Heuristic Based Scheduling for Toothpaste Filling Problem

S Promjun* and P Jarumaneeeroj
Industrial Engineering Department, Chulalongkorn University, Bangkok,10330, Thailand

Corresponding author *: Suwatthiphong.P@student.chula.ac.th

Abstract. This research focuses on the development of a heuristic method for the Toothpaste Filling Problem (TFP) whose complication lies on a vast variety of products (formulae and sizes), together with specific restrictions on filling machines and storage tanks. More specifically, filling machines are non-identical, each of which has different speeds and capacities for filling toothpaste of various sizes. As unloading requires storage tanks, each with much larger capacity when compared to one particular production order, a tank will be then matched with several production orders and be repositioned among filling machines to avoid unnecessary changeovers. We solve the TFP by means of a two-phase method, where the initial solution is constructed by an Earliest Due Date (EDD) dispatching rule. Once completed, the initial solution is then iteratively improved by a series of improvement operators, mimicking the concept of Variable Neighbourhood Search (VNS), until no improvement could be found. We test our heuristic on a set of sample instances acquired from one of the biggest consumer products manufacturers in Thailand. Impressively, when compared to the current practice, our proposed heuristic could help reduce makespan by 32% on average, or equivalently an annual saving of $0.6 million, mainly from the reduction on overtime payment and changeovers.

1. Introduction
Currently, the competition among consumer products businesses is quite intense, in terms of price, quality, and marketing strategy. As such, the companies need to introduce new products much more frequent in order to satisfy the needs of diversified customer segments. While these wide ranges of products may help strengthen the companies' business, they however pose a serious problem on the manufacturing side as more products imply more complicated production planning – and so higher production cost from additional changeovers.

Typically, a changeover is required when we change the production sequence from one to another order. And, the times required by changeovers may differ as they depend on the similarity – such as formulae and container sizes – between two consecutive orders. Production could have become worse with inefficient planning, as it may lead to plans with longer makespans, and so delivery delay. In order to avoid such circumstances, many need to rely on ad hoc schedules of additional work shifts, which are mere short-term solutions to this problem.

To help solve this problem in longer term, many dispatching rules are then devised [1]. But, these rules, unfortunately, work well under certain assumptions – and they might be even inapplicable due to the existence of some practical restrictions like compatibility and storage tank constraints found in the underlying Toothpaste Filling Problem (TFP) investigated in this paper.
Toothpaste is produced from a continuous line of two processes, that is, mixing and packing as shown in Figure 1. All ingredients will be first mixed and loaded into storage tanks of various sizes in mixing department. Once completed, these storage tanks will be, in turn, unloaded by filling machines, where the resulting products will be subsequently packed and stored in the company’s warehouse at the end of production line.

**Figure 1.** Toothpaste manufacturing process.

Based on our case study company’s production profile, toothpaste filling process is quite rich in the sense that there are about 30 different toothpaste formulae of various sizes to be produced each month – accounting for an approximately 80 SKUs in total. Since (i) a changeover may involve a change of toothpaste formulae, or a change of container sizes, or both, depending on the sequence of orders assigned to each machine, and (ii) filling machines are of different speeds, each with different production capacities, deciding which order to be placed in which machine is clearly not an easy task – as the finishing time of an order may vary and inevitably affects the finishing times of the rest. Moreover, as storage tank’s capacity is considerably large when compared to the amount required by one particular order, the tank will be then matched with several orders; but, only be unloaded by one machine at a time. Consequently, the tanks will be typically repositioned among filling machines to avoid unnecessary changeovers. This problem is made more complicated when there are emergency orders and ingredient shortages that prevent some orders to be fulfilled.

Due to the aforesaid complication, an efficiently thorough production plan is hard to be made resulting in high amount of overtime payment and number of overdue jobs. Based on previous statistics, we found that approximately 23 orders were overdue monthly; and, the company had to pay around $0.43 million annually for additional work shifts.

