Minimizing the Number of Workers in a Paced Mixed-Model Assembly Line

Xavier Delorme[a], Alexandre Dolgui[b,1], Sergey Kovalev[c], Mikhail Y. Kovalyov[d]

[a] *Ecole des Mines de Saint-Etienne, FAYOL-EMSE, CNRS UMR 6158 LIMOS, 158, cours Fauriel, 42023 Saint Etienne Cedex 2, France, E-mail: delorme@emse.fr*

[b] *IMT Atlantique, LS2N - UMR CNRS 6004, La Chantrerie, 4, rue Alfred Kastler - B.P. 20722, F-44307 Nantes Cedex 3, France, E-mail: alexandre.dolgui@imt-atlantique.fr*

[c] *INSEEC Business School, 25 rue de l'Université, 69007 Lyon, France, E-mail: skovalev@inseec.com*

[d] *United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus, E-mail: kovalyov_my@newman.bas-net.by*

Abstract

We study a problem of minimizing the maximum number of identical workers over all cycles of a paced assembly line comprised of m stations and executing n parts of k types. There are lower and upper bounds on the workforce requirements and the cycle time constraints. We show that this problem is equivalent to the same problem without the cycle time constraints and with fixed workforce requirements. We prove that the problem is NP-hard in the strong sense if $m = 3$ and if $m = 4$ and the workforce requirements are station independent, and present an Integer Linear Programming model, an enumeration algorithm and a dynamic programming algorithm. Polynomial in k and polynomial in n algorithms for special cases with two part types or two stations are also given. The relation to the Bottleneck Traveling Salesman Problem and its generalizations are discussed.

Keywords: workforce assignment, production line, computational complexity, algorithms

[1] Corresponding author

1 Introduction

We study the following workforce planning problem. There is a *paced assembly line* with m *stations* at which parts of distinct types of a set $P = \{1, \ldots, k\}$ are manufactured. Each part visits the stations in the same order $1, \ldots, m$ and fully occupies any station in the time period between its arrival and departure. With a time step C called the *cycle time*, a part at station i departs from this station and immediately arrives at station $i + 1$, $i = 1, \ldots, m - 1$, a new part arrives at station 1 and the part at station m leaves the line. Activities taking place between two successive part moves are called the *production cycle*. If part j is processed at station i in a production cycle, then the duration of all the operations on this part at this station in this cycle is a non-increasing function $p_{ij}(x_{ij})$ of the number x_{ij} of identical (fully skilled) workers assigned to station i and part j in this cycle. In the same cycle, workers assigned to different stations perform their operations in parallel. It is assumed that they do not move from one station to another in the same cycle. However, they can move instantly to another station after the cycle has finished. To be feasible with respect to the cycle time, the relation $p_{ij}(x_{ij}) \leq C$ must be satisfied for each station and each part in each production cycle. Besides, technological constraints define the lower and upper bounds on the values x_{ij} in each cycle: $0 \leq a_{ij} \leq x_{ij} \leq b_{ij}$, $i = 1, \ldots, m$, $j \in P$. We assume, without loss of generality, that $p_{ij}(b_{ij}) \leq C$ for all i and j.

A requirement of the global supply chain is that the production should keep a given proportion of the manufactured parts. To address this issue, the notion of a *Minimal Part Set (MPS)* is used, which was introduced into the operational research literature by Hitz [23]. Let the part proportion be given by the numbers $t_1, \ldots, t_k$, where t_j is the percentage of type j parts to be manufactured, and let GCD denote the greatest common divisor of the numbers $t_1, \ldots, t_k$. Denote $o_j = t_j/GCD$, $j \in P$, and $n = \sum_{j=1}^{k} o_j$. MPS is the multiset that consists of o_j copies of each type j part, $j \in P$. Note that, for a given part proportion $t_1, \ldots, t_k$, the numbers $o_1, \ldots, o_k$ and n are unique. In this paper, we represent MPS as the sequence $(o_1, \ldots, o_k)$. According to the theory of computational complexity, this representation satisfies the conditions of a reasonable encoding scheme (Garey and Johnson [18], p.10), while a representation that contains all n part copies does not satisfy these conditions.

An *MPS sequence* is a part sequence $\pi = (j_1, \ldots, j_n)$, $j_r \in P$, $r = 1, \ldots, n$, in which each

type j part occurs exactly o_j times, $j \in P$. According to the MPS sequence π, the parts visit the line cyclically in the order $j_1, \ldots, j_n, j_1, \ldots, j_n, \ldots$; see Fig. 1 for an illustration.

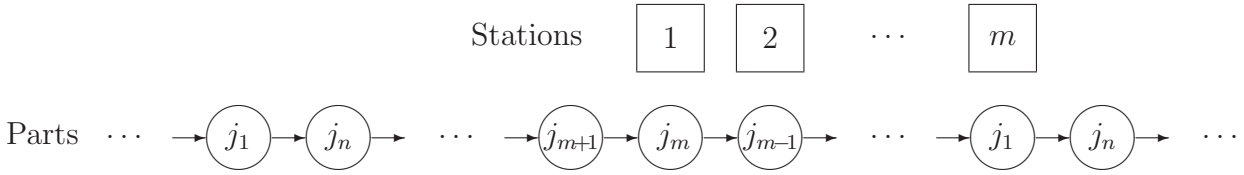

Figure 1: Parts moving according to an MPS sequence $\pi = (j_1, \ldots, j_n)$

In each production cycle, each station is assigned a part and this arrangement can be represented by a sequence $\sigma = (j'_1, \ldots, j'_m)$, where $j'_h \in P$ is the part assigned to station $m + 1 - h$, $h = 1, \ldots, m$. We call such an arrangement a *P-cycle*. For the example in Fig. 1, the current P-cycle is $(j_1, \ldots, j_m)$ and the previous P-cycle is $(j_n, j_1, \ldots, j_{m-1})$.

