ISBN: 978-981-18-5852-9
WCSE 2022 Spring Event: 2022 9th International Conference on Industrial Engineering and Applications
doi: 10.18178/wcse.2022.04.096

Level Rescheduling for Automotive Mixed-Model Assembly Lines with Selectivity Banks

Yingmeng Ji, Hui Sun [+]

Department of Industrial Engineering, Southeast University, Nanjing, China

Abstract. This paper studies a level rescheduling (LRS) problem, in which a selectivity bank is utilized to reshuffle an incoming car sequence prior to final assembly. The objective is to achieve evenly distributed material requirements over the planning horizon. A mathematical programming model is presented to describe the problem. Aimed at rapidly finding good solutions to this NP-hard problem, several heuristic solution procedures, which can be applied in the car storage and release processes respectively, are developed. Computational experiments show that both rule-based and algorithm-based procedures can solve the LRS problem effectively and efficiently. Furthermore, algorithm-based procedures produce noticeably better outcomes.

Keywords: mixed-model assembly line, level scheduling, resequencing, selectivity bank, genetic algorithm, beam search.

1. Introduction

Nowadays, mixed-model assembly lines (MMALs) are widely used in automotive manufacturing. On an automotive MMAL, different models of a common base product (car) are manufactured in intermixed sequences. Thus, it raises an imperative challenge in determining the sequence of car models launched down MMALs. In general, there are two different basic sequencing objectives, i.e., minimizing work overload and leveling part usage [1]. Both objectives stem from the production requirements of workstations in final assembly shops, where substantial manual operations are performed.

However, the above objectives do not consider the sequencing requirements of other departments such as body and paint shops. Body shops may need to follow a repetitive pattern of models, while paint shops normally desire car sequences with less frequent color changes [2]. Thus, in order to fully address the requirement of each department, a practical solution known as *resequencing* is to alter the incoming production sequence based on the requirement of a downstream department.

Two general forms of resequencing in the context of MMALs, i.e., virtual resequencing and physical resequencing, can be utilized [3]. When a virtual resequencing for automotive MMALs is applied, the initial car sequence remains unchanged and only the assignment of cars to customer orders is altered [4-6]. While in physical resequencing, some kind of resequencing buffer is operated to physically change the car sequence. Several buffer types such as automated storage and retrieval system (AS/RS), pull-off tables, and selectivity bank are reported in the literature [2-3]. Among them, selectivity bank is the most widely used buffer system due to low investment cost and relatively good resequencing performance.

Selectivity banks can be used for color batching in paint shops [7-10]. Besides, they can be employed before final assembly for reactive sequence restoration [11-12] and proactive sequence alternation [13-15]. So far, very few studies have discussed the intentional resequencing with selectivity banks based on the objective of leveling part usage, which is in line with the famous just-in-time (JIT) philosophy. References [13] and [14] presented heuristic rules for accomplishing this objective in the dynamic environment, whereas how to smooth part requirements in a static environment has not been investigated. Therefore, in-depth research on resequencing for leveling part usage is of particular importance, since facilitating JIT-part supply is a big concern for most of automotive manufacturers.

[+] Corresponding author. Tel.: +86 25 52090501; fax: +86 25 52090504.
E-mail address: hui.sun@seu.edu.cn.

Dedicated to leveling part usage, level scheduling (LS) is one of three major sequencing approaches [1] [16] [17]. This paper studies a resequencing problem with selectivity banks prior to final assembly in the context of automobile MMALs. The objective of resequencing is the same as LS, i.e., to obtain evenly distributed part requirements induced by the production sequence over the planning horizon.

The remainder of this paper is organized as follows. Section 2 presents a detailed description and a mathematical programming formulation for the level rescheduling (LRS) problem under study. Section 3 proposes several heuristic rules and algorithms to solve the problem. In Section 4, computational experiments are carried out to test the performance of the proposed solution procedures. Finally, concluding remarks are given in Section 5.

2. Problem Description

Figure 1 illustrates a selectivity bank that consists of 4 parallel lanes each having 5 spaces. While arriving at the point *S* in front of the bank, each car in the upstream sequence selects a lane to enter. Meanwhile, on the other end next to the downstream department cars are released successively from the bank. Note that cars in each lane must be released following the first-in-first-out (FIFO) rule, and cars in different lanes can be released alternately. The initial car sequence can thus be changed.

