Industrial Engineering
& Management Systems
Vol 15, No 1, March 2016, pp.1-10
ISSN 1598-7248 | EISSN 2234-6473 |
http://dx.doi.org/10.7232/iems.2016.15.1.001

Sequencing the Mixed Model Assembly Line with Multiple Stations to Minimize the Total Utility Work and Idle Time

Yearnmin Kim*, Won-Joon Choi
Department of Industrial Engineering University of Ulsan, Ulsan, Korea

(Received: August 11, 2015 / Accepted: December 10, 2015)

ABSTRACT

This paper presents a fast sequencing algorithm for a mixed model assembly line with multiple workstations which minimize the total utility work and idle time. We compare the proposed algorithms with another heuristic, the Tsai-based heuristic, for a sequencing problem that minimizes the total utility works. Numerical experiments are used to evaluate the performance and effectiveness of the proposed algorithm. The Tsai-based heuristic performs best in terms of utility work, but the fast sequencing algorithm performs well for both utility work and idle time. However, the computational complexity of the fast sequencing algorithm is $O(KN)$ while the Tsai-based algorithm is $O(KNlogN)$. Actual computational time of the fast sequencing heuristic is 2-6 times faster than that of the Tsai-based heuristic.

Keywords: Mixed-Model Assembly Line, Sequencing, Utility Work, Heuristics, Tsai-Based

* Corresponding Author, E-mail: ymkim@mail.ulsan.ac.kr

1. INTRODUCTION

This paper addresses sequencing mixed-model assembly lines with multiple workstations. It focuses on sequencing paced assembly lines with many distinct products. We assume that each product is characterized by the presence or absence of available options. This situation is common in the automobile industry, where each product can be defined by the presence or absence of air conditioning, dual air bags or single air bag, deluxe seats or standard seats, presence or absence of sun roofs, manual or automatic transmissions, etc. (Yano and Rachamadugu, 1991)

In many assembly systems, products are mounted on a conveyor belt and operators move along with the belt while working on a product. An operator can work on a product only when it is at his/her station. Each station is confined by two limits: an upstream limit, where products enter the station, and a downstream limit, where products leave the station. Therefore, it is desirable to evenly distribute products with high work content to reduce the possibility that an operator cannot finish his/her work before a product leaves his/her station.

The main theme throughout this paper is one of determining a sequence of products to spread out a particular model as smoothly as possible. Total utility work is then minimized. Utility work is the work not completed by the operator.

On a mixed-model assembly line, production planning involves a series of decisions:

(1) Line balancing: determines the cycle time, which is the constant time interval separating consecutive products, and assigns work to assembly operators.
(2) Product sequencing: determines the order in which products are fed into the assembly line.

This paper considers the product sequencing problem with an assumption of constant processing times. The cycle time and product mix are predetermined. The problem of sequencing N types of products on an as-

sembly line is formulated as an integer program with the objective of minimizing the total utility work. For a single station with arbitrary processing times, this problem is known to be strongly *NP*-hard. In this paper we provide a heuristic algorithm to solve this problem.

Several researchers have developed algorithms for production cost minimization in mixed-model sequencing. Thomopoulos (1967) and Macaskill (1972) develop approximate algorithms to minimize the sum of utility work, idleness, deficiency, and congestion. Okamura and Yamashina (1979) develop a heuristic algorithm to minimize the risk of stopping the conveyor. Yano and Rachamadugu (1991) offer a mixed integer programming formulation for the problem of minimizing total workload at a single station. Bolat (1994) suggests a lower bound on the total utility work in case of zero upstream travel time. Bolat and Yano (1992a) develop scheduling algorithms that minimize utility work in *0(N)* computation time for a single station on a paced assembly line. To ease computation, Bolat and Yano (1992b) also introduce a surrogate objective for utility work. Tsai (1995) investigates the problem of sequencing two types of products on an assembly line with two objectives: minimizing (1) the risk of conveyor stoppage and (2) the total utility work. He developed an optimal algorithm minimizing both objectives for a single station with two product types, each of which has a constant processing time. Bolat (1997) proposes a branch and trim procedure to minimize total utility work and develops a single-pass deterministic heuristic which retains only the best node at each level of the tree to generate good solutions efficiently. The proposed heuristic differs from the one developed by Bolat (1994) in that they consider operator's upstream walking time. Duplaga and Bragg (1998) address the mixed-model sequencing problem which smoothes out the rate of use of each component part by providing a controlled comparison of six sequencing heuristics. Several researchers such as Zeramdini *et al*. (2000), Korkmazel and Meral (2001), and Kotani and Ohno (2001) propose a bicriteria sequencing for mixed-model assembly lines, where two goals are: (1) keeping a constant rate of usage of parts and (2) smoothing the workload at work stations. McMullen and Frazier (2000) present a simulated annealing based heuristic that simultaneously considers both setups and the stability of parts usage rates. Xiaobo and Ohno (2000) propose a sequencing problem with the goal of minimizing total conveyor stoppage. Time-based goal chasing method which considers the assembly time of each product is applied to multiple work stations by Kurashige *et al*. (2002). Kim *et al*. (2015). solved the problem of unstable production chains by considering allocation rate conformance.