In order to help reduce these figures in longer term, a two-phase heuristic is herein proposed, where we construct an initial solution based on an easy-to-implement rule; and, once completed, such a solution is then iteratively improved by a series of improvement operators, namely Swap (SWAP) and Moving Exchange (MOVEX), until no improvement could be found. The implementation of our proposed heuristic is quite similar to that of the Variable Neighborhood Search (VNS); but, with no solution shaking as we repeatedly execute such a heuristic every time production information has been updated.

### 2. Literature Review

Scheduling is a common process found in most manufacturing; but, its detail may vary depending on characteristics of the underlying production processes. For example, the machines may be either identical or non-identical, each of which may be compatible with all or a subset of products. The jobs may also vary and require different operational setups as production sequences have been altered. In
the literature, these problems are known as sequence-dependent scheduling problems with parallel machines [1][2-3], which are proven to be NP-hard problems [4-5]. Thence, solving these instances of practical sizes to optimality is seemingly implausible.

In order to address this issue, many efficient heuristics are then devised and applied – though the finding of optimal solution is not guaranteed. Reference [6] transformed a variant of sequence-dependent scheduling problems with single machine, called the Float Glass Manufacturing Problem (FGMP), into the Travel Salesman Problem (TSP), and solved the resulting problem by means of the Variable Arc Exchange Heuristic [7]. As an extension to the FGMP, where multiple machines were brought into consideration, [8] investigated and transformed a so-called Tablets Film Coating Problem (TFCP) into the Vehicle Routing Problem with Time Window (VRPTW). They found that small TFCP instances, with up to 25 production orders, could be solved to optimality; but, unfortunately, not for practical TFCP instances comprising of at least 50 production orders. Accordingly, a two-phase heuristic was then devised. In their setting, the initial solutions were first constructed by easy-to-implement dispatching heuristics; and, once completed, such solutions were then iteratively improved by a series of local search operators, mimicking the concept of the Variable Neighborhood Search (VNS), until no improvement could be found.

Similar to [8], [9] solved practical crew transportation problems arisen in most oil and gas exploration companies by means of the VNS; but with different local search operators. More formally, the authors constructed an initial solution based on Solomon’s Push Forward Insertion Heuristic (PFIH), and iteratively improved such a solution by λ-interchange and cyclic transfer algorithms until one of the stopping criteria had been met. The VNS was also employed in the study of [10] who investigated the integrated production-inventory problem, known as the Production Routing Problem.

3. Problem Description

The Toothpaste Filling Problem (TFP) is a variant of sequence-dependent scheduling problems with non-identical parallel machines, as it requires different setup times for assignments of the same production sequence on different filling machines. And, in this case, two types of setups are involved, one for the changes of container sizes and another for those of toothpaste formulae. As changing container size requires more operational steps when compared to those of the formulae, the setup time is thus higher – as reported in Table 1.

| Machine Names | Container Sizes | Setup Times (Min.) | Both Container Sizes and Formulae |
|---------------|-----------------|--------------------|----------------------------------|
| A             | 40              | 15                 | 55                               |
| B             | 60              | 15                 | 75                               |
| C             | 60              | 15                 | 75                               |
| D             | 90              | 60                 | 150                              |
| E             | 45              | 10                 | 55                               |

While planners may be able to reduce setup times by minimizing total number of setups, this strategy, however, may lead to plans with longer makespans – and delivery delay – as machines need to wait for other orders with the same formula and container size. In order to avoid such a circumstance, storage tanks are instead repositioned among these machines. However, as storage tanks could only be unloaded by one machine at a time – but, each may be used for filling toothpaste of
various sizes (orders) – production planning is thus complicated and made more difficult for planners to find the optimal filling sequence on a daily basis. This situation might have become worse when there are emergency orders and other disruptions, including ingredient shortages, preventing orders to be fulfilled.

More importantly, as we may transform the TFP into the VRPTW, where a tour is defined by a sequence of orders assigned to each filling machine, solving practical TFP instances to optimality in a timely manner is less likely.

4. Methodology
Our proposed heuristic is a two-phase one, where the initial solution is constructed based on a simple and easy-to-implement dispatching rule, i.e. EDD; and, once done, it will be further improved by a series of improvement operators, including Swap (SWAP) and Moving Exchange (MOVEX), mimicking the concept of the Variable Neighborhood Search (VNS), until no improvement could be found. While the structure of such a heuristic is quite similar to those of [9] and [6], its implementation is far more complicated due to a large variety of toothpaste formulae and container sizes, together with storage tank and compatibility constraints.