Given the MPS sequence $\pi = (j_1, \ldots, j_n)$, let us consider a *directed graph* $G(\pi)$ with the set of *vertices* $\{j_1, \ldots, j_n\}$ and the set of *arcs* $\{(j_1, j_2), (j_2, j_3), \ldots, (j_{n-1}, j_n), (j_n, j_1)\}$. Thus, $G(\pi)$ is the graph called the *(directed) cycle*. We say that a P-cycle σ is π*-feasible* if and only if it is a *walk* in the graph $G(\pi)$, with no vertex and arc repetition if $m \leq n$. For example, if $m = n+1$, then $\sigma = (j_1, \ldots, j_n, j_1)$ is π-feasible and, if $m = 3$, then $\sigma = (j_6, j_7, j_8)$ is π-feasible. Let $\Phi(\pi)$ denote the set of all distinct π-feasible P-cycles. We have $|\Phi(\pi)| = n$. Furthermore, let $x_i^{(\sigma)}$ denote the number of workers assigned to station i in the P-cycle σ, $i = 1, \ldots, m$. The total number of workers used in this P-cycle is equal to $\sum_{i=1}^{m} x_i^{(\sigma)}$. Denote by x the structure with the entries $x_i^{(\sigma)}$, $i = 1, \ldots, m$, $\sigma \in \Phi(\pi)$. The total number of workers used in the long run of the MPS sequence $\pi = (j_1, \ldots, j_n)$ is equal to $T(\pi, x) = \max_{\sigma \in \Phi(\pi)} \{\sum_{i=1}^{m} x_i^{(\sigma)}\}$. The problem is to determine an MPS sequence π and the number of workers assigned to each station in each distinct P-cycle, determined by the structure x, such that the maximum total number of workers over all P-cycles, $T(\pi, x)$, is minimized. We denote this problem as MinMaxSum. Note that an MPS sequence can be described in many different ways, and there exist special cases of the MinMaxSum problem for which an optimal MPS sequence is described in $O(k)$ time, see Section 7.

MinMaxSum problem is a subproblem of the industrial problem studied within European project "amePLM" (Battaïa et al. [5], Dolgui et al. [17]). In the industrial case, two or three automotive engines (two or three part types) are produced in the proportion of 75% and 25%

in the case of two engines and 75% and 20% and 5% in the case of three engines on an assembly line comprising of 11 stations. Therefore, $m = 11$, $k = 2$ or $k = 3$ and $(o_1, o_2) = (3, 1)$ or $(o_1, o_2, o_3) = (15, 4, 1)$. The processing time functions are station independent and inversely proportional to the number of workers: $p_{ij}(x) = p_j/x$, $i = 1, \ldots, m$, $j \in P$. Each station can be assigned one, two, three or four workers, that is, $a_{ij} = 1$ and $b_{ij} = 4$ for all i and j. The cycle time is determined as the ratio of the total production time available in the planning horizon to the planned production volume in this horizon.

In the next section, a review of the relevant literature is given. In Section 3, we demonstrate that the MinMaxSum problem is equivalent to the same problem without the cycle time constraints $p_{ij}(x_{ij}) \leq C$ and with $a_{ij} = b_{ij} = w_{ij}$ for appropriately determined fixed workforce requirements w_{ij}, $i = 1, \ldots, m$, $j \in P$, for which we keep the same notation MinMaxSum. We suggest an $O\big(\min\big\{mnk^{n-1}, \frac{n!m}{(o_1-1)!o_2!\cdots o_k!}\big\}\big)$ time enumeration algorithm to solve this problem, which can be efficient if n is small. In Section 4, we provide a proof of the strong NP-hardness of the MinMaxSum problem in the case when $m = 4$ and the workforce requirements are station independent: $w_{ij} = w_j$, $i = 1, \ldots, m$, $j \in P$. Obviously, if the workforce requirements are part type independent: $w_{ij} = w_i$, $i = 1, \ldots, m$, $j \in P$, then all MPS sequences π have the same total workforce requirements $T(\pi) = \sum_{i=1}^{m} w_i$. An $O(n^{k+1}k^{2m+2-k})$ time dynamic programming algorithm for the MinMaxSum problem with $n \geq m$ is presented in Section 5. Section 6 contains an Integer Linear Programming (ILP) formulation of this problem. Polynomial in k and polynomial in n algorithms for three special cases with two part types or two stations are given in Section 7. The paper ends with a summary of the results and suggestions for future research.

2 Literature review

The MinMaxSum problem has much in common with the problem studied by Lee and Vairaktarakis [29] and it is a special case of the problem studied by Winch et al. [45]. However, [29] and [45] do not consider concise representation of the MPS assuming that all parts are of different types ($k = n$) and the workforce requirements are fixed ($a_{ij} = b_{ij}$, $i = 1, \ldots, m$, $j \in P$). Furthermore, [29] assumes that the MPS sequence is not repeated cyclically: the line is free before the first part of an MPS sequence arrives at station 1 and the production stops when the last part of this sequence departs from station m.

We denote the problem in [29] as FIX. Lee and Vairaktarakis proved that this problem is solvable in $O(n \log n)$ time if $m = 2$ and it is NP-hard in the strong sense if $m = 3$. They suggested an ILP formulation, lower bounds and heuristics for the general case of FIX. Note that an $O(n \log n)$ algorithm would be pseudo-polynomial with respect to the MINMAXSUM problem. Vairaktarakis and Winch [40] extended the results of Lee and Vairaktarakis [29] to the case of partially skilled workers and the criterion of minimizing the costs of their cross-training. They motivated their studies by the production of firetrucks. Vairaktarakis et al. [41] extended the results of [29] to consider a different protocol of part processing, according to which each part requires a number of linearly ordered operations and each operation requires a specific station in a production cycle. The objective is to minimize a linear combination of the workforce cost and costs associated with the number of production cycles, which is one of the decision variables.