The controlled comparison of a few sequencing heuristics which smoothes the workload at work stations is rare. Bolat (1997) compares a single pass deterministic (SPD) heuristic with the Yano and Rachamadugu (1991) algorithm. The problem of minimizing the total utility work in cases of multiple stations based on the Tsai's (1995) optimal algorithm is not studied further and compared also.

Section 2 formulates a mathematical program to minimize total utility work for a single station with N types of products. Section 3 presents a fast sequence algorithm for mixed model sequencing problems. Section 4 outlines the results and comparisons of a fast sequence algorithm and the other approach to this problem.

2. A MATHEMATICAL PROGRAMMING FORMULATION FOR MINIMIZING TOTAL UTILITY WORK

2.1 Problem Statements

In this section we will describe how to schedule jobs to minimize total utility work in a mixed model assembly line. We assume that all processing times at a station are constant over the scheduling horizon. Consider a station that receives products in a predetermined sequence σ. When the first product in σ arrives, the operator starts at the upstream limit of his/her station. Figure 1 shows the starting positions, operator's movement, idle time, utility time, and maximum displacement for the sequence of four jobs. While working on the j th job, the operator works with the product for the time t_j. The processing time of product i is p_i; hence, $p\sigma(i) = t_j$, provided that the operator completes the work within his/her station. Upon releasing a product, the operator walks upstream at a constant time w to catch the next product.

Conveyor moving time is C (station length/moving speed of conveyor); thus, the operator meets the next product after walking upstream for a time w unless the operator is less than time w from the upstream limit upon releasing the current product. In which case the operator stops at this limit and meets the next product there.

The starting time, finishing time (if a downstream limit exists), and utility work of the jth sequence are;

$$y_j, \quad f_j = \min\left\{y_j + t_j, C\right\}, \text{ and } \left[y_j + t_j - C\right]^+ \quad \text{for} \quad j = 1, 2, \cdots, n$$

2.2 Problem Formulation

In the development of our formulation, we make use of the following notations:

C = cycle time (station length/moving speed of conveyor)

t_j = amount of time allocated to the job on the jth position

y_j = start time of work on the jth position from the origin.

p_i = processing time for job i

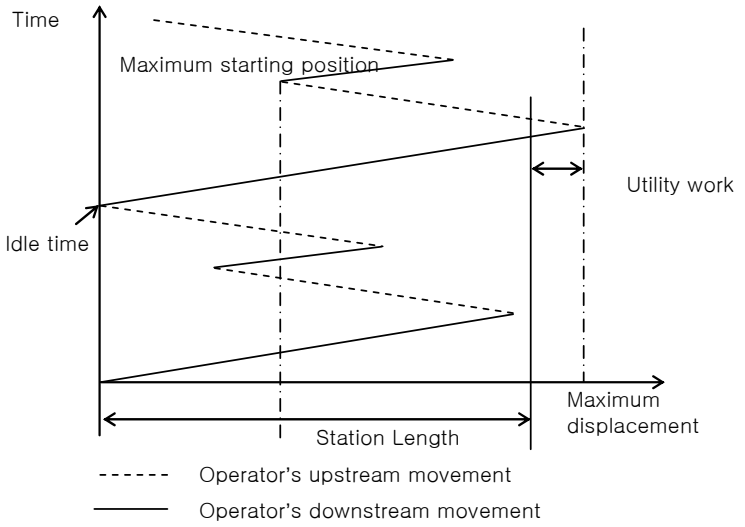

Figure 1. Schematic diagram of a station of mixed model assembly line.

$$[a]^+ = \max\{a, 0\}$$

w = upstream walking time of a worker.

$$x_{ij} = \begin{cases} 1 & \text{if job } i \text{ is assigned to the } j\text{th position.} \\ 0 & \text{otherwise} \end{cases}$$