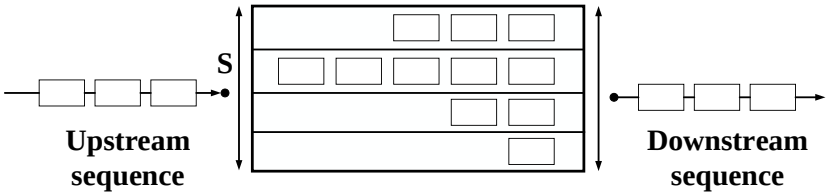


Fig. 1: A 4×5 selectivity bank with 4 lanes each having 5 spaces.

Consider a sequence of *T* cars from the upstream paint shop, each of which belongs to a model *m* ($m \in M$) demanding a specific set of parts to be installed in the final assembly shop. A selectivity bank with *L* lanes each having *W* spaces ($L \times W \geqslant T$) is employed to permute the incoming car sequence with the objective of leveling the part usage within the output sequence. Furthermore, the selectivity bank is assumed to be initially empty, which implies that all the cars can be stored in the bank simultaneously. Referring to [15], with the notations listed in Table 1 the part-oriented LRS problem based on selectivity banks can be formulated as follows.

Minimize

$$D = \sum_{t=1}^{T} \sum_{p \in P} \left(z_{pt} - r_p \cdot t \right)^2 \tag{1}$$

Subject to

$$\sum_{i=1}^{T} x_{it} = 1 \quad \forall t = 1, ..., T \tag{2}$$

$$\sum_{t=1}^{T} x_{it} = 1 \quad \forall i = 1, ..., T \tag{3}$$

$$\sum_{l=1}^{L} y_{il} = 1 \quad \forall i = 1, ..., T \tag{4}$$

$$\sum_{i=1}^{T} y_{il} \leq W \quad \forall l = 1, ..., L \tag{5}$$

$$\sum_{t=1}^{T} x_{it} \cdot t - \sum_{t=1}^{T} x_{jt} \cdot t \leq BI \cdot \left(2 - y_{il} - y_{jl} \right) \quad \forall i = 1, ..., T-1;\ j = i+1, ..., T;\ l = 1, ..., L \tag{6}$$

$$z_{pt} = \sum_{t'=1}^{t} \sum_{i=1}^{T} \sum_{m \in M} \left(x_{it'} \cdot c_{im} \cdot b_{mp} \right) \quad \forall t = 1, ..., T;\ p = 1, ..., P \tag{7}$$

$$x_{it}, y_{il} \in \{0,1\} \quad \forall i, t = 1, ..., T;\ l = 1, ..., L \tag{8}$$

Objective (1) minimizes the sum over all deviations of actual from target cumulative demands per production cycle t and part p resulting from the output sequence. Constraints (2) ensure that each position (i.e., cycle) of the downstream sequence receives exactly one car from the upstream sequence, while constraints (3) guarantee that each car is assigned to exactly one position within the downstream sequence. Constraints (4) ensure that each car in the upstream sequence is dispatched to exactly one lane of the selectivity bank. Constraints (5) ensure that each lane's capacity W is not exceeded. Constraints (6) ensure that for any two cars i and j with $i < j$ in the upstream sequence, car i must precede j in the downstream sequence if they are dispatched to the same lane. Thus, the FIFO rule will not be violated. Constraints (7) determine the actual cumulative part demands. Constraints (8) define two binary decision variables x_{it} and y_{il}.

When the number of lanes L is greater than or equal to T, there is no restriction in assigning cars to positions within the output sequence. Thus, the LRS problem turns into a classical level scheduling problem, which was shown to be NP-hard [18]. In order to deal with its computational intractability, we present several heuristic solution procedures.

