Heuristic algorithms for a delivery workload balancing problem in an assembly plant

Yoshiyuki KARUNO* and Kento FURUKAWA**

* Faculty of Mechanical Engineering, Kyoto Institute of Technology
Goshokaido-cho, Matsugasaki, Sakyo-ku, Kyoto-shi, Kyoto 606-8585, Japan
E-mail: karuno@kit.ac.jp

** Graduate School of Science and Technology, Kyoto Institute of Technology
Goshokaido-cho, Matsugasaki, Sakyo-ku, Kyoto-shi, Kyoto 606-8585, Japan

Received 17 February 2016

Abstract
In this paper, a delivery workload balancing problem in an assembly plant is considered. The problem is first described as a special case of the unrelated parallel machine scheduling problem of minimizing the makespan. Then, a polynomial time heuristic algorithm is proposed, which is regarded as a dynamic programming procedure to compute a linear partition of a specified list of given jobs. In addition, a simple improvement procedure based on local search technique is discussed. Numerical results indicate that from the viewpoints of solution quality and execution time both, the proposed heuristic is applicable in the practical situation.

Key words: Engineering optimization, Scheduling, Delivery workload balancing, Linear partitioning, Dynamic programming, Local search

1. Introduction

We consider a combinatorial optimization problem which might hide in a delivery workload management system of an automobile assembly plant (see Fukuyama et al., 1995). A large number of cargoes were brought from several machine shops of manufacturing parts to the cargo operating area of the assembly plant everyday. The cargoes were delivered by a tractor from the cargo operating area to an assembly line laid in the assembly plant. The tractor performed a number of such delivery routings periodically per day. In order to utilize the cargo operating area, the assembly plant divided it into a few of lanes, and the delivery workload management system determined which lane is assigned to each cargo. Following Fukuyama et al. (1995), we depict a schematic of the cargo operating area in Fig. 1, where the number of available lanes for each cargo is three. The delivery workload management system might face more complicated issues, since such a practical management system should generally treat many other kinds of hard constraints. However, the selection of a lane for each cargo was even one of most significant components of the delivery workload management system, and it is very interesting to consider the lane selection issue in a mathematical manner.

In this paper, we view the lane selection issue as a special case of the unrelated parallel machine scheduling problem of minimizing the makespan (e.g., see Vazirani, 2001). A job corresponds to a cargo, and a machine does to a delivery routing. A solution is defined to be a partition of a given set of jobs, where the partition number (in other words, the number of disjoint subsets of jobs) is set to be the number of available machines. The general form of the unrelated parallel machine scheduling problem of minimizing the makespan and even the identical parallel machine scheduling problem of minimizing the makespan have been known to be NP-hard. The special case of the unrelated parallel machine scheduling problem of minimizing the makespan to be discussed in this paper unfortunately still contains the NP-hard identical parallel machine scheduling problem of minimizing the makespan from the viewpoint of mathematical modeling. We design a heuristic algorithm based on a polynomial time dynamic programming procedure of linear partitioning (e.g., see Skiena, 2008). Since we have to regard the number of available lanes in the cargo operating area of the assembly plant, the proposed heuristic seeks for the best linear partition of a specified list of jobs among a restricted subset of linear
partitions. Due to this, we make some modifications in the basic dynamic programming procedure of linear partitioning, while the time complexity of the proposed heuristic remains polynomial.

We also discuss a simple local search procedure to improve a heuristic solution (e.g., a solution obtained by the modified dynamic programming procedure of linear partitioning). Given a heuristic solution, we see that exchanging the positions of two distinct jobs such that they have the same possible choices for selecting a machine delivers another heuristic solution. From this fact, we are going to define a square order neighborhood of a heuristic solution in an application of local search technique. The local search procedure terminates its computation process when there is no improvement in the neighborhood of the current heuristic solution. Generally, it should be remarked that the solution quality of a local search procedure may depend on the choice of the initial solution (e.g., see Ibaraki, 1989). Empirical performance of the proposed dynamic programming heuristic algorithm and that of the local search procedure are examined by conducting numerical experiments, and the results are reported.

![Fig. 1 An Illustration of Available Lanes for a Cargo in the Cargo Operating Area](image-url)

2. Problem Description

2.1. The Unrelated Parallel Machine Scheduling Problem

We first describe the general form of the unrelated parallel machine scheduling problem of minimizing the makespan. We use an abbreviation Unrelated \(\text{PMS} \) (Parallel Machine Scheduling) for short. In this paper, the makespan means the maximum of a machine load. When all available machines are assumed to be identical (and the processing times of a job on the machines are the same), the problem is known to be the identical parallel machine scheduling one. We also use Identical \(\text{PMS} \) as the abbreviation.

We are given a set of \( m \) machines and a set of \( n \) jobs. Let \( M = \{i \mid i = 1, 2, \ldots, m\} \) and \( N = \{j \mid j = 1, 2, \ldots, n\} \) denote the machine set and job set, respectively. Let \( p_{ij} > 0 \) denote an integral processing time of job \( j \in N \) on machine \( i \in M \). Each machine can process at most one job at a time, and each job must be processed by exactly one of the \( m \) machines. No preemption of the processing is allowed.