The workers in Winch et al. [45] are not identical: each worker possesses certain skills, stations are partitioned into skill groups and a worker can be assigned to a station only if he possesses the skill that determines the skill group of this station. The MINMAXSUM problem is a special case of the problem in [45], in which each worker possesses all skills. Winch et al. [45] proved that the MINMAXSUM problem is NP-hard in the strong sense for $m = 3$ and they stated that the $O(n \log n)$ algorithm in [29] can be adapted to solve the MINMAXSUM problem with $m = 2$ in $O(n \log n)$ time. They also designed lower bounds, heuristic algorithms and an optimal branch-and-bound procedure for the general problem.

Academic studies of the workforce planning problems for production lines were initiated by Akagi et al. [1] and Wilson [44]. Recent surveys of the results in this field are given by Battaïa and Dolgui [6], Battaïa et al. [5], De Bruecker et al. [16] and Dolgui et al. [17]. Further information can be found in Vairaktarakis et al. [42], Polat et al. [32], Dalle Mura and Dini [14], Yilmas and Yilmas [47], Zacharia and Nearchou [48] and Ritt et al. [34].

A close research area is cyclic scheduling and perfect periodic scheduling. In a cyclic scheduling problem, there is a set of jobs to be processed on a set of machines and each job must be repeated infinitely many times. A schedule must be found that repeats itself with a time step called the cycle time, satisfies a number of constraints and minimizes the cycle time. Levner et al. [30] provided a survey of the computational complexity results for cyclic scheduling problems. The papers of Bartholdi et al. [4], Dauscha et al. [15], Chretienne [13], Lutz and

Davis [31], Hanen and Munier [22], Kouvelis and Karabati [27], Hall et al. [21], Timkovsky [36], Hochbaum and Levin [24], Brucker and Kampmeyer [8], Kimbrel and Sviridenko [26], Sawik [35], Che et al. [11] and Bobrova and Servakh [7] contain additional results. A perfect periodic scheduling problem is similar, but there is a frequency request for each job and the objective is to find a schedule that minimizes deviations from the frequencies in each cycle. Perfect periodic scheduling problems have been studied by Bar-Noy et al. [3], among others.

The MinMaxSum problem, when formulated in graph theoretical terminology, can be called a MinMaxSum Traveling Salesman Problem (MinMaxSum TSP). In this problem, a Hamiltonian cycle in a complete directed graph has to be found such that the maximum sum of weights of m consecutive vertices is minimized. The class of MinMaxSum TSPs intersects with the Bottleneck TSP (Gilmore and Gomory [19], Burkard and Sandholzer [10], van der Veen [43], Burkard et al. [9], Vairaktarakis [37, 38], Kabadi and Punnen [25], LaRusic and Punnen [28]) and the TSP under Categorization (Punnen [33]), and it is also relevant to the k-Neighbor TSP (Woods et al. [46]) and the Maximum Scatter TSP (Arkin et al. [2], Chiang [12]).

3 Fixed workforce requirements and an enumeration algorithm

Consider an MPS sequence π. Observe that not all parts can be present in a P-cycle $\sigma \in \Phi(\pi)$, but, for any part, there is a P-cycle $\sigma \in \Phi(\pi)$ in which this part is present at any station. This observation implies two statements: 1) the cycle time constraint and the lower and upper bounds on the number of workers are satisfied if and only if for each pair (i, j), $i = 1, \ldots, m$, $j \in P$, there is a number of workers x such that $p_{ij}(x) \leq C$ and $a_{ij} \leq x \leq b_{ij}$, and 2) the MinMaxSum problem is equivalent to the problem, in which, for each pair (i, j), the number of workers is fixed to be equal to $w_{ij} := \min\{x \mid p_{ij}(x) \leq C, a_{ij} \leq x \leq b_{ij}\}$, $i = 1, \ldots, m$, $j \in P$. Recall that $p_{ij}(b_{ij}) \leq C$ is assumed for all i and j. If the functions $p_{ij}(x)$ are *invertible*, then $w_{ij} = \max\{a_{ij}, \lceil p_{ij}^{-1}(C) \rceil\}$, where $p_{ij}^{-1}(y)$ is the *inverse function* of $p_{ij}(x)$. If the functions $p_{ij}(x)$ are not invertible, then w_{ij} can be found in $O(\log_2(b_{ij} - a_{ij}))$ time by a bisection search over the range $[a_{ij}, b_{ij}]$. Thus, in $O(mk \log_2 \max_{i,j}\{b_{ij} - a_{ij}\})$ time, the MinMaxSum problem reduces to the equivalent problem without the cycle time constraints and with the fixed workforce requirements w_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, k$.

From now on, we assume that the MinMaxSum problem is to determine an MPS sequence π such that

$$T(\pi) = \max\Big\{F(\sigma) = \sum_{i=1}^{m} w_{m+1-i,j'_i} \mid \sigma = (j'_1, \ldots, j'_m) \in \Phi(\pi)\Big\}$$

is minimized. Let π^* denote an optimal solution of this problem.

