CHAPTER 5: SCHEDULING ALGORITHMS AND IMPROVEMENT METHODS

The following chapter discusses the scheduling algorithms and improvement methods used to develop the production schedule for the chemical batch processing line.

5.1 Scheduling Algorithms

The scheduling algorithms were developed to schedule orders for processing through the first stage of production only. The others stages' production order is based on a first-in-first-out policy, which is solely dependent on the processing order at the first set of parallel machines.

5.1.1 Musier and Evans Scheduling Algorithm (1989)

The Musier and Evans algorithm was developed specifically for the following model:

1. Plant consists of *m* parallel [machines] or production lines of different capacities or processing rates.
2. A set of *p* product types may be produced on the *m* [machines], but each product type may only be produced on a specified subset of processes [machines].
3. A set of *n* customer orders must be filled. A given order *i* must be filled by a due date d_i, or a penalty $F_i(t_i)$ is incurred, where t_i is the tardiness of order *i*.
4. The objective function minimizes an arbitrary function of tardiness:
$$\sum_{i=1}^{n} F_i(t_i).$$
5. If a cleanout is required for the product sequence *i-j*, then a cleaning time must be scheduled between the processing of batches *i* and *j*.

It is assumed that no excess product is produced and batches/production runs are not split among several customers, i.e., there is a one-to-one correspondence between batches and customer orders.

Musier and Evans provide the following steps to generate schedules in which orders are produced in campaigns:

1. Each of the *p* machines and the *n* customer orders is assigned a temporary label equal to a random number selected over the interval (0,1).
2. The *p* machines are sorted according to the increasing label value, as are the *n* customer orders. This yields two sorted lists—*P* for machines and *N* for the orders.
3. Consider the lowest-labeled machine *i* in list *P*.
4. Each customer order on list *N* is checked in sequence to determine if it can be scheduled on machine *i* without tardiness. If such an order *j* is found, then the order is scheduled to follow all orders previously scheduled on machine *i*. Order *j* is removed from list *N*.
5. Machine *i* is removed from list *P*. If *P* is nonempty go to step 3.

6. At this point, no scheduled batch is tardy. If *N* is empty, then a zero tardiness schedule has been found. If *N* is nonempty, then the orders on list *N* are considered in earliest due date sequence and inserted one-at-a-time into the schedule to minimize the total tardiness of the schedule.

Creating schedules with different starting points allows for more local-optimal solutions to be found and hence allows for a higher probability of obtaining a global-optimal or near-optimal solution when the improvement method is applied.

Modifying the scheduling algorithm to fit the production line under consideration involved only developing the list of customer orders (i.e., the *N* list). The parallel machines do not need to be considered because all the products must be processed in the first stage and all machines within the stage may only process one product at a time. This resulted in the following modified Musier and Evans scheduling algorithm:

1. Each of the *n* customer orders is assigned a temporary label equal to a random number selected over the interval (0,1).
2. The *n* customer orders are sorted according to the increasing label value. This yields a sorted *N* list.
3. Each customer order on list *N* is checked in sequence to determine if it can be scheduled on machine *i* without tardiness. If such an order *j* is found, then the order is scheduled to follow all orders previously scheduled on machine *i*. Order *j* is removed from list *N*.
4. At this point, no scheduled batch is tardy. If *N* is empty, then a zero tardiness schedule has been found. If *N* is nonempty, then the orders on list *N* are considered in earliest due date sequence and inserted one-at-a-time into the schedule to minimize the total tardiness of the schedule.

The following is an example of how the modified Musier and Evans scheduling algorithm works.

Given:

Customer Number	Dehy ST* per dehy batch		
Type	1	2	3
Changeover	240	240	300
Dehy	15.52	15.52	15.52
Mixer	16.53	17	25.65
Blocker	25.4	25.4	25.4
Extruder	19.84	19.84	19.84
Cutter	13.14	13.14	13.14
Total (Mixer-Cutter)	74.91	75.38	84.03

* in minutes

Customer Number	Product Type	Due Date (hours)	Quantity
1	1	8	5
2	2	13	10
3	3	5	3
4	2	10	25
5	2	23	8

