the consideration of a 'rolling' factory capacity or a planning horizon of more than one day could also be looked into, as well as changes in how the assembly and test facility is modeled. One final research topic that could be explored is product substitutability, where high-performance dies could be used towards satisfying a low-performance die requirement.

References

- Boushell, T., Fowler, J., Knutson, K. and Keha, A., 2005. Class-constrained lot-to-order matching problem for a semiconductor assembly and test facility. Arizona State University Industrial Engineering Working Paper ASUIE-ORPS-2005-004.
- Carlyle, M., Knutson, K. and Fowler, J., 2001. Bin covering algorithms in the second stage of the lot to order matching problem. *Journal of the Operational Research Society*, **52**, 1232-1243.
- Design Expert 6.0, 2000, Stat-Ease, Inc., 2021 East Hennepin Ave., Suite 191, Minneapolis, MN 55413, USA.
- Dowsland, K. and Dowsland, W., 1992. Packing problems. *European Journal of Operations Research*, **56**, 2-14.
- Dyckhoff, H., 1990. A typology of cutting and packing problems, *European Journal of Operations Research*, **44**, 145-159.
- Fowler, J., Knutson, K. and Carlyle, M., 2000. Comparison and evaluation of lot-to-order matching policies for a semiconductor assembly and test facility. *International Journal of Production Research*, **38**, 1841-1853.
- Frederix, F., 1996. Planning and scheduling multi-site semiconductor production chains: A survey of needs, current practices and integration issues, in *Advances in Design and Manufacturing*, v7, *IT and Manufacturing Partnerships*, Browne J, Haendler Mas R, Hlodversson O (eds), IOS Press, Amsterdam, 107-116.
- Knutson, K., 1998. A two-stage decomposition of the lot-to-order matching problem. Doctorial Dissertation, Arizona State University.
- Knutson, K., Kempf, K., Fowler, J. and Carlyle, M., 1999. Lot-to-order matching for a semiconductor assembly and test facility. *IIE Transactions*, **31**, 1103-1111.
- Pinedo, M., 1995. *Scheduling: Theory, Algorithms and Systems*. Prentice-Hall, Inc, Englewood Cliffs, New Jersey, USA.
- Pritsker, A., 1995. *Introduction to SLAM II*. John Wiley & Sons, New York.
- Semiconductor Industry Association, 2004. Economy. Available online at: http://www.sia-online.org/pre_facts.cfm (accessed 13 April 2005).
- Semiconductor Industry Association, 2004. Industry Statistics. Available online at: https://www.sia-online.org/downloads/market_shares_94-present.pdf (accessed 11 April 2005).