We now describe an enumeration algorithm for the MinMaxSum problem. Assume, without loss of generality, that $o_1 = \min_{i=1,\ldots,k}\{o_i\}$. Denote by Π the set of all MPS sequences, in which a copy of part 1 is in the first place. Note that we can limit our search of π^* to these sequences because of the cyclic processing of parts. Let us evaluate the cardinality of this set. The number of sequences of length $n-1$ with copies of part 1 in $o_1 - 1$ places is equal to $C_{n-1}^{o_1-1} = \frac{(n-1)!}{(o_1-1)!(n-o_1)!}$. For each sequence with all o_1 copies of part 1 placed, there are $C_{n-o_1}^{o_2} = \frac{(n-o_1)!}{o_2!(n-o_1-o_2)!}$ sequences with copies of part 2 in o_2 places. By continuing this line of reasoning,

$$|\Pi| = C_{n-1}^{o_1-1} \cdot C_{n-o_1}^{o_2} \cdots C_{o_k}^{o_k} = \frac{(n-1)!}{(o_1-1)!o_2! \cdots o_k!}.$$

On the other hand, since each component of a $\pi \in \Pi$ can take one of the values $1, \ldots, k$, $|\Pi| \leq k^{n-1}$.

The MinMaxSum problem reduces to calculating $T(\pi^*) = \min\{T(\pi) \mid \pi \in \Pi\}$. Since $|\Phi(\pi)| = n$, each value $T(\pi)$ can be calculated in $O(mn)$ time. Hence, the MinMaxSum problem can be solved in $O(mn|\Pi|) = O\big(\min\big\{mnk^{n-1}, \frac{n!m}{(o_1-1)!o_2!\cdots o_k!}\big\}\big)$ time by calculating $T(\pi)$ for all MPS sequences $\pi \in \Pi$. For example, if $k = 2$ then it can be solved in $O\big(\min\big\{2^n, n^{o_1-1}\big\}mn\big)$ time.

4 NP-hardness

Winch et al. [45] prove that the MinMaxSum problem is NP-hard in the strong sense if $m = 3$. The next theorem considers the case of station-independent workforce requirements, as they are in the industrial case that motivates our studies.

Theorem 1 *The* MinMaxSum *problem is NP-hard in the strong sense if* $m = 4$ *and* $w_{ij} = w_j$ *for* $i = 1, \ldots, m$ *and* $j \in P$.

Proof: We will use a reduction from the NP-complete in the strong sense 3-Partition problem (Garey and Johnson [18]).

3-PARTITION: Given $3q+1$ positive integer numbers $e_1, \ldots, e_{3q}$ and E satisfying $E/4 < e_j < E/2$, $j = 1, \ldots, 3q$, and $\sum_{j=1}^{3q} e_j = qE$, is there a partition of the set $\{1, \ldots, 3q\}$ into subsets $X_1, \ldots, X_q$ such that $\sum_{j \in X_t} e_j = E$ for $t = 1, \ldots, q$?

Given an instance of the 3-PARTITION problem, we construct the following instance of the MINMAXSUM problem. There are $m = 4$ stations and $k = 3q + 2$ parts, among which there are $3q$ *partition parts* denoted $1, \ldots, 3q$ and two *enforcer* parts denoted the *E-part* and the *0-part*. The numerical parameters of the parts are: $o_j = 1$ and $w_j = e_j$ for the partition parts $j = 1, \ldots, 3q$, $o_E = q$ and $w_E = E$ for the E-part, and $o_0 = 6q$ and $w_0 = 0$ for the 0-part. We now show that there is an MPS sequence $\pi \in \Pi$ for this instance with a value $T(\pi) \leq E$ if and only if the original instance of the 3-PARTITION problem has a solution.

Part "only if". Let there be an MPS sequence $\pi \in \Pi$ with a value $T(\pi) \leq E$. Observe that in π each E-part must be immediately preceded by three 0-parts and immediately followed by three 0-parts, because otherwise there is a P-cycle in which an E-part and a partition part are simultaneously on the line and the total workforce requirement of this P-cycle is greater than E. Denote by $X_1, \ldots, X_q$ the sets of partition parts between two consecutive subsequences $(0, 0, 0, E, 0, 0, 0)$ in π, where 0 denotes a 0-part and E denotes an E-part. Observe that each set X_t consists of exactly three partition parts, because otherwise, by the Dirichlet principle, there is a set X_t that includes four partition parts and these contribute more than E to $T(\pi)$ when processed in the same P-cycle. In order to satisfy $T(\pi) \leq E$, we must have $\sum_{j \in X_t} w_j = \sum_{j \in X_t} e_j \leq E$, $i = 1, \ldots, q$. Since $\sum_{i=1}^{q} \sum_{j \in X_t} e_j = E$, we obtain $\sum_{j \in X_t} e_j = E$, $i = 1, \ldots, q$, as required for the proof of the part "only if".

Part "if". Let $X_1, \ldots, X_q$ be a solution of the 3-PARTITION problem. Define the MPS sequence $\pi = ([X_1], 0, 0, 0, E, 0, 0, 0, [X_2], \ldots, 0, 0, 0, E, 0, 0, 0, [X_q], 0, 0, 0, E, 0, 0, 0)$, where $[X_t]$ is an arbitrary sequence of the partition parts of the set X_t, $t = 1, \ldots, q$. Obviously, $T(\pi) = E$, as required for the proof of the part "if". ∎

Theorem 1 can be directly applied to prove that the FIX problem in Lee and Vairaktarakis [29] is NP-hard in the strong sense if $m = 4$ and the workforce requirements are station-independent. The computational complexity of the case with $m = 3$ and the station-independent workforce requirements is an open question for both the MINMAXSUM and FIX problems.