If we assume that the first job in the sequence starts at time zero, the problem of minimizing the total utility work is formulated as follows. For a single station, the problem becomes:

$$Min\ Z = \sum_{j=1}^{n} \left[y_j + t_j - C \right]^+$$

s.t

$$y_j = \left[y_{j-1} + t_{j-1} - w \right]^+ \quad \forall j \tag{1}$$

$$\sum_i x_{ij} = 1 \quad \forall j \tag{2}$$

$$\sum_j x_{ij} = 1 \quad \forall i \tag{3}$$

$$t_j = \sum_i p_i x_{ij} \quad \forall j \tag{4}$$

$$x_{ij} = 0 \text{ or } 1,\ t_j \geq 0,\ p_j \geq 0,\ y_j \geq 0 \tag{5}$$

The objective function of the problem is minimizing the total utility time. Constraint (1) is the start time of work on the jth position from the origin if it assumes no downstream limit. If a downstream limit is allowed, then the constraint (1) is changed to $y_j = [\min((y_{j-1} + t_{j-1}), C) - w]^+$. Constraints (2), (3), and (4) are used for assigning the job i to the jth position, which has p_i = processing time. Constraints in (5) show binary condition and nonnegative condition.

3. SEQUENCING A MIXED MODEL ASSEMBLY LINE WITH MULTIPLE STATIONS

3.1 Overview of a Fast Sequencing Algorithm

A fast sequencing algorithm adopts the same search scheme as the heuristic branch and bound procedure of Bolat (1997); that is, branching occurs for each position in the sequence, but only the node with the smallest lower bound is retained at each level of the branch and bound tree. If we apply Tsai's algorithm (1995) to multiple stations, we must get the sequence of the remaining products explicitly and use it to calculate a lower bound on the utility work. But the fast sequencing algorithm adopts a lower bound on utility work without getting the sequence of the remaining products. The lower bound of this heuristic may normally be looser than that of Tsai's algorithm, but needs far less computational time for multiple stations.

Now we address how to get a lower bound on the utility work from the remaining products at a station (say station k). Suppose the numbers of the option jobs and the basic jobs in the set of the remaining products at station k are N_k^O and N_k^B respectively and the assembler starts work on the first product of the remaining products at the time s_k. We also define:

$t_{m,k}$: the process time of model m at station k, where k = 1, ⋯, K, m = 1, ⋯, M.
w: the operator upstream walking time
C: the conveyor moving time of station

If there are no restrictions on the boundaries, the assembler would finish the work of the last product upon time $s_k + N_k^O\left(t_k^O - w\right) + N_k^B\left(t_k^B - w\right) + w$. Hence,

$$\left(s_k + N_k^O\left(t_k^O - w\right) + N_k^B\left(t_k^B - w\right) + w - C\right)^+ \quad (1)$$

is a lower bound on the utility work (LBU_k) for the remaining products. For N jobs with K stations, above formulations can be described as Eq. (2)

$$\sum_{k=1}^{K}\left(s_k + \sum_{m=1}^{M} N_k^m\left(t_{mk} - w\right) + w - C\right)^+ \quad (2)$$

Summing up the lower bounds on the total utility work for stations $k = 1, \cdots, K$, we get a lower bound on the total utility work on the whole line. If there are multiple sequences with the smallest lower bound, we select one using a leveling scheme. For a single station, we select one according to the utility time and idle time. Bolat's (1997) lower bound of total utility work is another representation of lower bound where 'distance' is considered instead of our 'time' approach in contrast to Eq. (2).

3.2 An Example of Fast Sequencing Algorithms

In this example, at each iteration, it evaluates two jobs (basic and option job) on a deterministic basis. A fast sequencing heuristic can be viewed as a heuristic branch-and bound procedure in which only the best node is retained at each level of the tree. The best job is selected for each station in the sequence. The criteria are the utility work for the job under consideration and a lower bound on the utility work for the remainder of the sequence. If the utility time and the lower bound of each job are same, then we use idle time for tie-breaking.

We consider the station described for Figure 1, and two types of job with the 14 sec and 7 sec. Remaining products on station k is 10 and 12 each. Conveyor moving time is 15 sec and upstream walking time of a worker is 10 sec. If the start time of work at the station is assumed as 0, then we can get a lower bound of the problem using Eq. (1) in 3.1. For tie breaking, we calculate idle time using Eq. (3) if the operator waits for the job.

$$\left(w - f_k\right)^+ \quad (3)$$

A Single Station Example: $t_k^O = 14$, $t_k^B = 7$, $N_k^O = 10$, $N_k^B = 12$, $C = 15$, $w = 10$,

• Iteration 1:
Step 1: If we schedule job type O first, then $s_{k1} = [\min(0 + 14, 15) - 10]^+ = 4$ and $LBU_{k1} = [4+9\times(14-10)+12\times(7-10)+10-15]^+ = 0$, where $N_k^O = 9$ and $N_k^B = 12$.