Table 1: Notation

Notation	Description
T	Number of production cycles (and cars) within the initial sequence, indices t (i respectively) = 1, …, T
P	Set of parts, index p
M	Set of models, index m
L	Number of lanes in the selectivity bank, index l = 1, …, L
W	Number of spaces in each lane, index w = 1, …, W
BI	A very big integer
b_{mp}	Number of part p demanded by a car of model m
c_{im}	Binary parameter: 1, if the model of car i in the upstream sequence is m; 0 otherwise
d_m	Demand for cars of model m
r_p	Target consumption rate of part p, $r_p = \sum_{m \in M} \left(d_m \cdot b_{mp} \right) / T$
x_{it}	Binary decision variable: 1, if car i is assigned to position t of the downstream sequence; 0 otherwise
y_{il}	Binary decision variable: 1, if car i is stored in lane l of the selectivity bank; 0 otherwise
z_{pt}	Number of part p demanded by the cars assigned to the first t production cycles of the downstream sequence

3. Solution Methods

During the resequencing process with selectivity banks, two types of decisions, i.e., lane selection in the storage stage and car selection in the release stage, need to be made. Corresponding solution procedures for the two stages can be jointly applied to solve the LRS problem. In the following several heuristic rules and algorithms are presented.

3.1. Heuristic rules

We present a storage rule and two release rules as described below. The storage rule can be used for the lane selection, and the release rules can be employed for the car selection.

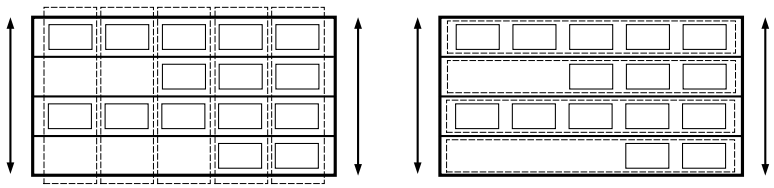
1) Storage rule

The spaces within a selectivity bank can be grouped into blocks in terms of their coordinates. For example, spaces in the same row or column (as depicted in Figure 2) can be considered in one block. A block is marked *safe* if the actual demand for each part (say p) by the cars currently located within this block is below the target demand, which is determined by $r_p \cdot N_s$. Here r_p is the target consumption rate of part p and

N_s is the number of spaces within this block. When a car arrives at the bank, the following storage rule can be applied to assign a lane for the car to enter.

- Storage rule: select the safe block with the highest priority and dispatch the incoming car to an empty space within the block. Otherwise dispatch the car to an empty space within the block with the highest priority.

The storage rule attempts to construct multiple good car subsequences within the selectivity bank so as to facilitate the following release process. Note that for the multiple blocks built by columns (as shown in Figure 2(a)), the block closest to the downstream department has the highest priority. While for the blocks formed by rows (as shown in Figure 2(b)), they can be prioritized based on the index numbers of the associated lanes. Additionally, when there are multiple available empty spaces in the same column within a block, randomly choose one.



(a) Blocking by columns **(b) Blocking by rows**

Fig. 2: Blocks consisting of spaces in a selectivity bank.

The storage rule attempts to construct multiple good car subsequences within the selectivity bank so as to facilitate the following release process. Note that for the multiple blocks built by columns (as shown in Figure 2(a)), the block closest to the downstream department has the highest priority. While for the blocks formed by rows (as shown in Figure 2(b)), they can be prioritized based on the index numbers of the associated lanes. Additionally, when there are multiple available empty spaces in the same column within a block, randomly choose one.

2) Release rule

Consider a selectivity bank filled with T cars. The following two rules, i.e., greedy-based rule and block-based rule, can be used to release the cars successively to the downstream department.

The greedy-based rule iteratively chooses a car to be released next from a candidate set S, which involves all the cars currently at the departure end of each lane. Specifically, at each iteration t ($t = 1, \ldots, T$), the rule selects the car in S that incurs the smallest increase in the objective value (ΔD) of the downstream sequence. ΔD is calculated by

$$\Delta D = \sum\nolimits_{p \in P} \left(z_{pt} - r_p \cdot t \right)^2 . \tag{9}$$

On the other hand, corresponding to the storage rule, block-based rule can be applied to release cars block by block. Specifically, for the blocks generated by columns, cars within each block are released successively based on the same greedy rule as above. While for the blocks generated by rows, only the cars at the departure end of the selectivity bank need to be selected via the greedy rule.

3.2. Heuristic algorithms

This section presents a genetic algorithm (GA) and two beam search (BS) algorithms. The GA is used to search for different schemes of car arrangement within the selectivity bank, and the BS algorithms are used to find the best output sequence given a car arrangement.