More formally, while storage tanks can only be unloaded by one machine at a time, each can be used for filling toothpaste of various sizes. Additionally, not all products could be filled by all machines due to the existence of compatibility between container sizes and filling machines. Consequently, every time an order is assigned, we need to check whether such an action is possible. If not, the next move in queue will be called; and repeat until all orders are completely assigned.

4.1. Preprocessing Steps
At the beginning of planning process, related input data will be acquired from the database, including Semi-Finished Product Codes (S/FC), Finished Product Codes (F/C), semi-finished product quantities, finished product quantities, delivery schedules, number of available storage tanks, and the amount of packing material. We also need to collect the times required by all changeovers (Table 1), as well as speeds and capacities of all filling machines as reported in Table 2 below.

| Machine Names | Speed (Pcs./Min.) | Container Sizes | Gram |
|---------------|------------------|------------------|
| A             | 80 - - - - - - 80 90 100 - | 140 150 160 180 - |
| B             | 80 15 20 25 40 80 - - | - - - - - - - - |
| C             | 70 15 25 40 80 - - | - - - - - - - - |
| D             | 160 - - - - - | 80 90 - - - - - - |

4.2. Initial Solution Construction

**Step 1** Check the current list of orders whether there are backorders from the previous planning cycles, maybe from ingredient shortages. If ones exist, they will be first scheduled based on the same rules as incoming production orders.

**Step 2** Schedule the incoming orders based on their due dates, i.e. place those with the earliest due dates in the production queue first (EDD). In this step, we need to check whether the amount of semi-finished products is sufficient for all production orders. If they are not enough, those associated orders will be removed from the production queue. Similarly, as completed orders will be subsequently packed downstream, orders with insufficient packing material will be also removed from the production queue until the next planning cycle begins.
Step 3 Check filling machines’ status, and so the ready times of all machines. If the machine is currently idle, the ready time will be set to the current planning cycle’s starting time; else, it will be set to the current order’s finishing time.

Step 4 Allocate orders in the production queue to each machine based on similarity between the current order and the next one in queue. Since changing container sizes requires more setup times, an order with the same container size and formula will be first allocated, followed by an order with the same container size but different formula, an order with different container size but same formula, and an order with different container size and formula, respectively.

Step 5 Consequently allocate orders to machines based on the above rules until the production queue is empty.

Step 6 Calculate makespan, lateness, and number of tardy jobs from such an allocation.

4.3. Improving the Solutions with Improvement Operators

Once an initial solution is constructed, it will be iteratively improved by two improvement operators, i.e. SWAP and MOVEX, until no improvement could be found, as described by the following pseudo code.

---

**Pseudo Code of the Proposed Heuristic**

1: **Input data:** S/PC, F/C, Quantities of Semi-finished products and finished products, Delivery schedules, Number of available storage tanks, Amount of packing material, Production orders.

2: **Construct** an initial solution using EDD and a set of predefined rules

3:  
4: while z = true
5:  
6: Call SWAP
7:  
8: Select a pair of orders from different machines
9:  
10: Check compatibility between orders and filling machines
11:  
12: if compatibility condition is violated, discard such a pair and select the new one (Back to Line 6)
13:  
14: else
15:  
16: Check the availability of storage tanks and other components
17:  
18: if an order is not ready at the time of swapping, discard such a pair and select the new one (Back to Line 6)
19:  
20: else
21:  
22: Check the availability of storage tanks for the whole production sequences
23:  
24: if swapping is possible for the whole production sequences on both machines, swap the positions of such a pair and compute makespan, lateness, and number of tardy jobs.
25:  
26: else
27:  
28: Discard such a pair and select the new one (Back to Line 6)
29:  
30: end if
31:  
32: end if
33:  
34: if no better solution is found within r iterations
35:  
36: Call MOVEX (the implementation of MOVEX is quite similar to that of SWAP)
37:  
38: if no better solution is found within r iterations, z = false
39:  
40: else
41:  
42: Return a better tour as new solution and back to Line 6
43:  
44: end if
45:  
46: end if
47:  
48: end while
49:  
50: return Best-found solution

---

Figure 2. Pseudo code of the proposed heuristic.
4.3.1. **SWAP Sub-routine.** SWAP is an easy, but powerful, improvement operator, where we swap a pair of orders between two different machines and check whether a better solution is found. While this could be easily done in typical VRP-like problems, care should be however taken in this problem as not all pairs of orders could be swapped due to compatibility and storage tank constraints.