If we schedule job type B first, then $s_{k1} = [\min(0 + 7, 15) - 10]^+ = 0$ and $LBU_{k1} = [0+10\times(14-10)+11\times(7-10)+10-15]^+ = 0$, where $N_k^O = 10$ and $N_k^B = 11$

Step 2: The utility time of job type O and B are same. And lower bounds of each job type are same, we must calculate the idle time of each job type for tie breaking.
Step 3: The idle time of job type O is $[10-14]^+ = 0$. The idle time of job type B is $[10-7]^+ = 3$. The idle time of job type O (= 0) is smaller than that of job type B (= 3), so we choose job type O.

• Iteration 2:
Step 1: If we schedule job type O, the sequence is OO, and $s_{k2} = [\min(4+14, 15)-10]^+ = 5$ and $LBU_{k2} = [5+8\times(14-10)+12\times(7-10)+10-15]^+ = 0$, where $N_k^O = 8$ and $N_k^B = 12$.

If we schedule job type B, the sequence is OB, and $s_{k2} = [\min(4+7, 15)-10]^+ = 1$ and $LBU_{k2} = [1+9\times(14-10)+11\times(7-10)+10-15]^+ = 0$, where $N_k^O = 9$ and $N_k^B = 11$.

Step 2: Because the utility time of the sequence OO is $[4+14-15]^+ = 3$, and the utility time of the sequence OB is $[4+7-15]^+ = 0$, and lower bounds of each job type are the same, we select job type B and resulting sequence is OB.

For brevity we will skip the rest of this iteration.

We get a near optimal sequence of this problem if we apply the above algorithm. Therefore, performance of the algorithm will be presented in section 4. The computational time is *O(KN)*, where K is the number of stations and N is the number of job type.

3.3 Two-Stage Fast Sequencing Algorithm

This heuristic is identical to the fast sequencing heuristic, except that it examines all combinations of the products over the current stage and the next stage. It locates the combination of the products yielding the smallest lower bound on utility work and selects that product at the current stage as the next product in the sequence. The process of calculating the lower bound is the same as in the fast sequencing heuristic.

3.4 Tsai-based Heuristic

For evaluating the performance and effectiveness of the proposed algorithm, we want to use Tsai-based heuristic for comparison. This heuristic is an extension of Tsai's heuristic for a single station to multiple stations. The Tsai-based heuristic adopts the search scheme of a heuristic branch and bound procedure in which branching occurs for each position in the sequence, but only the node with the smallest estimated lower bound is retained at each level of the branch and bound tree.

Suppose that at iteration p (p = 1, ⋯, D), a sequence ($i_1, i_2, \cdots, i_{p-1}$) is given. Let $\{M_1, M_2, \cdots, M_I\}$ be the models remaining to be sequenced. Then the next model i_p is selected among $M_1, M_2, \cdots, M_I$, which is likely to yield the smallest increase in utility work. We check every possible sequence,

$(i_1, i_2, \cdots, i_{p-1}, M_1)$,
$(i_1, i_2, \cdots, i_{p-1}, M_2)$, $\cdots$, $(i_1, i_2, \cdots, i_{p-1}, M_l)$.

For each sequence, we get an "estimated" lower bound on the total utility work of an optimal completion in the following way:

First, we calculate an "estimated" lower bound on the total utility work of an optimal completion from the sequence at the station, for each station. Here, stations are handled independently. Second, we add the "estimated" lower bound on the utility work at each station to get the "estimated" lower bound on the assembly line's total utility work of an optimal completion from the sequence.

Now we address how to get an "estimated" lower bound on total utility work of an optimal completion from $(i_1, i_2, \cdots, i_{p-1}, M_p)$ at a station (say station k). Suppose s_k denotes the worker's starting position for the next product at the current station after all the products in a sequence $(i_1, i_2, \cdots, i_{p-1}, M_p)$ are processed. We apply the following rule of thumb. We dispatch units of basic operations, since processing a basic operation will move the starting position backward and if the worker does not cross the upstream boundary of the station, will not cause any utility work. Next, we apply the SEQUENCE algorithm (Tsai's algorithm for getting an optimal sequence to the single station problem) to the remaining products. (The number of the remaining products is D-p.) Then presuming that the concatenation of the sequence $(i_1, i_2, \cdots, i_{p-1}, M_p)$, the subsequence of the dispatched units, and the subsequence created by SEQUENCE is an optimal completion from the sequence $(i_1, i_2, \cdots, i_{p-1}, M_p)$, we calculate the utility work yielded by the resulting concatenation. This will be counted as an "estimated" lower bound on the total utility work of an optimal completion from $(i_1, i_2, \cdots, i_{p-1}, M_p)$ at station k.