5 Dynamic programming algorithm for the case $n \geq m$

In this section, we assume that $n \geq m$. The MinMaxSum problem can be formulated as the following constrained bottleneck path problem. There is a vertex weighted acyclic directed graph $G = (V, A)$, in which the vertices are denoted as $v = (v_1, \ldots, v_m)$ and correspond to the same P-cycles $(v_1, \ldots, v_m)$, $v_i \in P$, $i = 1, \ldots, m$. There are n copies of each vertex. All the vertex copies are grouped into n same *layers*

$$L_r = \big\{v = (v_1, \ldots, v_m) \mid v_i \in P, i = 1, \ldots, m\big\},\ r = 1, \ldots, n.$$

We have $|L_r| \leq k^m$, $r = 1, \ldots, n$. The arcs of the set A are defined as follows. There is an arc (v, u) going from a vertex $v \in L_r$ to a vertex $u \in L_{r+1}$ if and only if $(v_2, v_3, \ldots, v_m) = (u_1, \ldots, u_{m-1})$, $1 \leq r \leq n-1$. The existence of the latter arc means that the P-cycle u can immediately follow the P-cycle v when the same MPS sequence rotates along the stations.

A vector $(o_1(v), \ldots, o_k(v))$ of *o-weights* is associated with each vertex v, whose components are equal to zero but the component $o_h(v) = 1$ if $v_m = h$, that is, if a copy of part type h is at the station 1 in the P-cycle $v = (v_1, \ldots, v_m)$. A *w-weight* $w(v) = \sum_{i=1}^{m} w_{i,v_{m+1-i}}$ is associated with each vertex v, which is the number of workers required for the P-cycle $v = (v_1, \ldots, v_m)$.

The MinMaxSum problem reduces to the following constrained bottleneck problem, denoted as CBP, of finding a *path* $\sigma = (v^{(1)}, \ldots, v^{(n)})$, $v^{(r)} \in L_r$, in the graph G such that

(i) $v^{(1)} = (v_1^{(1)}, \ldots, v_m^{(1)})$, $v^{(n)} = (v_1^{(n)}, v_1^{(1)}, \ldots, v_{m-1}^{(1)})$,

(ii) $\sum_{v \in \sigma} o_h(v) = o_h$, $h = 1, \ldots, k$, and

(iii) the maximum w-weight, $\max_{v \in \sigma} w(v)$, is minimized.

Consider an arbitrary MPS sequence $(j_1, \ldots, j_n)$. The requirement (i) states that if the first P-cycle of this sequence is $(j_1, \ldots, j_m)$, then the last (n-th) P-cycle of this sequence must be $(j_n, j_1, j_2, \ldots, j_{m-1})$. The requirement (ii) limits the search to the MPS sequences in which the number of occurrences of part type h is equal to o_h, $h \in P$. The requirement (iii) addresses the criterion of minimizing the number of workers over all P-cycles.

The problem can be solved by the following dynamic programming algorithm, denoted as DP-CBP. For each pair of vertices $v^{(1)}$ and $v^{(n)}$ that satisfy (i), apply a standard dynamic programming technique that associates the value of a recursive function $F_r(v^{(1)}, x_1, \ldots, x_k, u^{(r)}) =$

$\max_{v\in\sigma} w(v)$ with each (partial) path $\sigma = (v^{(1)}, u^{(2)}, u^{(3)}, \ldots, u^{(r)})$ from the vertex $v^{(1)}$ to a vertex $u^{(r)} \in L_r$, $r = 2, 3, \ldots, n$. The arguments of this function, called the *state variables* or *state*, are defined such that $x_h = \sum_{v\in\sigma} o_h(v)$ is the number of occurrences of the part type h in σ, $h = 1, \ldots, k$, and $u^{(r)}$ is the last vertex of the path. The function F_r represents the number of workers required for the first r number of P-cycles. It is clear that a partial path with the minimal value of $F_r(v^{(1)}, x_1, \ldots, x_k, u^{(r)})$ *dominates* all the other partial paths in the same state $(v^{(1)}, x_1, \ldots, x_k, u^{(r)})$ in the sense that if a path in this state can be extended to a path including all n number of P-cycles and corresponding to an optimal MPS sequence, then the dominant path can do the same.

The initialization is $F_1(v^{(1)}, a_1, \ldots, a_k, v^{(1)}) = w(v^{(1)})$, where a_h is the number of occurrences of the part type h, $h = 1, \ldots, k$, in the P-cycle $v^{(1)}$, and $F_r(v^{(1)}, x_1, \ldots, x_k, u) = \infty$ for $r = 2, 3, \ldots, n$ and all other states $(v^{(1)}, x_1, \ldots, x_k, u)$. The recursion for $x_h \in \{0, 1, \ldots, o_h\}$ and $u^{(r)} \in L_r$, $r = 2, 3, \ldots, n$, is

$$F_r(v^{(1)}, x_1, \ldots, x_k, u^{(r)}) =$$

$$\min_{u^{(r-1)}:\ u^{(r-1)}\in L_{r-1}, (u^{(r-1)}, u^{(r)})\in A} \max\left\{F_{r-1}(v^{(1)}, x_1 - o_1(u^{(r)}), \ldots, x_k - o_k(u^{(r)}), u^{(r-1)}), w(u^{(r)})\right\}.$$

For a given $v^{(1)}$ and $v^{(n)}$, the minimum number of workers is equal to $F_n(v^{(1)}, o_1, \ldots, o_k, v^{(n)})$ and the corresponding MPS sequence can be found by backtracking. An optimal solution of the MinMaxSum problem has the minimum value $F_n(v^{(1)}, o_1, \ldots, o_k, v^{(n)})$ over all the ordered pairs $(v^{(1)}, v^{(n)})$ satisfying (i).