1) Genetic algorithm

Invented by John Holland in 1960s, GA is a biologically-inspired metaheuristic for optimization. This paper presents a GA that can be used to seek for the best car arrangement within a selectivity bank. Algorithm 1 shows the pseudocode of the proposed GA.

Algorithm 1: Genetic Algorithm

1: Initialize population
2: **While** iteration < $\text{iteration}_{\max}$
3: Calculate the fitness of individuals
4: Select parental population
5: Perform crossover operations
6: Perform mutation operations
7: Replace selected individuals with the best car arrangement found so far
8: iteration = iteration + 1
9: **End**
10: **Return** the best car arrangement

An individual (chromosome) in Algorithm 1 corresponds to a car arrangement within a $L \times W$ selectivity bank. It consists of L subsequences, each of which involves W position indices of cars (in the upstream sequence) that are dispatched to the same lane. Figure 3 shows an individual representing the arrangement of 9 cars within a 3×3 selectivity bank. During initialization individuals are generated randomly. The fitness function is defined by

$$\text{Fitness} = 1/(D+1), \tag{10}$$

where D is the objective function value of the *optimal* output sequence resulting from a given car arrangement. Apparently a car arrangement with a larger fitness value is preferred.

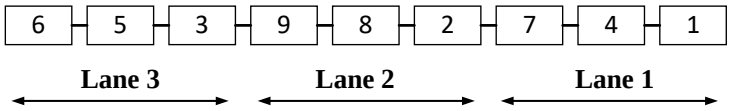


Fig. 3: A chromosome consisting of 3 subsequences.

The selection algorithm used in the proposed GA is roulette wheel, which is based on the principle that the probability for being selected is proportional to the fitness of a solution.

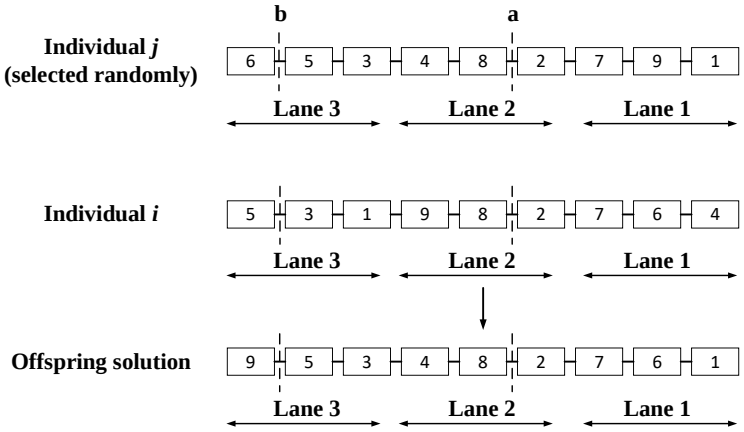


Fig. 4: An illustrative example of two-point crossover.

Two-point crossover is applied in the GA. Each individual i in the population is checked to see whether or not crossover operation should be conducted. If it is needed, another individual j in the population is chosen randomly. Then replace the genes (i.e., index numbers of cars) between two arbitrarily selected points (say a and b) of i with the corresponding genes of j to generate an offspring solution. Finally, substitute the missing genes in i after replacement for the duplicated genes in the offspring solution. As illustrated in Figure 4, cars 1 and 9 take the place of cars 4 and 5, respectively.

Furthermore, an individual can also be considered as a $L \times W$ matrix composed of the position indices of cars in the initial sequence. Therefore, while performing the mutation operation, a new solution can be generated by transposing the elements of the first $\min\{L, W\}$ rows and columns in the original matrix. Note that here rows are numbered successively with the index numbers of lanes, and columns are numbered consecutively from the departure to arrival ends of the selectivity bank. Figure 5 shows that a new solution is obtained by transposing the original 3×3 gene matrix.

Finally, at the end of each iteration, replace a few individuals selected randomly with the best solution found so far, which guarantees that the best solution is always in the population.