Then, summing all the "estimated" lower bounds on the total utility work for stations $k = 1, \cdots, K$, we get the "estimated" lower bound on the total utility work on the whole line for an optimal completion from the sequence $(i_1, i_2, \cdots, i_{p-1}, M_p)$. We note that the concatenation mentioned above may not yield an optimal completion. Thus, the real value of the total utility work of an optimal completion may be lower than the obtained "estimated" lower bound, since the "estimated" value can be biased.

We identify the sequence $(i_1, i_2, \cdots, i_{p-1}, M_{p^*})$ having the smallest "estimated" lower bound on total utility work.

4. EXPERIMENTAL RESULTS

We use total utility work, total idle time, and the computing time as performance measures to compare the algorithms. Also for estimating the quality of a heuristic with respect to the measure of total utility work, we get a lower bound on the total utility work as follows: First, we solve the sequencing problem for minimizing the total utility work at each station independent of each other. Here we apply Tsai's (1995) algorithm, which gives an optimal solution for a single station problem and get the minimum utility work for each station. Next, we add up the minimum utility work of each station and the sum is the lower bound on the total utility work for our problem.

We introduce two parameters, option difficulty and option intensity, to capture the profile of the input data:

Option difficulty of station *k*, $\psi_k = \left(t_k^O - c\right)/\left(c - t_k^B\right)$: Option difficulty measures the relative difficulty of the option job over the basic job at a station where c is the cycle time. We confine the values of option job time within above 30% of cycle time for reflecting the real production situation and get option job time randomly in each station. For example, when we consider 3 stations where option difficulty is set to 1, basic job time and option job time for 3 stations can be 0.73, 1.27 in station 1, 0.85, 1.15 in station 2, 0.92, 1.08 in station 3.

Option intensity of station *k*, ρ_k : Option intensity of station *k* measures the relative ratio of the number of the option jobs to the basic jobs at station *k*. ρ_k is defined to be 1 if the weighted average of the processing time of the products is equal to the cycle time.

We generated several cases to test the performance of each algorithm. First, we considered the cases of 2, 3, 4, and 5 stations with production quantities of 50 to 500 with an increment of 50 (ψ_k of each station was fixed to 1 and ρ_k was distributed over 0.33 and 3).

Table 1 shows the performance of each algorithm when ρ_k is set to 0.33. In this table, we can see that almost all of the utility times for each algorithm approach the lower bound of the total utility work. The reason for the small utility time of each algorithm is that each station is lightly loaded due to the small value of ρ_k. As the total number of stations increases, the two-stage fast sequencing heuristic and the fast sequencing heuristic yield smaller idle times than the other heuristics. In Table 1, "SEQUENCE' means Tsai based algorithm. "Fast" means fast sequencing heuristic and "2 Stage" means two-stage fast sequencing heuristic.

Table 2 show the results for the case where ρ_k is set to 3. In this case, each station is heavily loaded in comparison to Table 1. The Tsai-based heuristic always provides the smallest utility times. We note that the utility times of the Tsai-based heuristic approach the lower bounds of the total utility works. As the total number of the stations increases, the two-stage fast sequencing heuristic and the fast sequencing heuristic produce small idle times and small utility times as well. In Table 1 and 2, Fast and Two-stage algorithm show better results in case of idle time in comparison to the SEQUENCE algorithm. That's why SEQUENCE algorithm minimizes total utility work only.

For measuring the computing time, we used an IBM compatible Samsung PC with Intel Pentium® 3805U (1.90 GHz) that has 4GB RAM, 64-bit file system. The operating system was Windows 7. Table 3

Table 1. Performance of algorithms in case of ρ_k equal to 0.33

Station = 2, Job Type = 4

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SEQUENCE	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	1.25	2.30	3.10	3.73	3.55	4.28	4.36	4.41	5.07	5.51
Fast	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.32	1.04	1.32	2.04	1.91	2.63	2.96	3.63	3.95	4.67
2 Stage	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.32	1.04	1.36	2.09	1.96	2.68	3.00	3.73	4.05	4.77

Station = 3, Job Type = 8

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SEQUENCE	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	1.84	2.90	3.20	2.64	4.15	4.45	4.80	6.32	5.36	7.63
Fast	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.37	1.29	1.47	1.89	2.16	3.06	3.48	4.18	4.29	5.50
2 Stage	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.37	1.32	1.54	1.96	2.23	3.13	3.55	4.30	4.41	5.62

Station = 4, Job Type = 16

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SEQUENCE	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	1.77	1.39	4.44	4.79	4.55	6.63	8.13	9.15	10.06	9.34
Fast	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	1.79	0.57	2.23	1.54	1.81	3.24	4.05	4.09	5.08	4.04
2 Stage	Utility	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	1.79	0.55	2.27	1.51	1.72	3.11	4.25	4.02	5.28	4.15