- **Step 1** Select a pair of orders from different machines to be swapped.
- **Step 2** Check whether these orders are compatible with the selected filling machines. If not, discard such a pair and proceed to the next one.
- **Step 3** Check the availability of both storage tanks and the amount of semi-finished products. If at least one of them is not available at the times of swapping, discard such a pair and proceed to the next one.
- **Step 4** Check whether these orders are currently unloaded by other machines at the times of swapping. If yes, discard such a pair and proceed to the next one.
- **Step 5** Swap the positions of the selected order pair and calculate the resulting makespan, lateness, and number of tardy jobs.
- **Step 6** If the new solution is better than the original solution, store this new solution as the current one and repeat from Step 1 until no improvement could be found.

4.3.2. **MOVEX Sub-routine.** If SWAP could not find a better solution, Moving Exchange (MOVEX) will be subsequently called. The concept of MOVEX is quite simple as we try to reduce makespan by moving an order from a machine with the longest processing time, i.e. the one that defines makespan, to others with less processing times. As usual, we need to check whether such an order could be moved; and if a better solution is found, SWAP will be called and repeat until MOVEX terminates.

5. **Heuristic Results**

In order to evaluate the performance of our proposed heuristic, we have computed and compared the makespans from such a heuristic with those of current planning and EDD-based planning over a 30-day period as reported in Table 3. All information was acquired from one of the biggest consumer products manufacturers in Thailand; and the experiments were conducted on a computer with Intel® Core™ i5-6200U CPU 2.30 GHz. Ram 8.00 GB. using Python as a programming language. Impressively, our proposed heuristic could reduce production makespan and lateness by 32% and 53% on average; and, it takes only few seconds to perform the whole computation.

6. **Conclusion and Future Works**

This paper focuses on the development of a heuristic for toothpaste filling process so that makespan, lateness, and number of tardy jobs, could be reduced. The devised heuristic is a two-phase one, where the initial solution is first constructed based on EDD and a set of allocation rules; and, once completed, it will be iteratively improved by a series of improvement operators, namely SWAP and MOVEX, until no improvement could be found. Based on our experiments, the proposed heuristic could potentially reduce production makespan by 32% on average, or equivalently a saving of $0.6 million per year — mainly from the reduction of overtime payment and changeovers. Additionally, lateness could be reduced by 53%.

When compared to the current planning, or other simple dispatching rules, our proposed heuristic is much more efficient and it takes just few seconds to complete the whole computation. Accordingly, we can execute this heuristic every time the system has been updated, either by the arrival of emergency order or other disruptions. This heuristic could also be adapted to the planning of other consumer products having the same production process, like shampoo or liquid soap — as these products are typically produced by the same manufacture; but, maybe, at different sites. As such, substantial saving could be realized.
Table 3. A comparison of makespans from different methodologies.