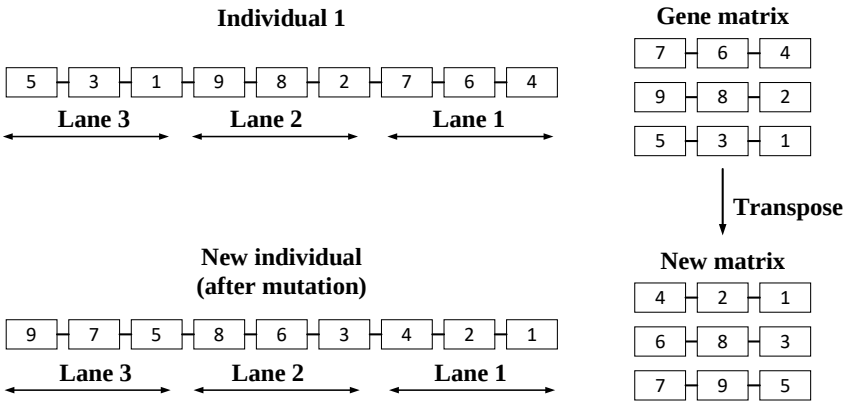


Fig. 5: An illustrative example of mutation operation.

2) Beam search

BS is a truncated graph search heuristic. Originally introduced in solving the speech recognition problems [18], BS has been implemented successfully in multiple fields such as speech recognition [18], machine translation [19], and scheduling and sequencing [20-21].

Based on a tree representation of the search process, BS starts with the root node of stage 0, then constructs the search tree step by step. At each stage, it creates all successors of the current nodes. However, only a predetermined number (known as the beam width, *BW*) of promising nodes are identified by heuristic and retained for further branching. The construction steps are repeated until the final stage is reached. The best solution returned is the result of BS.

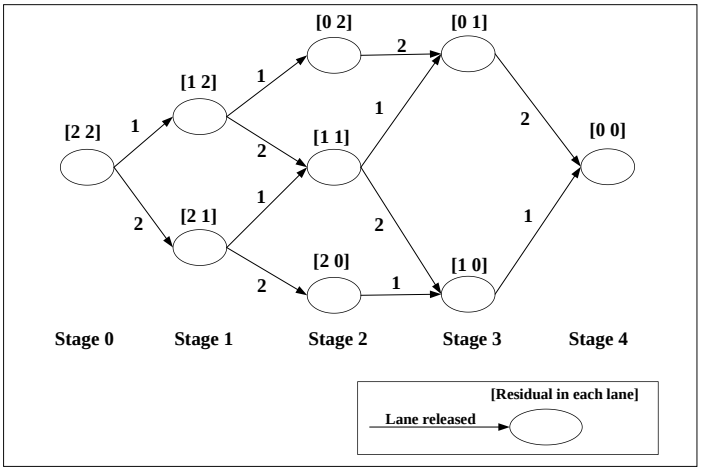


Fig. 6: A graph describing the release process from a 2×2 selectivity bank.

Still consider a selectivity bank filled with *T* cars. Aiming at quickly finding the optimal or near-optimal output sequence, this paper presents two BS algorithms, namely, greedy-based BS and global-based BS. Both algorithms are based on the same graph structure, and the difference between them lies in the filtering process. The graph in Figure 6 illustrates the release process given a 2×2 selectivity bank filled with 4 cars.

The nodes at each stage represent all possible states indicating the current car arrangement within the bank. An arc connecting two nodes at stages t and $t-1$ indicates a state transition between them, and also specifies the lane from which a car is released during this transition.

a) Greedy-based search

At each stage t (t = 1, …, T), all generated nodes are ranked in ascending order of the total deviation (D_t) of actual from target cumulative part demands associated with the partial downstream sequence constructed so far, which can be calculated using the objective function (1). A fixed percentage (say g%) of BW nodes corresponding to the first smallest D_t values are then retained, and the remaining $1-g$% of BW nodes are chosen randomly with the intention of reducing the myopic behaviour of the greedy selection.