1 Assign a random number to each customer.

Customer Number	Product Type	Due Date	Quantity	Random Number
1	1	8	5	0.49
2	2	13	10	0.63
3	3	5	3	0.31
4	2	10	25	0.05
5	2	23	8	0.85

2 Sort random numbers according to increasing values.
This forms the *N* list.

Customer Number	Product Type	Due Date	Quantity	Random Number
4	2	10	25	0.05
3	3	5	3	0.31
1	1	8	5	0.49
2	2	13	10	0.63
5	2	23	8	0.85

3 Check each customer order on the *N* list to determine if it can be scheduled without tardiness.

N list	4	3	1	2	5

Processing time (Changeover + Dehy * Quantity + Total (Mixer-Cutter)) / 60 min/hr
(240 + 15.52 * 25 + 75.38) / 60 = 11.72 hr

Customer Number	Product Type	Due Date	Quantity	Total Processing Time	
4	2	10	25	11.72	late

Since Customer 4 cannot be scheduled without being late, it is removed from the partial sequence list and the next job in the *N list* is placed in the partial sequence list.

N list	4	3	1	2	5

Processing time (Changeover + Dehy * Quantity + Total (Mixer-Cutter)) / 60 min/hr
(300 + 15.52 * 3 + 84.03) / 60 = 7.18 hr

Customer Number	Product Type	Due Date	Quantity	Total Processing Time	
3	3	5	3	7.18	late

Since Customer 3 cannot be scheduled without being late, it is removed from the partial sequence list and the next job in the *N list* is placed in the partial sequence list.

N list	4	3	1	2	5

Processing time (Changeover + Dehy * Quantity + Total (Mixer-Cutter)) / 60 min/hr
(240 + 15.52 * 5 + 74.91) / 60 = 6.54 hr

Customer Number	Product Type	Due Date	Quantity	Total Processing Time
1	1	8	5	6.54

Since Customer 1 can be scheduled without being late, it is left on the partial sequence list and the next job in the *N list* is placed in the partial sequence list after Customer 1.

time (hours)

The next customer is checked to see if their order can be processed without being late.

Customer Number	Product Type	Due Date	Quantity	Total Processing Time	
1	1	8	5	6.54	
2	2	13	10	14.38	late

Since Customer 2 cannot be scheduled with out being late, it is removed from the partial sequence and the next customer is considered.

N list	4	3	2	5

Customer Number	Product Type	Due Date	Quantity	Total Processing Time
1	1	8	5	6.54
5	2	23	8	13.87

time (hours)

4 Consider remaining jobs in *N* list in earliest due date sequence.

N list	4	3	2

Customer Number	Product Type	Due Date	Quantity	Total Processing Time
1	1	8	5	6.54
5	2	23	8	13.87
3	3	5	3	21.04
4	2	10	25	32.77
2	2	13	10	35.35

time (hours)

The result is a schedule with the following customer order: {1, 5, 3, 4, 2} and this gives a total processing time of 35.35 hours.

5.1.2 Ku and Karimi's Sequence Building Algorithm (1991)

Ku and Karimi's scheduling algorithm requires an initial sequence. They develop the initial sequence based on an increasing order of due date/penalty. This algorithm involves *N* iterations, one for each of the customer orders. With each iteration, another order is inserted into the partial sequence so that it creates a new partial sequence that has the minimal tardiness

penalty. When all the orders have been inserted into the partial sequence, a good sequence (a feasible job sequence that minimizes tardiness penalty) has been formed. The Sequence Building Algorithm (Ku & Karimi 1991) is as follows.

> At the end of n iterations (n = 1,2,3,…,N), it produces a partial sequence of ψ_n of n products. For n = 1, it uses the first product from the list as ψ_1. In subsequent iterations (n = 2,3,…,N), it generates n n-product partial sequences by *inserting the* n^{th} product from the list at every position (1,2,3,…,n in that precise order) in ψ_{n-1}. During the insertion, the relative positions of the other products in ψ_{n-1} are kept unchanged. It then evaluates the penalty for each of these n partial sequences and selects the best of them as ψ_n. In case of multiple n-product sequences with the same minimum penalty, the one that was generated *first* is selected.

Two modifications were needed. Since penalties are not considered in this research, the first change was to develop the initial sequence based on an increasing order of due dates. The second modification occurred when the algorithm calls for an evaluation of the sequence penalty. For this research, the algorithm evaluated the overall tardiness of the sequence instead of the sequence penalty.