In this paper, a feasible solution of problem Unrelated \(\text{PMS}\) is represented by a partition of the given job set \( N \) into disjoint subsets \( J_i \) with \( i = 1, 2, \ldots, m \). In a feasible solution \( J = (J_1, J_2, \ldots, J_m) \), jobs in \( J_i \) are assigned to machine \( i \in M \). We assume that the jobs in \( J_i \) are processed in an arbitrary order of them on the machine. Let

\[
p(J_i) = \sum_{j \in J_i} p_{ij}
\]

denote the load of machine \( i \in M \). The objective is to minimize the maximum of a machine load, which is denoted by

\[
\ell_{\text{max}}(J) = \max_{1 \leq i \leq m \{ p(J_i) \}}.
\]

For a more detailed explanation, let \( \ell_{\text{max}} \) denote the minimum of the maximum machine load for a given instance of problem Unrelated \(\text{PMS} \). Problem Unrelated \(\text{PMS}\) asks to find an optimal solution \( J^* = (J_1^*, J_2^*, \ldots, J_m^*) \) such that it is a feasible solution, and it attains \( \ell_{\text{max}}(J^*) = \ell_{\text{max}}^* \). Problem Unrelated \(\text{PMS}\) has been known to be NP-hard (e.g., see Garey and Johnson, 1979), while it has also been known to be polynomially 2-approximable (e.g., see Lenstra et al., 1990; Vazirani, 2001). In this paper, we consider the following special case of problem Unrelated \(\text{PMS}\).
2.2. Formulation of the Special Case

In the special case of problem Unrelated PMS, we regard the notation $M$ as a list of $m$ machines. That is, it also implies the sequence $M = (1, 2, \ldots, m)$ of machines. Let $M[s, t]$ denote a subsequence of machines from machine $s \in M$ to machine $t \in M$ with $1 \leq s \leq t \leq m$, which we call a partial machine list of $M$. Each job $j \in N$ relates to a partial list $M[s_j, t_j] \subseteq M$ of preferable machines with $1 \leq s_j \leq t_j \leq m$. Let $p_j > 0$ denote the processing time of each job $j \in N$ on a preferable machine $i \in M[s_j, t_j]$ of the job, and further, let $B > 0$ denote a sufficiently large integral constant. Then, the processing time $p_{ij}$ of job $j \in N$ on machine $i \in M$ is defined by

$$p_{ij} = \begin{cases} p_j & \text{if } i \in M[s_j, t_j], \\ B & \text{otherwise.} \end{cases}$$

Moreover, for any job $j \in N$ with the first preferable machine $s_j$ and for a given integral constant $L$ with $1 \leq L \leq m$, the last preferable machine of the job is defined by

$$t_j = \min\{m, s_j + L - 1\},$$

and renumber the $n$ jobs so that they meet

$$1 = s_1 \leq s_2 \leq \cdots \leq s_n \leq m,$$

and hence it also holds

$$L = t_1 \leq t_2 \leq \cdots \leq t_n \leq m.$$

Note that for each job $j \in N$, the preferable machine list is non-empty, i.e., $M[s_j, t_j] \neq \emptyset$, since it holds $s_j \leq t_j$ by Eq. (2). In the practical situation, the positive integer $L$ means the number of available lanes in the cargo operating area (e.g., $L = 3$ in Fig. 1), and there are at most $L$ possible delivery routings for each cargo (e.g., in Fig. 1, there are three possible delivery routings for the first cargo by $M[s_1, t_1] = M[1, 3] = (1, 2, 3)$). On the other hand, the number $m$ of machines indicates the number of delivery routings in the planning horizon (e.g., per day).

We refer to the special case of the unrelated parallel machine scheduling problem of minimizing the maximum machine load as Preferential PMS for short. As problem Unrelated PMS, a feasible solution of problem Preferential PMS is represented by a partition of the given job set $N$ into disjoint subsets $J_i$ with $i = 1, 2, \ldots, m$. Also we refer to a feasible solution $J = J' = (J_1', J_2', \ldots, J_m')$ as an optimal one if it attains $\ell_{\text{max}}(J') = \ell_{\text{max}}^*$.

**Proposition 1.** For an instance of problem Preferential PMS, there exists a feasible solution $J = (J_1, J_2, \ldots, J_m)$ such that it satisfies $\ell_{\text{max}}(J) < B$.

**Proof.** Recall that each job $j \in N$ has at least one preferable machine $s_j \in M$. Consider a solution $J = (J_1, J_2, \ldots, J_m)$ such that for each machine $i = 1, 2, \ldots, m$, $J_i = \{j \in N \mid s_j = i\}$. That is, each job is assigned to a machine by the first preferable machine rule (for short, FPM rule). In the solution $J$, each job $j \in N$ requires the processing time $p_j$ on the machine $s_j$. Moreover, the solution $J$ is feasible (i.e., subsets $J_i$ with $i = 1, 2, \ldots, m$ are disjoint, and it holds $J_1 \cup J_2 \cup \cdots \cup J_m = N$). The integral constant $B$ is assumed to be sufficiently large, and hence we have $\ell_{\text{max}}(J) \leq \sum_{j \in N} p_j < B$. (Q.E.D.)

**Proposition 2.** For an instance of problem Preferential PMS, there exists an optimal solution $J' = (J_1', J_2', \ldots, J_m')$ with $J_1' \neq \emptyset$.