Station = 5, Job Type = 32

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	0.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
SEQUENCE	Utility	0.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13
	Idle	0.00	1.81	2.61	5.68	4.11	6.42	5.47	10.11	9.34	12.68
Fast	Utility	0.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.00	1.56	0.00	4.58	3.26	5.59	2.81	6.49	5.70	9.58
2 Stage	Utility	0.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.00	1.51	0.00	4.48	3.02	5.66	2.86	6.41	5.47	10.05

shows the computation time (CPU seconds) for the case where ρ_k = 0.33 and ψ_k = 1. Table 4 shows the computation times for the case where ρ_k = 3.0 and ψ_k = 1. In Table 3 and in Table 4, we omit the computational time of Two-Stage algorithm, because Two-Stage algorithm always takes a few more times than Fast algorithm because of its two stage calculation. In Table 4, we see each computation time is longer than that of Table 3 due to option intensity effects. The Tsai based heuristic is more vulnerable to option intensity than the fast sequencing heuristic as shown in Table 4. The fast sequencing heuristic always has shorter computation time

Table 2. Performance of algorithms in case of ρ_k equal to 3.00

Station = 2, Job Type = 4

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.00
SEQUENCE	Utility	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.27	0.80	0.27	0.29	1.62	0.80	1.62	0.80	0.27	0.80
Fast	Utility	0.00	0.00	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.14	0.14	0.14	0.00	0.82	0.14	0.82	0.14	0.14	0.14
2 Stage	Utility	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00
	Idle	0.00	0.00	0.00	0.00	0.82	0.00	0.82	0.00	0.00	0.00

Station = 3, Job Type = 8

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00	0.06
SEQUENCE	Utility	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.07
	Idle	0.48	1.02	0.48	0.29	0.48	1.30	0.48	1.30	0.48	0.29
Fast	Utility	0.00	0.00	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.60
	Idle	0.29	0.29	0.29	0.21	0.29	0.29	0.29	0.29	0.29	0.14
2 Stage	Utility	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.33
	Idle	0.08	0.08	0.08	0.00	0.08	0.08	0.08	0.08	0.08	0.00

Station = 4, Job Type = 16

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	0.00	0.00	0.00	0.06	0.00	0.06	0.00	0.00	0.80	0.00
SEQUENCE	Utility	0.00	0.00	0.00	0.26	0.00	0.06	0.00	0.00	1.14	0.00
	Idle	0.82	4.24	0.90	0.20	0.61	5.47	0.61	1.48	0.15	4.37
Fast	Utility	0.00	0.00	0.00	0.60	0.00	0.60	0.00	0.00	2.20	0.00
	Idle	0.99	3.59	0.79	0.77	0.79	5.20	0.79	0.79	0.95	3.34
2 Stage	Utility	0.00	0.00	0.00	0.33	0.00	0.33	0.00	0.00	1.26	0.00
	Idle	0.69	3.44	0.08	0.14	0.08	4.88	0.08	0.29	0.14	3.34

Station = 5, Job Type = 32

Name	Type	50	100	150	200	250	300	350	400	450	500
Lower Bound on Total Utility Work	Utility	7.73	0.00	0.00	0.06	0.00	0.06	0.00	1.13	0.00	0.06
SEQUENCE	Utility	7.73	0.00	0.00	0.07	0.00	0.06	0.39	1.13	0.00	0.57
	Idle	0.00	6.87	1.33	1.44	6.02	7.20	1.50	14.07	10.00	1.39
Fast	Utility	8.64	0.00	0.00	0.60	0.00	0.60	0.00	1.59	0.00	0.60
	Idle	0.00	6.51	1.05	2.94	6.04	7.44	1.65	13.87	8.65	1.29
2 Stage	Utility	8.19	0.00	0.00	0.33	0.00	0.33	0.00	1.39	0.00	0.33
	Idle	0.06	6.29	0.49	0.89	6.04	7.49	0.49	14.06	8.20	0.36

than that of Tsai based heuristic.

Table 5 shows the computation times (CPU seconds) for the case where ρ_k = 1 and ψ_k = 0.33. Table 6 shows the computation times for the case where ρ_k = 1 and ψ_k = 3. In Table 6, we see that each computation time is no longer greater than that of Table 5. Option difficulty effects did not appear in Table 6. However the fast sequencing heuristic always has shorter computation times than that of the Tsai based heuristic.