The following is an example of how the modified Ku and Karimi scheduling algorithm is applied to the same information for given in the previous example.

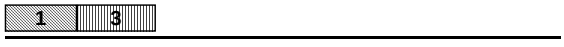
Customer Number	Product Type	Due Date	Quantity
3	3	5	3
1	1	8	5
4	2	10	25
2	2	13	10
5	2	23	8

2 Iteration 1: Place first job in partial sequence list and determine tardiness.

time (hours)

tardiness: (300 + 84.03 + 3*15.52)/60 - 5 = 2.18

3 Iteration 2: Place second job all postions in partial sequence list and determine respective tardiness.



tardiness: 8.72

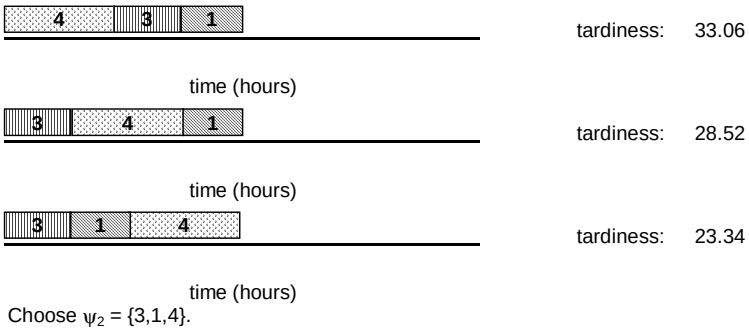
time (hours)

tardiness: 7.89

time (hours)

Keep partial sequence with least tardiness. Choose ψ_2 = {3,1}.

4 Iteration 3: Place third job all postions in partial sequence list and determine respective tardiness.



Choose ψ_2 = {3,1,4}.

The iterations are continued in a similar manner until all 5 jobs have been placed in such a way that the total tardiness is minimized. For this example, the resulting sequence is: {3, 1, 2, 5, 4} and this gives a total processing time of 30.10 hours and a total of 37.18 tardy hours.

5.1.3 Greedy Heuristic Algorithm

The greedy heuristic algorithm is based on an earliest-due-date policy. For orders with the same due dates, larger order quantities have priority. Step 1 of the Ku and Karimi example shows how this algorithm is applied. For this example, the resulting sequence is: {3, 1, 4, 2, 5} and this gives a total processing time of 30.10 hours and a total of 45.46 tardy hours.

5.2 Musier and Evans Heuristic Improvement Method

The Musier and Evans heuristic improvement method (HIM) (1989) is based on the "idea that 1-optimal and 2-optimal schedules are sufficiently close to the global optimum and can be found with a reasonable computational effort." Solutions that are 1-optimal occur "when removing a single batch from the schedule and replacing it in a new position" does not reduce the total tardiness of the schedule. Similarly, 2-optimal solutions occur when a pair-wise removal and insertion of batches does not improve the total tardiness of the schedule. Schedule variations on locally-optimal schedules, involving single batches and pairs of batches, are performed until no further improvement may be made to the objective function. For the 2-optimal solution, Musier and Evans only perform pair-wise interchanges of batch positions in order to reduce the computational effort by $O((n+m)^2)$ operations.

The following is an example of how to apply the HIM heuristic to the results of the earliest due date greedy heuristic.

Given: Final job sequence from EDD Greed Heuristic.

Customer Number	Product Type	Due Date	Quantity
3	3	5	3
1	1	8	5
4	2	10	25
2	2	13	10
5	2	23	8

1 Determine total tardiness of job sequence.

tardiness: 45.46

time (hours)

2 Move first job to all possible positions and determine repsective tardiness.

tardiness: 45.46

time (hours)

tardiness: 46.29

time (hours)

tardiness: 61.34

time (hours)

tardiness: 51.50

time (hours)

tardiness: 41.21

time (hours)

Select job sequence with minimum tardiness, if it is different from the initial sequence then repeat step 2. Otherwise, move the second job to all possible positions in the sequence and determine it's respective tardiness.