**Proof.** Suppose that $J' = (J_1', J_2', \ldots, J_m')$ is an optimal solution with $J_1' = \emptyset$. Then, one of subsets $J_2', \ldots, J_m'$ contains job 1 with $s_1 = 1$ (see Eq. (3)), which is denoted by $J_h$ with $2 \leq h \leq L$, since the maximum machine load of the optimal solution $J'$ meets $\ell_{\text{max}}(J') < B$ by Proposition 1. Construct a solution $J' = (J_1', J_2', \ldots, J_m')$ from the optimal solution $J'$ by moving job 1 from $J_h$ to $J_1'$. That is, $J_1 = \{1\}$, $J_h' = J_h \setminus \{1\}$, and $J_i' = J_i$ for each $i \in M \setminus \{1, h\}$. For the modified solution $J'$, it holds $\max(p(J_1'), p(J'_h)) \leq p(J'_h)$, and $p(J_i') = p(J_i)$ for each $i \in M \setminus \{1, h\}$. Hence, we have $\ell_{\text{max}}(J') \leq \ell_{\text{max}}(J')$, which implies the $J'$ with $J_1' \neq \emptyset$ is also an optimal solution. (Q.E.D.)
2.3. Lane Selection in the Cargo Operating Area

As mentioned before, in our mathematical modeling, a job corresponds to a cargo which has been brought from a machine shop of manufacturing parts to the assembly plant. Each cargo should be thrown into exactly one of $L$ lanes, which the delivery workload management system has selected for the cargo (see the schematic of Fig. 1). A machine corresponds to a periodic delivery routing from the cargo operating area to an assembly line performed by a tractor. The entire delivery work of the tractor in the planning horizon consists of $m$ periodic delivery routings. In each periodic delivery routing, a subset of cargoes is picked up by the tractor. That is, each subset $J_i$ of jobs indicates the subset of cargoes picked up in the $i$-th delivery routing, and machine load $p(J_i)$ indicates the workload of the $i$-th delivery routing. A processing time of each job in the mathematical model may be estimated, for example, according to the number of loading operations required for the driver of tractor to supply the parts of the corresponding cargo into the racks situated along the assembly line.

From the above relationship between the unrelated parallel machine scheduling problem and the actual cargo operating area, it can be seen that each cargo which is picked up in the $i$-th delivery routing has been assigned to the lane with index $((i-1) \mod L) + 1$ (e.g., again see Fig. 1, in which the cargoes of the fifth delivery routing are assigned to the second lane, since $L = 3$). Moreover, let $\Delta > 0$ denote the time interval between any two consecutive delivery routings in the planning horizon, and let $a_j$ denote an arrival time of cargo $j$ in the cargo operating area such that $0 \leq a_j < m \times \Delta$ for each $j = 1, 2, \ldots, n$. Then, the first preferable machine of the corresponding job $j \in N$ is specified by $s_j = \lfloor a_j / \Delta \rfloor + 1$.

Before the assembly plant divided the cargo operating area into a few of lanes, constructing a subset of cargoes to be picked up for a delivery routing depended on the decision making of the driver of tractor (see Fukuyama et al., 1995). Due to such a delivery workload assignment, it was very hard for the assembly plant to know the progress of delivery work from the cargo operating area to an assembly line at any time of a day. It was the reason why the assembly plant wanted to introduce a delivery workload management system.

3. The Proposed Heuristic

In this section, we regard the notation $N$ for given $n$ jobs as a list, as well as the $M$ for given $m$ machines. That is, $N = (1, 2, \ldots, n)$. Recall that we have renumbered the $n$ jobs in $O(n \log n)$ time so that they satisfy Eq. (3). We assume in the first part of this section that the preferable machine set is given by $M[s_j, t_j] = M$ for each job $j = 1, 2, \ldots, n$ (i.e., the setting is the same as problem Identical_PMS), and hence it holds $p_{ij} = p_j$ for each machine $i = 1, 2, \ldots, m$ (see Eq. (1)). This makes it easy to understand the basic dynamic programming procedure of linear partitioning (e.g., see Skiena, 2008).

After that, we describe the proposed heuristic algorithm for problem Preferential_PMS by modifying the basic dynamic programming procedure.

3.1. Basic Dynamic Programming Procedure

Let $N[j, k]$ denote a subsequence of jobs in the job list $N$ from $j \in N$ to $k \in N$ with $1 \leq j \leq k \leq n$. We also call $N[j, k]$ a partial job list of $N$ from job $j$ to job $k$. The definition implies $N[1, n] = N$. A linear partition of the job list $N$ with partition number $m$ is represented by a set of $m$ disjoint and consecutive partial job lists,

- $N[1, \delta[1]]$,
- $N[\delta[1] + 1, \delta[2]]$,
- $\cdots$
- $N[\delta[m-1] + 1, \delta[m]]$,

where $\delta[i] \in N$ denotes a delimiting job of the $i$-th region $N[\delta[i-1] + 1, \delta[i]]$ for $i = 1, 2, \ldots, m$ such that it meets

$$1 \leq \delta[1] \leq \delta[2] \leq \cdots \leq \delta[m] = n,$$

and let $\delta[0] = 0$ for notational convenience. We construct a solution $J$ from a linear partition such that

$$J_i = [\delta[i-1] + 1, \delta[i-1] + 2, \ldots, \delta[i]]$$

for each $i = 1, 2, \ldots, m$.

By the definition of a linear partition, it satisfies that $J_h \cap J_i = \emptyset$ for $1 \leq h < i \leq m$, and that $J_1 \cup J_2 \cup \cdots \cup J_m = N$. Hence, the $J$ is a feasible solution. In addition, it can be viewed as a heuristic solution for a given instance of problem Identical_PMS, while the problem has been known to be polynomially 4/3-approximable by the largest processing time first rule (e.g., see Graham, 1969; Pinedo, 1995).
Notice that finding a linear partition of the job list \( N \) is to determine the \( m \) delimiting jobs \( \delta[1], \delta[2], \ldots, \delta[m] \). For \( 1 \leq i \leq m \) and for \( 1 \leq j \leq n \), let \( z[i, j] \) denote the minimum of the maximum sum of processing times for a subsequence of jobs in a linear partition of partial job list \( N[1, j] \) with partition number \( i \). Then, the \( z[m, n] \) means the minimum of the maximum machine load of a feasible solution corresponding to a linear partition of the job list \( N \) with the number \( m \) of machines. A known recursive computation process is described as follows (see Skiena, 2008).