In summary, we can say that the Tsai-based heuristic performs well in terms of the utility work and the fast sequencing heuristic performs well in terms of utility

work and idle time. However the computation time of the fast sequencing heuristic is much better that that of the Tsai based heuristic. The computational complexity of the fast sequencing heuristic is *O(KN)* because we use only the smallest lower bound where *K* means the number of station and *N* means the number of jobs. However, The computational complexity of Tsai-based heuristic is *O(KNlog(N))*, where *log(N)* times more step is required for sequencing the jobs (Tsai, 1995). Actual computational time of the fast sequencing heuristic is 2-6 times faster than that of the Tsai-based heuristic.

Hence, we suggest the Tsai-based heuristic as "the" algorithm for getting a sequence with minimal utility work; however, in the case of a fast algorithm to solve problems concerned with idle time, the fast sequencing heuristic is a possible alternative.

Table 3. Calculation time of each algorithm in case of ρ_k = 0.33 and ψ_k = 1

Station = 2, Job Type = 4

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	0.18	2.77	5.17	11.42	16.80	21.78	27.11	36.30	44.26	14.96
Fast	0.04	0.91	3.28	3.36	3.53	5.05	5.27	5.72	7.68	2.19

Station = 3, Job Type = 8

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	0.51	2.32	11.21	15.10	20.98	31.88	37.98	51.34	59.53	78.14
Fast	0.46	1.18	2.73	2.55	3.44	5.87	5.47	8.73	7.11	9.67

Station = 4, Job Type = 16

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	2.24	8.19	15.61	26.66	34.90	39.92	51.27	69.76	77.35	125.41
Fast	0.98	3.04	3.71	4.99	6.37	6.57	7.61	10.19	9.97	12.43

Station = 5, Job Type = 32

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	5.14	4.28	11.98	6.46	9.74	12.92	42.85	89.07	45.67	38.96
Fast	3.06	0.14	1.88	0.50	0.44	0.48	8.33	9.76	8.65	8.60

Table 4. Calculation time of each algorithm in case of ρ_k = 3 and ψ_k = 1

Station = 2, Job Type = 4

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	2.12	0.64	1.61	2.65	4.14	6.11	8.13	10.65	12.37	15.29
Fast	0.60	0.11	0.14	0.19	0.21	0.28	0.29	1.16	0.53	0.41

Station = 3, Job Type = 8

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	3.17	1.05	2.26	3.42	5.34	8.34	10.30	14.89	17.77	23.59
Fast	0.07	0.13	0.21	0.28	0.82	0.50	0.47	0.55	0.64	2.44

Station = 4, Job Type = 16

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	4.73	3.50	8.27	5.00	34.91	52.12	66.93	83.39	102.21	122.19
Fast	0.12	2.40	1.99	0.42	8.81	11.78	13.67	18.75	17.77	19.83

Station = 5, Job Type = 32

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	4.53	13.30	23.71	38.72	51.54	75.46	87.09	95.91	130.00	159.81
Fast	2.04	5.67	9.04	10.51	13.47	16.03	18.79	20.64	23.58	26.50

Table 5. Calculation time of each algorithm in case of $\rho_k = 1$ and $\psi_k = 0.33$

Station = 2, Job Type = 4

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	1.95	4.28	1.33	7.81	15.97	26.17	37.02	10.02	13.26	59.22
Fast	0.87	1.16	0.11	0.16	0.96	6.47	6.22	0.34	0.40	10.43

Station = 3, Job Type = 8

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	0.48	5.96	2.42	4.15	6.17	7.85	11.80	14.42	19.57	23.70
Fast	0.98	0.11	0.23	0.31	0.29	0.37	0.43	0.46	0.63	0.63

Station = 4, Job Type = 16

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	4.43	1.42	3.01	11.20	8.36	11.03	15.62	20.28	29.53	30.80
Fast	0.08	0.17	0.26	3.22	0.43	0.54	0.64	0.72	7.01	0.84

Station = 5, Job Type = 32

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	4.10	11.01	23.80	37.04	46.27	74.91	23.51	28.57	88.59	98.31
Fast	0.12	0.31	7.16	10.47	11.06	15.50	1.54	6.52	23.91	22.53

Table 6. Calculation time of each algorithm in case of $\rho_k = 1$ and $\psi_k = 3.00$

Station = 2, Job Type = 4

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	0.21	2.74	7.57	13.45	19.64	22.31	31.78	40.41	49.33	56.65
Fast	0.06	1.14	1.77	3.55	4.25	4.30	6.37	7.50	7.93	4.80

Station = 3, Job Type = 8

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	0.62	6.53	12.10	18.59	26.98	36.79	52.71	13.19	17.56	23.50
Fast	4.04	2.36	3.76	5.87	7.55	9.45	8.96	0.49	0.56	0.61