For this example, the sequence {1, 4, 2, 5, 3} has the minimum tardiness. Since it is different from the initial sequence, Step 2 is repeated with this new sequence.

tardiness: 41.21

time (hours)

tardiness: 56.23

time (hours)

tardiness: 51.42

time (hours)

tardiness: 43.05

time (hours)

tardiness: 43.69

time (hours)

For this example, the sequence {1, 4, 2, 5, 3} has the minimum tardiness. Since it is the same as the initial sequence, the next job in the sequence, 4, is placed in all posible positions to determine the the respective tardiness.

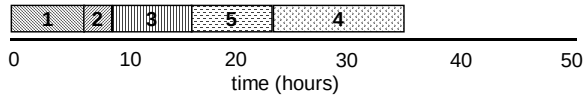
This process is continued until all jobs cannot be moved to another postion in the sequence without improving the total tardiness, in which case the job sequence has reached 1-optimal. For this example, the resulting sequence is: {1, 2, 3, 5, 4} and this gives a total processing time of 25.32 hours.

The 2-optimal algorithm is then applied to the resulting 1-optimal schedule. An example of the 2-optimal algorithm follows.

Given: The following job sequence is 1-optimal.

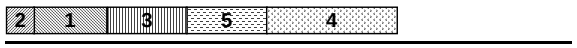
Customer Number	Product Type	Due Date	Quantity
1	1	8	5
2	2	13	10
3	3	5	3
5	2	23	8
4	2	10	25

1 Determine total tardiness of job sequence.



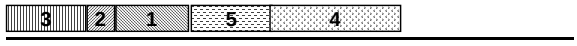
tardiness: 46.67

2 Exchange the position of the first job with the other jobs and determine the respective tardiness.



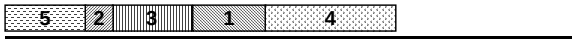
tardiness: 51.67

time (hours)



tardiness: 46.49

time (hours)



tardiness: 51.82

time (hours)

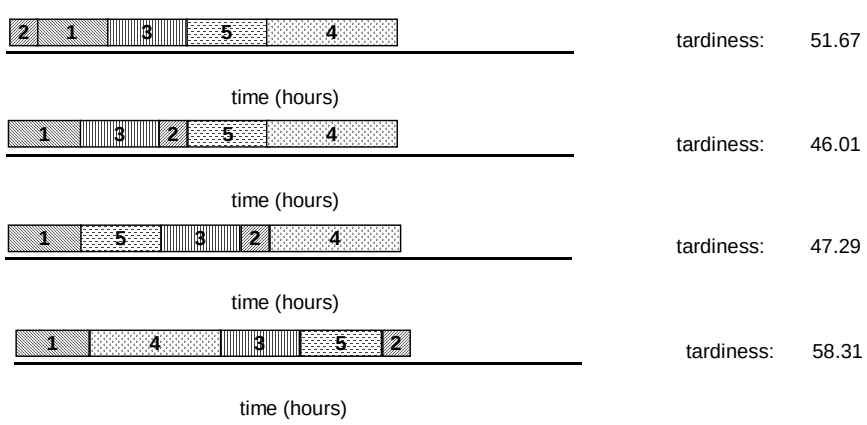
tardiness: 50.17

time (hours)

Select job sequence with minimum tardiness, if it is different from the initial sequence then repeat step 2. Otherwise, move the second job to all possible positions in the sequence and determine it's respective tardiness.

For this example, the sequence {1, 2, 3, 5, 4} has the minimum tardiness. Since it is the same as the initial sequence, the next job in the sequence, 2, is placed in all posible positions to determine the the respective tardiness.

3



For this example, the sequence {1, 3, 2, 5, 4} has the minimum tardiness. Since it is different from the initial sequence, the first customer order of the new job in the sequence is placed in all posible positions to determine the respective tardiness.

This process is continued until all jobs cannot be moved to another postion in the sequence without improving the total tardiness, in which case the job sequence has reached 2-optimal. For the Greedy Heuristic example shown above, the resulting 2-optimal sequence is: {3, 1, 2, 5, 4} and this gives a total processing time of 29.32 hours.

This chapter presented the scheduling algorithms that will be used to schedule customer orders. The next chapter will discuss the simulation model that will be used to evaluate how these scheduling algorithms perform in a real world setting.