| Sample No. | Current Plan | EDD Makespan (Min.) | Heuristic Makespan (Min.) | Time (Sec.) | % Improvement |
|------------|--------------|---------------------|---------------------------|-------------|---------------|
| 1          | 2,904        | 2,447               | 1,979                     | 6.90        | 32%           |
| 2          | 2,891        | 2,081               | 1,591                     | 1.71        | 45%           |
| 3          | 1,858        | 1,261               | 999                       | 2.80        | 46%           |
| 4          | 3,182        | 3,316               | 2,128                     | 4.85        | 33%           |
| 5          | 2,780        | 2,044               | 1,634                     | 6.52        | 41%           |
| 6          | 1,789        | 2,010               | 1,238                     | 5.19        | 31%           |
| 7          | 1,976        | 1,854               | 1,832                     | 2.73        | 7%            |
| 8          | 1,594        | 2,431               | 1,570                     | 4.81        | 2%            |
| 9          | 1,962        | 1,826               | 1,365                     | 3.26        | 30%           |
| 10         | 1,645        | 1,867               | 1,482                     | 3.40        | 10%           |
| 11         | 1,771        | 2,981               | 1,319                     | 4.21        | 26%           |
| 12         | 2,030        | 2,497               | 1,427                     | 4.90        | 30%           |
| 13         | 1,915        | 2,905               | 1,279                     | 3.60        | 33%           |
| 14         | 2,126        | 1,814               | 1,467                     | 5.78        | 31%           |
| 15         | 2,206        | 2,935               | 2,030                     | 7.26        | 8%            |
| 16         | 2,465        | 3,213               | 1,349                     | 4.95        | 45%           |
| 17         | 2,760        | 1,329               | 1,255                     | 4.30        | 55%           |
| 18         | 1,625        | 1,821               | 1,297                     | 6.54        | 20%           |
| 19         | 3,082        | 1,868               | 1,381                     | 8.79        | 55%           |
| 20         | 2,115        | 1,263               | 1,080                     | 3.92        | 49%           |
| 21         | 2,565        | 2,541               | 1,873                     | 5.20        | 27%           |
| 22         | 1,833        | 1,576               | 1,315                     | 3.10        | 28%           |
| 23         | 1,665        | 2,071               | 1,123                     | 7.35        | 33%           |
| 24         | 1,705        | 1,733               | 865                       | 3.10        | 49%           |
| 25         | 4,337        | 3,466               | 2,371                     | 4.54        | 45%           |
| 26         | 1,680        | 1,502               | 1,261                     | 4.17        | 25%           |
| 27         | 1,772        | 2,357               | 1,504                     | 4.13        | 15%           |
| 28         | 1,730        | 1,227               | 1,050                     | 2.82        | 39%           |
| 29         | 1,165        | 2,036               | 1,042                     | 3.92        | 11%           |
| 30         | 2,470        | 2,059               | 1,319                     | 3.84        | 47%           |

References

[1] Pinedo M 2008 Scheduling: Theory, Algorithms, and Systems. Springer Publishing Company NY
[2] Bilge U, Furkan K, Kurtulan M, Pekgun,P  2004 A Tabu Search Algorithm for Parallel Machine Total Tardiness Problem Computers & Operations Research pp 397–414
[3] Zadeh B S, Moghaddam R T, Moghadam A T, Rastgar I 2017 Solving a bi-objective unrelated parallel batch processing machines scheduling problem A comparison study Computers and Operations Research. Conference on Industrial Engineering and Applications pp 71–90
[4] Karabulut  K and Tasgetiren F 2014 A variable iterated greedy algorithm for the traveling salesman problem with time windows Information Sciences pp 279,383-395
[5] Zhang D, Cai S, Ye F, Si Y W, Nguyen T T 2017 A hybrid algorithm for a vehicle routing problem with realistic constraints Science Direct Information Sciences pp 167–182

[6] Kitisook K and Jarumaneeroj P 2017 A Variable Arc Exchange Heuristic for Float Glass Manufacturing Problems 4th International Conference on Industrial Engineering and Applications

[7] Helsgaun K 2000 An Effective Implementation of The Lin-Kernighan Traveling Salesman Heuristic European Journal of Operational Research pp 106–130

[8] Pontrakul C and Jarumaneeroj P 2018 Heuristic Based Scheduling for Tablets Film Coating Process ICACS '18 Proceedings of the 2018 2nd International Conference on Algorithms, Computing and Systems pp 238–242

[9] Jarumaneeroj P and Kunaporn S 2017 A Variable Neighborhood Search Approach for Crew Transportation Problems 4th International Conference on Industrial Engineering and Applications

[10] Qiu Y, Wang L, Xu X, Fang X, Pardalos P M 2018 A variable neighborhood search Heuristic algorithm for production routing problems Applied Soft Computing pp 311–318