The boundary conditions are expressed by

\[
\begin{align*}
z[1, j] &= \sum_{h=1}^{j} p_h \quad \text{for all } j = 1, 2, \ldots, n, \\
z[i, 1] &= p_1 \quad \text{for all } i = 1, 2, \ldots, m, 
\end{align*}
\]

which are obvious by the definition of \( z[i, j] \). For \( 2 \leq i \leq m \) and \( 2 \leq j \leq n \), it holds

\[
z[i, j] = \min_{1 \leq k < j} \left\{ \max \left( z[i - 1, k], \sum_{h=k+1}^{j} p_h \right) \right\}. 
\]

(7)

The recursive computation process by Eqs. (5)–(7) can calculate the value \( z[m, n] \), and it is regarded as a dynamic programming procedure, which we simply call procedure DP. The \( k = k^* \) taking the minimum in Eq. (7) for \( i = m \) and \( j = n \) indicates the delimiting job of the \((m - 1)\)-th region, i.e., \( \delta[m - 1] = k^* \), and \( k^* + 1 \) is the head job of the \( m \)-th region.

For a more efficient computation, the value

\[
c_j = \sum_{h=1}^{j} p_h 
\]

is prepared at the initialization step. For notational convenience, let \( c_0 = 0 \). Utilizing the cumulative values, the sum of processing times of partial job list \( N[j, k] \) from job \( j \) to job \( k \) can be referred in \( O(1) \) time by \( c_k - c_{j-1} \) for \( 1 \leq j \leq k \leq n \).

The above description follows a textbook written by Skiena (2008). Finding the minimum with respect to the \( k \) in Eq. (7) requires \( O(n) \) time, and hence procedure DP runs in \( \{O(m) \times O(n)\} \times O(n) = O(mn^2) \) time for a given number \( m \) of machines.

### 3.2. Modified Dynamic Programming Procedure

For an instance of problem Preferential_PMS, we have seen from Proposition 1 that an optimal solution assigns a job \( j \in N \) to a preferable machine in \( M[j, t_j] \). When we apply procedure DP provided in the previous subsection to an instance of problem Preferential_PMS, we exclude from the consideration a linear partition of the job list \( N \) such that some job is assigned to a non-preferable machine.

A job \( j \in N \) with either \( t_j < i \) or \( s_j > i \) cannot be assigned to machine \( i \in M \) in an optimal solution. Let \( u_i \) and \( v_i \) denote the lower and upper indexes of jobs which may be possible to be assigned to machine \( i \in M \), respectively. Precisely, for each machine \( i = 1, 2, \ldots, m \), the lower and upper bounds are defined by

\[
u_i = \max\{j \mid s_j \leq i, \ j = 1, 2, \ldots, n\},
\]

(9)

Notice that from the monotonousness of machine indexes \( s_j \) and also that of \( t_j \) (see Eq. (3)), a job \( j \in N \) with either \( j < u_i \) or \( j > v_i \) cannot be assigned to machine \( i \in M \) in an optimal solution. Then, if \( u_i > v_i \) for a certain machine \( i \in M \), we see that for the machine \( i \), it holds

\[
\{j \in N \mid i \in M[j, t_j]\} = \emptyset,
\]

i.e., no job prefers the machine \( i \in M \). On the other hand, if \( u_i \leq v_i \) for a certain machine \( i \in M \), we can show that for the machine \( i \), there is at least one job \( j \in N \) with \( i \in M[j, t_j] \). In fact, there exists a job \( j \in N \) with \( u_i \leq j \leq v_i \) such that it clearly satisfies \( s_j \leq i \leq t_j \).

In order to assign a job \( j \in N \) to a preferable machine in \( M[j, t_j] \), the boundary conditions Eqs. (5) and (6) are modified by

\[
z[1, j] = \begin{cases} 
\sum_{h=1}^{j} p_h & \text{for } j = 1 (= u_1), \ldots, v_1, \\
\text{(undefined)} & \text{for } j = v_1 + 1, \ldots, n,
\end{cases}
\]

(10)
\[ z[i, 1] = p_i \quad \text{for all } i = 1, 2, \ldots, m. \quad (11) \]

The recursive for \( 2 \leq i \leq m \) and \( 2 \leq j \leq n \) in Eq. (7) is also modified by

\[
 z[i, j] = \begin{cases} 
 z[i-1, j] & \text{for } j = 2, \ldots, u_i - 1, \\
 \min_{\rho \in \{0, 1\}} \left\{ \max\left( z[i-1, k], \sum_{h=k+1}^{j} p_h \right) \right\} & \text{for } j = u_i, \ldots, v, \\
 (\text{undefined}) & \text{for } j = v + 1, \ldots, n,
\end{cases}
\quad (12)
\]

where the parameters indicating the interval of seeking for the \( k \) which takes the minimum are defined by

\[ a[i] = \max(\min\{u_{i-1}, v_{i-1}\}, u_i - 1), \]

\[ \tau[i, j] = \min\{v_{i-1}, j\}. \]

For the case of \( j < u_i \) in Eq. (12) (during the loop of a certain partition number \( i \in \{2, 3, \ldots, m\} \)), any job \( k \in N[i, j] \) satisfies \( t_k < i \). Hence, we should set the \( i \)-th region to be an empty region where no job is contained, which implies that \( z[i, j] = \max\{z[i-1, j], 0\} = z[i-1, j] \).