b) Global-based search

At each stage t (t = 1, …, T), all nodes newly generated are ranked in ascending order of E_t, which is calculated by

$$E_t = D_t + K_t . \tag{11}$$

Then the nodes corresponding to the first smallest E_t values are kept for further branching. Here E_t in (11) represents an estimated total deviation of actual from target cumulative part demands within the final output sequence. D_t, as defined earlier, is the total deviation of part demands resulting from the partial sequence constructed until now, while Kt estimates the total deviation of part demands caused by unreleased cars. Assume that ideally the actual demand for each part p within each of the remaining $T-t$ production cycles is a constant r_p', i.e., $r_p' = n_p / (T-t)$, where np denotes the number of part p required by the unreleased cars. K_t is calculated by

$$K_t = \sum_{t'=t+1}^{T} \sum_{p \in P} \left(e_{pt'} - r_p \cdot t' \right)^2 , \tag{12}$$

where e_{pt}' represents the estimated actual demand for part p until stage t',

$$e_{pt'} = \left\lfloor (t'-t) \cdot r_{p'} + 0.5 \right\rfloor + q_{pt} . \tag{13}$$

Here q_{pt} is the actual demand for part p up to stage t. Apparently, during the filtering process the global-based BS attempts to evaluate the nodes from a global viewpoint.

4. Computational Experiments

Computational experiments are conducted to test the performance of the heuristic rules and algorithms proposed in this paper. The parameter values used in generating problem instances are given in Table 2. For each of 18 combinations of T, M, and P, 10 problem instances are generated. The models are assumed to be uniformly distributed within the initial sequence. In addition, as suggested by Aigbedo and Monden [23], assume that the demand for each part installed on a car unit of a specific model is 0, 1, 2, 3, or 4, and the quantity is determined based on a probability function of (0.400, 0.410, 0.150, 0.038, 0.002). Corresponding to the T values, three selectivity banks with different configurations ($L \times W$) are used for resequencing respectively.

Table 2: Parameters Used in Experiments

Parameter	Notation	Values
Number of cars	T	30, 56, 100
Number of models	M	5, 10, 20
Number of parts	P	10, 20
Bank configuration	$L \times W$	5×6, 7×8, 10×10

All solution procedures are coded in C++ language with Microsoft's Visual Studio 2015 complier. Experiments are run on a 1.80 GHz PC with 16.0 GB RAM. For the GA algorithm, the population size is set to 20, the number of generations is set to 15. The crossover probability and mutation probability are set to 0.9 and 0.2, respectively. At the end of each iteration, one individual chosen arbitrarily is replaced with the

best car arrangement found so far. Besides, the beam width BW in the two BS algorithms is set to 20. While applying the greedy-based BS, only one node at each stage is selected randomly for further branching. The determination of these parameters are based on existing literature (e.g., [15]) and a series of numerical experiments. Four rule-based resequencing methods (as displayed in Table 3) are generated by combining the presented storage and release rules. Additionally, two algorithm-based resequencing methods are brought by coupling the GA with the BS algorithms. All 180 instances are solved using each of the six combined procedures.

Table 3: Rule-based Resequencing Methods

Rule-based Method	Storage Rule	Release Rule
Method 1	Storage with blocks built by columns	Greedy-based release
Method 2	Storage with blocks built by rows	Greedy-based release
Method 3	Storage with blocks built by columns	Block-based release
Method 4	Storage with blocks built by rows	Block-based release

For each combination of *T*, *M*, and *P*, the averaged objective values over 10 instances associated with each solution procedure are recorded, and the corresponding percentage reductions in the objective value after resequencing are listed in Table 4. It shows that all the four rule-based procedures can effectively improve the part usage rates of the initial sequences, resulting in a noticeable reduction (from 22% to 78%) in the objective value. Besides, method 1 has the best overall performance, followed by methods 3, 2, and 4.

On the other hand, it displays that the two algorithm-based procedures produce comparable results for all instances in acceptable running times. Moreover, both procedures obtain substantially better part usage rates derived from the downstream sequences, resulting in an impressive reduction of at least 79% in the objective value. Apparently, the rule-based procedures are inferior to the algorithm-based procedures in resequencing result, however they can solve the LRS problem instances in a much shorter time duration of less than 0.5 s.