The running time of the algorithm DP-CBP can be evaluated as $O(nT_1T_2T_3)$, where T_1 is the number of ordered pairs $(v^{(1)}, v^{(n)})$ satisfying (i), T_2 is the number of distinct tuples $(x_1, \ldots, x_k, u^{(r)})$, $x_h \in \{0, 1, \ldots, k\}$, $u^{(r)} \in L_r$, and T_3 is the number of vertices $u^{(r-1)}$ such that $u^{(r-1)} \in L_{r-1}$ and $(u^{(r-1)}, u^{(r)}) \in A$ for a given $u^{(r)} \in L_r$. We have $T_1 \le k^{m+1}$, $T_2 \le (\prod_{h=1}^{k} o_h) \max_{1\le r\le n}\{|L_r|\} \le (n/k)^k k^m = n^k k^{m-k}$ and $T_3 \le k$. Therefore, the running time of the algorithm DP-CBP is $O(n^{k+1}k^{2m+2-k})$. If m and k are fixed, then the algorithm DP-CBP is pseudo-polynomial, which contributes to the computational complexity classification of the MinMaxSum problem.

6 ILP formulation

It is convenient to use the *modulo operation* $\mod(g,n)$ such that $\mod(g,n) = b$ if $g = \lfloor a/n \rfloor n + b$ and g, n and b are positive integer numbers. Since the divisor is always n in this paper, we simplify the modulo operation notation to $\langle g \rangle$, $\langle g \rangle = \mod(g,n)$. The MinMaxSum problem can be formulated as the following integer linear programming problem denoted as ILP. Denote the set of non-negative integer numbers by Z_0. Introduce the non-negative integer variable T (total workforce requirement) and 0-1 variables y_{jp} such that $y_{jp} = 1$ if and only if a copy of part type p is in position j of an MPS sequence.

ILP **problem**:

$$\min T,$$

subject to

$$\sum_{p=1}^{k}(w_{1p}y_{\langle j \rangle p} + w_{2p}y_{\langle j+1 \rangle p} + \cdots + w_{mp}y_{\langle j+m-1 \rangle p}) \leq T,\ j = 1,\ldots,n, \tag{1}$$

$$\sum_{p=1}^{k} y_{jp} = 1, j = 1,\ldots,n, \tag{2}$$

$$\sum_{j=1}^{n} y_{jp} = o_p, p = 1,\ldots,k, \tag{3}$$

$$T \in Z_0,\ y_{jp} \in Z_0,\ j = 1,\ldots,n,\ p = 1,\ldots,k.$$

The left-hand side of the constraints (1) represents the total workforce requirement for each of the n distinct P-cycles of an MPS sequence. The relation in (1) for $j = 1$ addresses the case when station 1 is occupied by the first part copy of an MPS sequence, for $j = 2$ it addresses the case when station 1 is occupied by the second part copy of this sequence, and so on. Constraints (2) guarantee that each position of an MPS sequence is occupied by exactly one part copy. Constraints (3) require that the number of positions occupied by the copies of part type p is equal to o_p for each p. The ILP problem has $nk + 1$ variables and $2n + k$ constraints. Both dimensions are pseudo-polynomial because of $n = \sum_{j=1}^{k} o_j$. It is unknown if there exists an ILP formulation with the independent of n number of variables and constraints. The ILP problem is similar to that given by Lee and Vairaktarakis [29], but not the same.

7 Polynomially and pseudo-polynomially solvable cases

In sub-section 7.1, we suggest an $O(1)$ time algorithm for the case of two part types and station independent workforce requirements, i.e. $k = 2$ and $w_{i1} = w_1$, $w_{i2} = w_2$, $i = 1, \ldots, m$. In sub-section 7.2, we consider the case of $m = 2$. We show that this case reduces to a specific Bottleneck Traveling Salesman Problem (BTSP), for which $O(n^2)$ and $O(n \log n)$ time algorithms are known. Three sub-cases accept faster algorithms. If the workforce requirements are agreeable such that $w_{11} \geq \cdots \geq w_{1k}$ and $w_{21} \geq \cdots \geq w_{2k}$, then the corresponding BTSP, and hence MinMaxSum problem, can be solved in $O(n)$ time. If the workforce requirements are reversely agreeable such that $w_{11} \leq \cdots \leq w_{1k}$ and $w_{21} \geq \cdots \geq w_{2k}$, then the corresponding BTSP, and hence, MinMaxSum can be solved in $O(k)$ time. If the workforce requirements are station-independent such that $w_{1j} = w_{2j} = w_j$, $j = 1, \ldots, k$, then we present an $O(k \log k)$ time algorithm. We start with a proof of the following useful lemma.

Lemma 1 *If the workforce requirements are station-independent, then the* MinMaxSum *problem with $m > n$ reduces to the same problem with* $\mod(m, n)$ *stations.*

Proof: Let $m = rn + \delta$, where r and δ are non-negative integer numbers and $\delta \leq n - 1$. Thus, $\delta = \mod(m, n)$. We will show that this general case reduces to the case with $r = 0$. Assume that $r \geq 1$. Consider an arbitrary MPS sequence $(j_1, \ldots, j_n)$ and an arbitrary P-cycle, in which part copies $j_\delta, \ldots, j_1$ occupy stations $1, \ldots, \delta$, respectively, and r same sets of the n part copies $j_n, j_{n-1}, \ldots, j_1$ occupy rn stations $m - \delta, m - \delta + 1, \ldots, m$, respectively. Denote the multiset of the part copies assigned to the stations $m - \delta, m - \delta + 1, \ldots, m$ as M. In the next P-cycle, the part copy j_1 leaves the last station m and the same part copy j_1 arrives at the station $m - \delta$. The multiset M does not change. It is clear that it is the same in each of the n distinct P-cycles. Since the workforce requirements are station-independent, the contribution of the multiset M to the objective function is equal to $F_M = r \sum_{j=1}^{k} w_j o_j$. The remaining contribution is determined by the P-cycles for the first δ stations. Denote the minimum workforce requirement for the problem with δ stations as F_δ. Since the stations are identical, the minimum workforce requirement for the original problem with $rn + \delta$ stations is equal to $F_M + F_\delta$. ∎

In one of the industrial cases, $m = 11$, $k = 2$ and $n = o_1 + o_2 = 4$. Due to Lemma 1, this case reduces to the same case with $m = 3$.