For the case of \( j > v \) in Eq. (12), a job \( k \in [v_i + 1, v_i + 2, \ldots, j] \) meets \( s_k > i \). Hence, we should not assign such a job \( k \) to any of the first \( i \) regions. For the same reason, we should not assign a job \( k \in [v_i - 1, v_i - 1 + 2, \ldots, j] \) to any of the first \( (i-1) \) regions. Thus, for the case of \( u_i \leq j \leq v_i \) in Eq. (12), we recursively define \( \tau[i, j] = \min\{v_{i-1}, j\} \) as the upper limit of the delimiting job of the \((i-1)\)-th region.

Also for the case of \( u_i \leq j \leq v_i \) in Eq. (12), the head job \( h \) of the \((i-1)\)-th region should satisfy \( h \geq u_i \). Hence, the delimiting job \( k \) of the \((i-1)\)-th region should meet \( k \geq u_i - 1 \). Further, if it holds \( u_{i-1} \leq v_{i-1} \), then for a job \( k' \in N \) with \( k' < u_{i-1} \), we recursively see that \( z[i-1, k'] = \max\{z[i-2, k'], 0\} = z[i-2, k'] \). This means that the \((i-1)\)-region is an empty one. Hence, together with Proposition 2, it suffices to regard each job \( k \in N \) with \( k \geq u_{i-1} \) as the delimiting job \( k \) of the \((i-1)\)-th region. On the other hand, if it holds \( u_{i-1} > v_{i-1} \) (i.e., no job prefers machine \((i-1) \in M\)), then the job \( k' = v_{i-1} < u_{i-1} \) satisfies \( t_{k'} < i - 1 \) and \( s_{k'+1} = i > i - 1 \). (Suppose that it holds \( s_{k'+1} > i \). Then we have \( u_i > v_i \), which is a contradiction.) Hence, the head job \( h \) of the \((i-1)\)-th region should satisfy \( h \geq v_{i-1} + 1 \), and the delimiting job \( k \) of the \((i-1)\)-th region should meet \( k \geq v_{i-1} \). Thus, for the case of \( u_i \leq j \leq v_i \) in Eq. (12), we recursively define \( a[i] = \max(\min\{u_{i-1}, v_{i-1}\}, u_i - 1) \) as the lower limit of the delimiting job of the \((i-1)\)-th region.

We refer to the modified version of procedure DP as algorithm MDP for short. Since we can evaluate \( O(v_i) = O(n) \) for each \( i = 2, 3, \ldots, m \), finding the minimum with respect to the \( k \) in Eq. (12) requires \( O(n) \) time. Hence, as procedure DP, algorithm MDP runs in \( O(mn^2) \) time. The \( O(n \log n) \) time renumbering of jobs (see Eq. (3)) is dominated by the main body of the modified dynamic programming procedure.

**Proposition 3.** For an instance of problem Preferential_PMS, algorithm MDP obtain a feasible solution \( J = (J_1, J_2, \ldots, J_m) \) such that it satisfies \( \ell_{\text{max}}(J) < B \) in \( O(mn^2) \) time.

### 4. Local Search Based Improving

In the actual assembly plant, a neighborhood search procedure was applied to the delivery workload management system (see Fukuyama et al., 1995). However, it did not seem to care for the choice of the initial solution. In this section, we design a simple local search procedure. In the following section, we will examine two alternatives, the first preferable machine (FPM) rule (see Eq. (4)) and the modified dynamic programming procedure (MDP) of linear partitioning, as the initial solution of the local search procedure by conducting numerical experiments.

Let \( J = (J_1, J_2, \ldots, J_m) \) be a feasible solution for an instance of problem Preferential_PMS such that it satisfies \( \ell_{\text{max}}(J) < B \) (e.g., see Propositions 1 and 3). Consider a job \( j \in N \) whose preferable machine set is \( M[s_j, t_j] \subseteq M \), and suppose that the job \( j \) is assigned to a preferable machine \( i \in M[s_j, t_j] \) in the solution \( J \). Then, by moving job \( j \) from machine \( i \) to another preferable machine \( i' \in M[s_j, t_j] \setminus \{i\} \), we obtain a feasible solution \( J' \neq J \) with \( \ell_{\text{max}}(J') < B \). We call such a solution \( J' \) a neighborhood solution of the current solution \( J \).

Also consider two distinct jobs \( j \in N \) and \( k \in N \) such that they satisfy \( M[s_j, t_j] = M[s_k, t_k] \), and suppose that in the current solution \( J \), job \( j \) is assigned to a preferable machine \( i \in M[s_j, t_j] \), while the other job \( k \) is assigned to a different
preferable machine \( j' \in M \setminus \{ i \} \). Then, by exchanging the positions of the two distinct jobs (that is, moving job \( j \) from machine \( i \) to machine \( j' \), while moving job \( k \) from machine \( i' \) to machine \( i \)), we obtain a feasible solution \( J' (\neq J) \) with \( \ell \max(J') < B \). We also call such a solution \( J' \) a neighborhood solution of the current solution \( J \). For convenience, we refer to the former as a shifting neighborhood solution, and to the latter as an exchanging one.

Let \( N_{\delta}(J) \) denote the set of shifting neighborhood solutions of a feasible solution \( J \) with \( \ell \max(J) < B \), and let \( N_{\delta}(J) \) denote the set of exchanging neighborhood solutions of the feasible solution \( J \). Then, \( N_{\alpha}(J) = N_{\delta}(J) \cup N_{\delta}(J) \) denotes the neighborhood of the feasible solution \( J \) to be defined in this paper. If there exists a neighborhood solution \( J' \in N_{\alpha}(J) \) of the current solution \( J \) such that \( \ell \max(J') < \ell \max(J) \), we say that there is an improvement in \( N_{\alpha}(J) \). For a feasible solution \( J \) with \( \ell \max(J) < B \), it holds \( |N_{\alpha}(J)| = O(n^2) \).