Table 4: Results of Six Resequencing Procedures

(*T*, *M*, *P*)	Ave. Obj. Value before Reseq.	Rule-based Method 1	Rule-based Method 2	Rule-based Method 3	Rule-based Method 4	GA + Greedy-based BS		GA + Global-based BS	
		% reduction in ave. obj. value after reseq.	*% reduction in ave. obj. value after reseq.*	*% reduction in ave. obj. value after reseq.*	*% reduction in ave. obj. value after reseq.*	*% reduction in ave. obj. value after reseq.*	*Ave. CPU time (s)*	*% reduction in ave. obj. value after reseq.*	*Ave. CPU time (s)*
(30, 5, 10)	714.90	77.9%	54.3%	72.8%	52.0%	90.7%	5.90	90.7%	6.60
(30, 10, 10)	885.62	57.5%	35.7%	52.7%	31.9%	85.1%	5.68	85.1%	6.56
(30, 20, 10)	1044.11	62.3%	52.5%	37.2%	41.1%	84.8%	5.58	85.2%	6.53
(30, 5, 20)	1634.21	66.3%	51.2%	60.9%	66.9%	90.7%	5.94	90.7%	6.99
(30, 10, 20)	1640.12	58.6%	41.7%	36.7%	29.4%	81.3%	5.84	81.4%	7.13
(30, 20, 20)	2155.77	62.6%	45.7%	38.5%	42.5%	79.0%	5.75	79.7%	7.21
(56, 5, 10)	3103.00	65.6%	64.3%	74.2%	41.2%	92.6%	22.87	92.8%	24.68
(56, 10, 10)	3523.75	63.2%	65.3%	57.6%	21.6%	91.6%	22.95	91.6%	25.40
(56, 20, 10)	2883.43	61.8%	61.4%	46.8%	39.9%	89.6%	22.20	90.1%	25.55
(56, 5, 20)	5139.24	68.9%	65.9%	77.5%	48.1%	94.0%	22.70	94.3%	27.43
(56, 10, 20)	8832.04	72.0%	59.4%	54.7%	50.2%	91.4%	22.21	91.5%	27.12
(56, 20, 20)	7035.78	62.0%	57.7%	43.2%	26.8%	84.8%	22.22	85.3%	28.76
(100, 5, 10)	6528.96	69.5%	68.7%	69.9%	60.0%	96.0%	78.76	96.2%	88.43
(100, 10, 10)	8164.32	71.6%	52.5%	63.2%	52.4%	93.9%	78.22	94.7%	89.33
(100, 20, 10)	10744.49	76.4%	67.3%	53.0%	55.3%	94.9%	81.62	95.2%	90.16
(100, 5, 20)	16272.30	64.8%	53.2%	72.2%	58.2%	95.5%	80.01	95.7%	100.67
(100, 10, 20)	15516.92	71.6%	45.6%	46.2%	22.9%	91.7%	79.30	93.2%	99.21
(100, 20, 20)	24248.09	76.4%	65.6%	47.5%	42.7%	92.0%	74.91	92.8%	98.47

In summary, the proposed heuristic rules and algorithms can be jointly applied to solve the LRS problem effectively and efficiently. Furthermore, the algorithm-based procedures perform significantly better than the rule-based procedures.

5. Summary and Conclusions

This paper studies the LRS problem prior to automotive final assembly with selectivity banks. A mathematical programming model was presented to describe the problem. Several heuristic rules and algorithms that can be applied in the car storage and release stages respectively were developed. Numerical experiments were carried out to evaluate the performance of the proposed solution procedures. It was shown that they can be jointly applied in resequencing, so that the part usage rates in initial sequences can be effectively improved after permutation. Besides, the solution approaches combining the GA and BS algorithms significantly outperform the rule-based solution approaches.

In the future, research efforts can be made to improve the algorithms proposed in this paper. Specifically, for the GA algorithm, different coding methods and genetic operations can be considered. While for the BS algorithms, iterated search can be performed, meanwhile a strongly tightened lower bound of the objective value associated with a beam node needs to be determined. On the other hand, further exploration of more complex scenarios of resequencing with selectivity banks, such as multi-objective resequencing or real-time resequencing, would also be a valuable contribution.

6. References

[1] N. Boysen, M. Fliedner, and A. Scholl. Sequencing mixed-model assembly lines: Survey, classification and model critique. *Eur. J. Oper. Res.* 2009, **192**(2): 349-373.

[2] F. Ding, and H. Sun. Sequence alteration and restoration related to sequqenced parts delivery on an automobile mixed-model assembly line with multiple departments. *Int. J. Prod. Res*. 2004, **42**(8): 1525-1543.