7.1 Two part types

In this sub-section, we study the case in which $k = 2$, $w_{i1} = w_1$, $w_{i2} = w_2$, $i = 1, \ldots, m$, and $n \geq m$. It is the model for one of the industrial cases. If $w_1 = w_2$, then any MPS sequence is optimal. Assume, without loss of generality, that $w_1 > w_2$. To facilitate further discussion, we now present an integer linear programming formulation for this case, which is an adaptation of the ILP problem in Section 6. It is denoted as ILP$(k = 2, w_{ij} = w_j, n \geq m)$.

Introduce 0-1 variables z_j such that $z_j = 1$ ($z_j = 0$) if a copy of part type 1 (part type 2) is in position j of an MPS sequence. Denote $z = (z_1, \ldots, z_n)$. Re-call that $\langle g \rangle = \mod(g, n)$. With the variables z_j, the constraint in (1) for a given j transforms to

$$(w_1 z_{\langle j \rangle} + w_2(1 - z_{\langle j \rangle})) + (w_1 z_{\langle j+1 \rangle} + w_2(1 - z_{\langle j+1 \rangle})) + \cdots + (w_1 z_{\langle j+m-1 \rangle} + w_2(1 - z_{\langle j+m-1 \rangle})) \leq T,$$

which is equivalent to

$$A_j(z) := z_{\langle j \rangle} + z_{\langle j+1 \rangle} + \cdots + z_{\langle j+m-1 \rangle} \leq \frac{T - mw_2}{w_1 - w_2}. \tag{4}$$

ILP$(k = 2, w_{ij} = w_j, n \geq m)$ problem:

$$\min \quad T,$$

subject to

$$A_j(z) \leq \frac{T - mw_2}{w_1 - w_2}, \ j = 1, \ldots, n, \tag{5}$$

$$\sum_{j=1}^{n} z_j = o_1, \tag{6}$$

$$T \in Z_0, \ z_j \in \{0, 1\}, \ j = 1, \ldots, n.$$

It can easily be seen that the ILP$(k = 2, w_{ij} = w_j, n \geq m)$ problem is equivalent to minimizing $G(z) = \max_{j=1,\ldots,n}\{A_j(z)\}$, subject to (6) and the 0-1 constraints. Denote the optimal objective function value of this problem as G^*. Let us represent constraints (5) in the matrix form as $Az \leq v$. An example of the matrix A for $m = 3$ and $n = 9$ is given below.

$$A = \begin{array}{|ccccccccc|} \hline z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 & z_8 & z_9 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

Given a 0-1 vector z, we call index j *selected* if $z_j = 1$.

Theorem 2 *The vector z^*, in which the selected indices are $\lceil \frac{i \cdot n}{o_1} \rceil$, $i = 1, \ldots, o_1$, is an optimal solution of the* ILP$(k = 2, w_{ij} = w_j, n \geq m)$ *problem.*

Proof: Consider z^*. Let us evaluate the largest selected index among $1, \ldots, j+m-1$, which we denote as $x_{\max}$: $\frac{x_{\max} \cdot n}{o_1} \leq \lceil \frac{x_{\max} \cdot n}{o_1} \rceil \leq j+m-1$, from where it follows that $x_{\max} \leq \lfloor \frac{(j+m-1) \cdot o_1}{n} \rfloor := UB$. Similarly, let us evaluate the smallest selected index among $j, j+1, \ldots, n$, which we denote as $x_{\min}$: $j \leq \lceil \frac{x_{\min} \cdot n}{o_1} \rceil < \frac{x_{\min} \cdot n}{o_1} + 1$, from where it follows that $x_{\min} \geq \lfloor \frac{(j-1) \cdot o_1}{n} \rfloor + 1 := LB$. We deduce that in z^*, the maximum number of the selected indices among the m indices including and following index j does not exceed $UB - LB + 1 = \lfloor \frac{(j-1) \cdot o_1}{n} + \frac{m \cdot o_1}{n} \rfloor - \lfloor \frac{(j-1) \cdot o_1}{n} \rfloor$ for any j. It can be easily shown that $\lfloor a+b \rfloor \leq \lfloor a \rfloor + \lceil b \rceil$ for any non-negative numbers a and b. Therefore, $G(z^*) \leq UB - LB + 1 \leq \lceil \frac{m \cdot o_1}{n} \rceil$.

To prove the theorem, it is sufficient to show that $G^* \geq \lceil \frac{m \cdot o_1}{n} \rceil$. Assume that $G^* \leq \lceil \frac{m \cdot o_1}{n} \rceil - 1 < \frac{m \cdot o_1}{n}$, which means that there exists z such that $Az < v_0$ and $\sum_{j=1}^{n} z_j = o_1$, where v_0 is the transposed vector $(\frac{m \cdot o_1}{n}, \ldots, \frac{m \cdot o_1}{n})$. By summing the n relations $A_j(z) < \frac{m \cdot o_1}{n}$, we obtain $m \sum_{j=1}^{n} z_j < m o_1$, and since $\sum_{j=1}^{n} z_j = o_1$, we obtain $m o_1 < m o_1$, which is a contradiction. ∎