The local search procedure to be designed in this paper starts from a feasible solution \( J \) with \( \ell \max(J) < B \) such as ones obtained by FPM rule (see Eq. (4)) and by algorithm MDP (see Proposition 3). Then, it examines all the neighborhood solutions in \( N_{\alpha}(J) \) of the current solution \( J \). If there is no improvement in \( N_{\alpha}(J) \), the local search procedure outputs the current solution \( J \) as a heuristic solution, and it terminates the computation process. Otherwise, it chooses a neighborhood solution \( J' \in N_{\alpha}(J) \) with the best improvement, and updates the current solution by \( J := J' \), and continues to exploit the neighborhood \( N_{\alpha}(J) \) of the updated current solution \( J \). As seen obviously, the proposed local search procedure follows the most simplest framework of local search technique which has been known well, e.g., see Ibaraki (1989).

5. Numerical Results

In this section, we conduct numerical experiments to examine the empirical performance of the proposed dynamic programming heuristic algorithm and that of the local search procedure. Instances of problem Preferential_PMS to be tested are randomly generated as follows.

- The number of machines (i.e., the number of delivery routings):
  \[ m = 100. \]

- The number of jobs (i.e., the number of cargoes):
  \[ n \in \{1000, 2500, 10000\}. \]

- An integer specifying the number of preferable machines for each job (i.e., the number of available lanes for each cargo):
  \[ L \in \{1, 2, \ldots, 10\}. \]

- The mean processing time:
  \[ \mu \in \{10, 50\}. \]

- The coefficient of variation of processing times:
  \[ \sigma/\mu \in \{0.1, 0.5\}, \] where \( \sigma \) denotes the standard deviation.

- A processing time of each job \( j \in N \) on a preferable machine:
  \[ p_j \in \text{uniformly random integers with respect to the mean value } \mu \text{ and the coefficient of variation } \sigma/\mu. \]

Further, we assume the time interval of any two consecutive delivery routings (see Section 2.3) as \( \Delta = 120 \), and generate the arrival time of each cargo \( j = 1, 2, \ldots, n \) in the uniform interval \([0, (m-1) \times \Delta]\) of random integers. Then, the first preferable machine of each job \( j \in N \) is given by \( s_j = \lfloor a_j/\Delta \rfloor + 1 \) (with checking the assumption of \( s_1 = 1 \) in Eq. (3)). The last preferable machine \( t_j \) of each job \( j \in N \) is given by the \( s_j \) and Eq. (2).

For a generated instance of problem Preferential_PMS, a lower bound on the minimum \( \ell \max \) of the maximum machine load can be provided by the average machine load over the \( m \) machines and the maximum of a processing time on a preferable machine, i.e.,
\[
\ell_{LB} = \max \left\{ \frac{\sum_{j \in N} p_j}{m}, \max_{1 \leq j \leq n} \{ p_j \} \right\},
\] (13)
and for a heuristic solution \( J \), we define the relative difference of the maximum machine load \( \ell \max(J) \) from the lower bound by
\[
\text{R.D.} = \frac{\ell \max(J) - \ell_{LB}}{\ell_{LB}},
\] (14)
which is an upper bound of the relative error from the optimal value.

The program is written in C language, and runs on a laptop personal computer with Windows 7 Home Premium (64bit) with SP1, Intel Core i5 2410 CPU (2.3 GHz) and 4GB memory. In all numerical results to be shown in the following tables, each of the data indicates the mean value in a set of twenty test instances.

Table 1 shows the relative difference of the maximum machine load obtained by algorithm MDP from the lower bound on the optimum $\ell_{\text{max}}$ (see Eqs. (13) and (14)). For the case of a single lane (i.e., $L = 1$), there is no choice for each job to select a lane. That is, any job must be assigned to the first preferable machine, and the assignment is optimal. Except for the case of $n = 1000$ and $L = 2$, the relative difference is less than 20 [\%]. In particular, for the cases of larger $n \in \{2500, 10000\}$ and $L \geq 3$, the relative difference is no more than 6 [\%] in the average. We also observe that the number of lanes with $L \geq 4$ seems to have a tiny effect on the improvement of the relative difference. For reference, the table also gives the relative difference of the maximum machine load obtained by FPM rule (see Eq. (4)) from the lower bound on the optimum $\ell_{\text{max}}$. The number of lanes is disregarded in FPM rule, and no effect of increasing the number of lanes is observed on the relative difference.

Table 2 indicates the maximum machine load obtained by algorithm MDP. For each number $n$ of jobs, the maximum machine loads are unchanged from $L = 6$ to $L = 10$. As in Table 1, the number of lanes with $L \geq 4$ seems to have a tiny effect on the improvement of the maximum machine load. Of course, a system characteristic may also relate to the observation. Hence, it would be interesting to examine the effect of increasing the number of available lanes on the improvement of the minimum (i.e., the optimum) of the maximum machine load. For reference, the table also provides the maximum machine load obtained by FPM rule. As mentioned above, the number of lanes is disregarded in FPM rule, and no effect of increasing the number of lanes is observed on the improvement of the maximum machine load.

Table 3 illustrates the relative difference of the maximum machine load obtained by the local search procedure from the lower bound on the optimum $\ell_{\text{max}}$. In this table, the local search procedure starting with the initial solution obtained by algorithm MDP is denoted by MDP+, while the local search procedure starting with the initial solution obtained by FPM is denoted by FPM+. The solution quality of the local search procedure obviously depends on the initial solution. We can...
have the same observation in Table 4, which shows the maximum machine load obtained by the local search procedure.