[3] N. Boysen, A. Scholl, and N. Wopperer. Resequencing of mixed-model assembly lines: Survey and research agenda. *Eur. J. Oper. Res*. 2012, **216**(3): 594-604.

[4] T. Epping, W. Hochstattler, and P. Oertel. Complexity results on a paint shop problem. *Discrete Appl. Math*. 2004, **136**(2-3): 217-226.

[5] Y. Xu, and J. Zhou. A virtual resequencing problem in automobile paint shops. In: E. Qi, J. Shen, and R. Dou (eds.). *Proc. of the 22nd International Conference on Industrial Engineering and Engineering Management: Core Theory and Applications of Industrial Engineering*. Paris: Atlantis Press. 2016, pp. 71-80.

[6] C. Cao, and H. Sun. Virtual level rescheduling for automotive mixed-model assembly lines with beam search algorithms. In: *Proc. of 2019 IEEE 6th International Conference on Industrial Engineering and Applications*. New York: IEEE. 2019, pp. 225-229.

[7] S. Spieckermann, K. Gutenschwager, and S. Voß. A sequential ordering problem in automotive paint shops. *Int. J. Prod. Res*. 2004, **42**(9): 1865-1878.

[8] H. Sun, and J. Han. A study on implementing color-batching with selectivity banks in automotive paint shops. *J. Manuf. Syst*. 2017, **44**: 42-52.

[9] J. Leng, C. Jin, A. Vogl, and H. Liu. Deep reinforcement learning for a color-batching resequencing problem. *J. Manuf. Syst*. 2020, **56**: 175-187.

[10] S. Bysko, J. Krystek, and S. Bysko. Automotive Paint Shop 4.0. *Comput. Ind. Eng.* 2020, **139:** 105546.

[11] X. Fournier and B. Agard. Improvement of earliness and lateness by postponement on an automotive production line. *Int. J. Flexible Manuf. Syst.* 2007, **19**(2): 107-121.

[12] S. Meissner. Controlling in just-in-sequence flow-production. *Logistics Research*. 2010, **2**(1): 45-53.

[13] W. Choi, and H. Shin. A real-time sequence control system for the level production of the automobile assembly line. *Comput. Ind. Eng.* 1997, **33**(3-4): 769-772.

[14] D. Moon, C. Song, and J. Ha. A dynamic algorithm for the control of automotive painted body storage. *Simulation*. 2005, **81**(11): 773-787.

[15] N. Boysen, and M. Zenker. A decomposition approach for the car resequencing problem with selectivity banks. *Comput. Oper. Res*. 2013, **40**(1): 98-108.

[16] W. Kubiak. Minimizing variation of production rates in just-in-time systems: A survey. *Eur. J. Oper. Res*. 1993, **66**(3): 259-271.

[17] T. Dahmala, and W. Kubiak. A brief survey of just-in-time sequencing for mixed-model systems. *Int. J. Oper. Res*. 2005, **2**(2): 38-47.

[18] Z. Jin, and F. Ding. A transformed two-stage method for reducing the part-usage variation and a comparison of the product-level and part-level solutions in sequencing mixed-model assembly lines. *Eur. J. Oper. Res*. 2000, **127**(1): 203-216.

[19] B. Lowerre. The Harpy Speech Recognition System. *PhD thesis*. 1976.

[20] P. Koehn. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In: R. E. Frederking, and K.B. Taylor (eds.). *Machine Translation: From Real Users to Research. AMTA 2004. Lecture Notes in Computer Science*. Berlin, Heidelburg: Springer. 2004, pp. 115-124.

[21] I. Sabuncuoglu, and M. Bayiz. Job shop scheduling with beam search. *Eur. J. Oper. Res.* 1999, **118**(2): 390-412.

[22] E. Erdal, G. Yasin, and S. Ihsan. Mixed-model assembly line sequencing using beam search. *Int. J. Prod. Res.* 2007, **45**(22): 5265-5284.

[23] H. Aigbedo, and Y. Monden. A simulation analysis for two-level sequence-scheduling for just-in-time (JIT) mixed-model assembly lines. *Int. J. Prod. Res*. 1996, **34**(11): 3107-3124.