Station = 4, Job Type = 16

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	4.29	1.19	2.81	4.89	7.99	10.48	15.00	18.92	25.09	29.89
Fast	0.07	0.16	0.26	0.35	0.47	0.52	0.68	0.68	0.85	0.98

Station = 5, Job Type = 32

Name	50	100	150	200	250	300	350	400	450	500
SEQUENCE	3.95	1.39	3.56	6.40	9.76	14.44	18.27	24.07	32.21	38.55
Fast	0.12	0.18	0.37	0.53	0.70	0.78	0.89	1.01	1.32	1.33

5. CONCLUSION

In this paper we addressed sequencing mixed-model paced assembly lines with multiple workstations under an assumption that each product is characterized by the presence or absence of a set of available options. We compared three heuristics for finding a sequence with a small amount of utility work. Among the proposed heuristics, the Tsai-based heuristic performed well in terms of the utility work and the fast sequencing heuristic performed well in terms of utility work and idle time. In this paper, we addressed the situation where each station is affected by-at most-a single option. Designing an algorithm, which can handle the case of more than a sin-

gle option, remains a topic for further research. Finally, a comparison with algorithms such as the Bolat (1997) and Yano and Rachmadugu (1991) algorithm over the same data is also required.

REFERENCES

Bolat, A. (1997), Efficient Methods for Sequencing Minimum Job Sets on Mixed Model Assembly Lines, *Naval Research Logistics*, **44**, 419-437.

Bolat, A. (1994), Sequencing jobs on an automobile assembly line: objectives and procedures, *International Journal of Prod. Res.*, **32**(5), 1219-1236.

Bolat, A. and Yano, C. (1992a), Scheduling Algorithms to minimize Utility Work at a Single Station on a Paced Assembly, *Production Planning and Control*, **3**(4), 393-405.

Bolat, A. and Yano, C. (1992b), A surrogate Objective for Utility Work in Paced Assembly Lines, *Production Planning and Control*, **3**(4), 406-412.

Duplaga, E. A. and Bragg, D. J. (1998), Mixed-model assembly line sequencing heuristics for smoothing component parts usage: a comparative analysis, *International Journal of Production Research*, **36**(8), 2209-2224.

Kim, T., Ji, B., and Cho, H. (2015), Heuristics-Based Algrithm for Production Planning Considering Allocation Rate Conformance to Prevent Unstable Production Chain, *Industrial Engineering and Management Systems*, **14**(4), 413-419.

Kim, Y., Choi, W., and Park, C. (2001), A Quick Sequencing Algorithm for a Mixed Model Assembly Line with Multiple Stations, *KORMS Spring Proceedings*, 300-303.

Korkmazel, T. and Meral, S. (2001), Bicriteria sequencing methods for the mixed-model assembly line in just-in-time production systems, *European Journal of Operational Research*, **131**, 188-207.

Kotani, S., Ito, T. and Ohno, K. (2004), Sequencing problem for a mixed-model assembly line in the Toyota production system, *International Journal of Production Research*, **42**(23), 4955-4974.

Kurashige, K., Yanagawa, Y., Miyazaki, S., and Kameyana, Y. (2002), Time-based goal chasing method for mixed-model assembly line problem with multiple work stations, *Production Planning and Control*, **13**(8), 735-745.

Macaskill, J. L. C. (1972), Production-Line Balances for Mixed-Model Lines, *Management Science*, **19**(4), 423-434.

McMullen, P. R. and Frazier, G. V. (2000), A Simulated annealing approach to mixed-model sequencing with multiple objectives on a just-in-time line, *IIE Transactions*, **32**, 679-686.

Okamura, K. and Yamashina, H. (1979), A Heuristic Algorithm for the Assembly Line Model-Mix Sequencing Problem to Minimize the Risk of Stopping the Conveyor, *International J. of Production Research*, **17**(3), 233-247.

Thomopolos, N. T. (1967), Line Balancing-Sequencing for Mixed-Model Assembly, *Management Science*, **14**(2), 59-75.

Tsai, L. (1995), Mixed-model Sequencing to Minimize Utility Work and the Risk of Conveyor Stoppage, *Management Science*, **41**(3), 485-495.

Xiaobo, Z. and Ohno, K. (2000), Properties of a sequencing problem for a mixed model assembly line with conveyor stoppage, *European Journal of Operational Research*, **124**, 560-570.

Yano, C. A. and Racchamadugu, R. (1991), Sequencing to Minimize Work Overload in Assembly Lines with Product Options, *Management Science*, **37**(5), 572-586.

Zeramdini, W., Aigbedo, H. and Moden, Y. (2000), Bicriteria sequencing for just-in-time mixed-model assembly lines, I*nternational Journal of Production Research*, **38**(15), 3451-3470.