Local search algorithm MDP+ (i.e., the local search procedure with the proposed dynamic programming heuristic algorithm) obtains some improved maximum machine load with the relative difference from a lower bound no more than 5 [%] in the average for the test instances with \( L = 3 \) and \( n = 2500 \). On the other hand, we see that for most of the tested instances, the heuristic solutions obtained by algorithm MDP are returned as the outputs of the local search algorithm. The local search procedure improves the solution quality of FPM rule. However, a more improvement of the local search procedure itself would be desired in future research, e.g., by defining a more efficient neighborhood of the current solution.

| # Lanes, \( L \) | \# Jobs, \( n \) | 1000 | 2500 | 10000 |
|-----------------|----------------|------|------|-------|
|                 | MDP+ | FPM+ | MDP+ | FPM+ | MDP+ | FPM+ |
| 1               | 1.02  | 1.02  | 0.63 | 0.63 | 0.29 | 0.29 |
| 2               | 0.27  | 0.73  | 0.10 | 0.48 | 0.02 | 0.25 |
| 3               | 0.15  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |
| 4               | 0.13  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |
| 5               | 0.12  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |
| 6               | 0.12  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |
| 7               | 0.12  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |
| 8               | 0.12  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |
| 9               | 0.12  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |
| 10              | 0.12  | 0.72  | 0.05 | 0.48 | 0.01 | 0.25 |

\( m = 100, \, \mu = 10, \, \sigma/\mu = 0.5 \)

| # Lanes, \( L \) | \# Jobs, \( n \) | 1000 | 2500 | 10000 |
|-----------------|----------------|------|------|-------|
|                 | MDP+ | FPM+ | MDP+ | FPM+ | MDP+ | FPM+ |
| 1               | 202.6 | 202.6 | 408.3 | 408.3 | 1294.8 | 1294.8 |
| 2               | 127.0 | 173.6 | 274.7 | 370.7 | 1020.8 | 1252.3 |
| 3               | 115.8 | 172.8 | 264.2 | 369.5 | 1012.7 | 1252.3 |
| 4               | 113.2 | 172.8 | 262.1 | 369.5 | 1012.5 | 1252.3 |
| 5               | 112.5 | 172.8 | 261.9 | 369.5 | 1012.5 | 1252.3 |
| 6               | 112.1 | 172.8 | 261.9 | 369.5 | 1012.5 | 1252.3 |
| 7               | 112.1 | 172.8 | 261.9 | 369.5 | 1012.5 | 1252.3 |
| 8               | 112.1 | 172.8 | 261.9 | 369.5 | 1012.5 | 1252.3 |
| 9               | 112.1 | 172.8 | 261.9 | 369.5 | 1012.5 | 1252.3 |
| 10              | 112.1 | 172.8 | 261.9 | 369.5 | 1012.5 | 1252.3 |

\( m = 100, \, \mu = 10, \, \sigma/\mu = 0.5 \)

Table 5 shows the execution time of each local search algorithm, which involves the execution time to obtain an initial heuristic solution, of course. Also in this table, the local search procedure starting with the initial solution obtained by MDP is denoted by algorithm MDP+, while the local search procedure starting with the initial solution obtained by FPM is denoted by algorithm FPM+. As an observation, local search algorithm MDP+ spends most of the execution time to call the MDP. Even for larger \( n \in \{2500, 10000\} \) (with \( m = 100 \)), the execution time of local search algorithm MDP+ is less that 1 [sec]. As mentioned before, in the actual delivery workload management system, a neighborhood search procedure was adopted for the lane selection issue (see Fukuyama et al., 1995). Due to the short execution time as well as the solution quality, algorithms MDP and MDP+ are applicable to obtain a practical solution in the actual delivery workload management system. When the number \( L \) of available lanes becomes larger, the difference of \( v_i - u_i + 1 \) (which corresponds to the number of possible jobs assigned to machine \( i \in M \)) intends to be larger (see Eqs. (8) and (9)). Such a larger difference of \( v_i - u_i + 1 \) incurs a longer execution time of algorithm MDP in the second case of the recursive of Eq. (12).

Table 6 examines the effect of the mean processing time on the maximum machine load, and Table 7 does the effect of the coefficient of variation of processing times on the maximum machine load. In these tables, local search algorithm MDP+ is adopted to obtain a heuristic solution. In Table 6, we see that the maximum machine load is scaled by the mean processing time. In Table 7, we observe that although the mean processing time is fixed as \( \mu = 10 \), the larger coefficient \( \sigma/\mu = 0.5 \) incurs larger values of the maximum machine load than the smaller coefficient \( \sigma/\mu = 0.1 \). This
implies that in the practical situation, it would be effective to introduce a standardization of cargo constructing manner in order to level the processing times.

### Table 6 Effect of the Mean Processing Time on the Maximum of a Machine Load

| # Lanes, L | # Jobs, n | 1000 | 2500 | 10000 |
|-----------|-----------|------|------|-------|
|           |           | MDP+ | FPM+ | MDP+ | FPM+ |
| 1         |           | 202.6 | 1004.3 | 408.3 | 2020.8 |
| 2         |           | 127.0 | 626.9 | 274.7 | 1366.9 |
| 3         |           | 115.8 | 570.7 | 264.2 | 1314.9 |
| 4         |           | 113.2 | 562.7 | 262.1 | 1305.7 |
| 5         |           | 112.5 | 558.8 | 261.9 | 1304.9 |
| 6         |           | 112.1 | 556.9 | 261.9 | 1304.6 |
| 7         |           | 112.1 | 556.8 | 261.9 | 1304.6 |
| 8         |           | 112.1 | 556.8 | 261.9 | 1304.6 |
| 9         |           | 112.1 | 556.8 | 261.9 | 1304.6 |
| 10        |           | 112.1 | 556.8 | 261.9 | 1304.6 |

**MDP+, m = 100, \( \sigma/\mu = 0.5 \)**

### Table 7 Effect of the Coefficient of Variation on the Maximum of a Machine Load

| # Lanes, L | # Jobs, n | 1000 | 2500 | 10000 |
|-----------|-----------|------|------|-------|
|           | \( \sigma/\mu = 0.1 \) | \( \sigma/\mu = 0.5 \) | \( \sigma/\mu = 0.1 \) | \( \sigma/\mu = 0.5 \) |
| 1         | 193.8 | 202.6 | 383.0 | 408.3 | 1270.5 | 1294.8 |
| 2         | 120.5 | 127.0 | 269.0 | 274.7 | 1016.5 | 1020.8 |
| 3         | 112.4 | 115.8 | 260.1 | 264.2 | 1009.5 | 1012.7 |
| 4         | 110.4 | 113.2 | 259.1 | 262.1 | 1009.4 | 1012.5 |
| 5         | 110.0 | 112.5 | 258.9 | 261.9 | 1009.4 | 1012.5 |
| 6         | 109.9 | 112.1 | 258.9 | 261.9 | 1009.4 | 1012.5 |
| 7         | 109.9 | 112.1 | 258.9 | 261.9 | 1009.4 | 1012.5 |
| 8         | 109.9 | 112.1 | 258.9 | 261.9 | 1009.4 | 1012.5 |
| 9         | 109.9 | 112.1 | 258.9 | 261.9 | 1009.4 | 1012.5 |
| 10        | 109.9 | 112.1 | 258.9 | 261.9 | 1009.4 | 1012.5 |

**MDP+, m = 100, \( \mu = 10 \)**

### 6. Concluding Remarks

In this paper, we considered a delivery workload balancing problem in an assembly plant. We first described the problem as a special case of the unrelated parallel machine scheduling problem of minimizing the makespan, which includes the NP-hard identical parallel machine scheduling problem of minimizing the makespan. Then, we proposed an \( O(m^2n^2) \) time heuristic algorithm, where \( m \) is the number of machines and \( n \) is the number of jobs. The proposed heuristic algorithm is based on a dynamic programming procedure of linear partitioning. We also discussed an improvement...
We conducted numerical experiments to examine the empirical performance of the proposed heuristic algorithm, and the effect of the improvement procedure. The numerical results showed that for the test instances with three lanes, \( m = 100 \) and \( n \in [2500, 10000] \), the local search procedure with the proposed dynamic programming heuristic algorithm obtained the maximum machine load with the relative difference from a lower bound no more than 5 \( \% \) in the average, and ran fast on a laptop personal computer with the execution time less than 1 [sec]. For the larger test instances, the proposed dynamic programming heuristic algorithm itself (that is, without the local search procedure) obtained the maximum machine load with the relative difference from a lower bound no more than 6 \( \% \) in the average.

Of course, we are concerned for a more effective local search procedure for future research. On the other hand, the unrelated parallel machine scheduling problem of minimizing the makespan has been known to be polynomially approximable with a constant factor. Hence, it would be significant to design an approximation algorithm with a dedicated constant factor for the special case of the unrelated parallel machine scheduling problem of minimizing the makespan. It would also be significant to design an exact algorithm for the problem with a fixed small number of available lanes. This may make it possible to investigate some system characteristic by numerical manners. Moreover, we would like to remark that we considered a lane with a sufficiently large capacity in the cargo operating area. However, the cargo operating area has probably a limited space. It would be interesting from a practical viewpoint to regard a finite capacity of the lanes (either the number capacity of cargoes or the work amount capacity per delivery routing).

**Acknowledgment**

We would like to express our gratitude to two anonymous referees whose constructive suggestions contributed much to improving the organization of this paper. We also appreciate the Program Committee of International Symposium on Scheduling 2015 (ISS 2015, July 2015, Kobe, Japan), co-sponsored by Manufacturing Systems Devison of the Japan Society of Mechanical Engineers. We presented a part of the result in the Proceedings (Karuno and Furukawa, 2015). This refined version includes an enhanced heuristic procedure and a considerably extended numerical section.

**References**

Fukuyama, T., Asahara, T., and Tomimori, T., Internal conveyance system– Scheduling for parts delivery to the automobile assembly line (in Japanese), *Proceedings of 1995 Production Scheduling Symposium (Domestic)*, ISCIE, (1995), pp. 19–24.

Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, (1979), Freeman, San Francisco.

Graham, R.L., Bounds on multiprocessing timing anomalies, *SIAM Journal on Applied Mathematics*, Vol. 17, (1969), pp. 416-429.

Ibaraki, T., *Algorithms and Data Structures* (in Japanese), (1989), Shoko-do, Tokyo.

Karuno, Y. and Furukawa, K., Dynamic programming based heuristic algorithm for a delivery work leveling problem, *Proceedings of International Symposium on Scheduling 2015*, SSJ, (2015), pp. 46–51.

Lenstra, J.K., Shmoys, D.B., and Tardos, É., Approximation algorithms for scheduling unrelated parallel machines, *Mathematical Programming*, Vol. 46, (1990), pp. 259–271.

Pinedo, M., *Scheduling: Theory, Algorithms, and Systems*, (1995), Prentice Hall, New Jersey.

Skiena, S.S., *The Algorithm Design Manual*, (2008), Springer, London.

Vazirani, V.V., *Approximation Algorithms, 2nd Ed.*, (2001), Springer, Berlin.