It follows from the above proof that $G(z^*) = \lceil \frac{m \cdot o_1}{n} \rceil$. Therefore, taking into account (4), the minimum number of workers is equal to $\lceil \frac{m \cdot o_1}{n} \rceil (w_1 - w_2) + m w_2$. We deduce that the ILP$(k = 2, w_{ij} = w_j, n \geq m)$ problem is solvable in $O(1)$ time. The MinMaxSum problem with $k = 2$ and $w_{i1} = w_1$, $w_{i2} = w_2$, $i = 1, \ldots, m$, can be solved in $O(1)$ time as well. If $m = rn + \delta$, $r \geq 0$, $0 \leq \delta \leq n-1$, then the optimal solution value is equal to

$$r\big(w_1 o_1 + w_2(n - o_1)\big) + \left\lceil \frac{\delta \cdot o_1}{n} \right\rceil (w_1 - w_2) + \delta w_2 = r(w_1 - w_2) o_1 + m w_2 + (w_1 - w_2) \left\lceil \frac{\delta \cdot o_1}{n} \right\rceil.$$

7.2 Two stations

In this sub-section, we assume that $m = 2$. The Bottleneck Traveling Salesman Problem (BTSP) can be formulated as follows. There is a complete directed graph with n vertices and arc costs c_{vu}. The problem is to find a Hamiltonian cycle such that the maximum arc cost in this cycle is minimized. Let the part copies be numbered $1, \ldots, n$. It is clear that the MinMaxSum problem with $m = 2$ is equivalent to the BTSP with the arc costs $c_{vu} = w^0_{2v} + w^0_{1u}$, where $w^0_{iv} = w_{ij}$, $i = 1, 2$, if the part copy v is of type j. We denote this problem as the BTSP-W.

Gilmore et al. [20] provided an $O(n^2)$ algorithm for the BTSP with costs such that $c_{vu} \geq c_{v+1,u}$ for all v and u. Re-number the part copies such that the workforce requirements at the second station satisfy $w^0_{2v} \geq w^0_{2,v+1}$, $v = 1, \ldots, n-1$. Then $c_{vu} = w^0_{2v} + w^0_{1u} \geq w^0_{2,v+1} + w^0_{1u} = c_{v+1,u}$ for all v and u. We deduce that the BTSP-W problem is solvable in $O(n^2)$ time by the algorithm in [20]. Furthermore, Vairaktarakis [37] developed an $O(n \log n)$ time algorithm for BTSP-W. The question of whether it can be solved in time polynomial in k is open.

We say that the workforce requirements are *agreeable* if they satisfy $w_{11} \geq \cdots \geq w_{1k}$ and $w_{21} \geq \cdots \geq w_{2k}$, and that they are *reversely agreeable* if they satisfy $w_{11} \leq \cdots \leq w_{1k}$ and $w_{21} \geq \cdots \geq w_{2k}$. We denote the BTSP-W with agreeable and reversely agreeable workforce requirements as the BTSP-WA and the BTSP-WRA, respectively.

van der Veen [43] developed an $O(n)$ time algorithm for the BTSP with costs satisfying $c_{vu} \geq c_{v+1,u}$ and $c_{vu} \geq c_{v,u+1}$ for all v and u. For the BTSP-WA, we have $c_{vu} = w^0_{2v} + w^0_{1u} \geq w^0_{2,v+1} + w^0_{1u} = c_{v+1,u}$ and $c_{vu} = w^0_{2v} + w^0_{1u} \geq w^0_{2v} + w^0_{1,u+1} = c_{v,u+1}$ for all v and u. Hence, the BTSP-WA can be solved in $O(n)$ time. The question of whether it can be solved in time polynomial in k is open.

Gilmore et al. [20] showed that $(1, \ldots, n)$ is an optimal solution of the BTSP if $c_{vu} \geq c_{v+1,u}$ and $c_{vu} \leq c_{v,u+1}$ for all v and u. For the BTSP-WRA, we have $c_{vu} = w^0_{2v} + w^0_{1u} \geq w^0_{2,v+1} + w^0_{1u} = c_{v+1,u}$ and $c_{vu} = w^0_{2v} + w^0_{1u} \leq w^0_{2v} + w^0_{1,u+1} = c_{v,u+1}$ for all v and u. Hence, $(1, \ldots, n)$ is an optimal solution of the BTSP-WRA. Observe that the sequence $(1, \ldots, n)$ can be represented using $O(k)$ memory units because parts of the same type appear consecutively in it. Therefore, the BTSP-WRA can be solved in $O(k)$ time.

Finally, consider the case of $m = 2$ and station-independent workforce requirements such that $w_{1j} = w_{2j} = w_j$, $j = 1, \ldots, k$. Note that this is a sub-case of the BTSP-WA, and therefore, it is solvable in $O(n)$ time. Let $w^0_v = w_j$ if the part copy v is of type j. Number the part copies in the *Least Workforce Requirement (LWR)* order such that $w^0_1 \leq \cdots \leq w^0_n$. We call the sequence of part copies $\pi = (1, n, 2, n-1, 3, n-2, 4, \ldots, \lfloor n/2 \rfloor, \lfloor n/2 \rfloor + 1)$ including all n copies the *Up-and-Down* sequence. In this sequence, the part copy $n - r + 2$ is inserted between the part copies $r - 1$ and r for $r = 2, 3, \ldots, \lfloor n/2 \rfloor$.

Theorem 3 *The Up-and-Down sequence is optimal for the* MinMaxSum *problem if* $m = 2$ *and* $w_{1j} = w_{2j} = w_j$, $j = 1, \ldots, k$.

Proof: Let W^* denote the optimal objective value for the MinMaxSum problem. We now

show that $W^* \geq w^0_{n-r+1} + w^0_{r+1}$ for $r = 1, \ldots, \lfloor n/2 \rfloor$, which is sufficient for the proof.

Since the part copy n with the largest workforce requirement appears on the line in two cycles with another part copy, $W^* \geq w^0_n + w^0_2$ must be satisfied. Assume that $W^* < w^0_{n-1} + w^0_3$. It follows that, in the optimal sequence, the part copy $n-1$ appears on the line with the part copy 1 in one cycle and with the part copy 2 in the other cycle. Then the part copy n appears in the same cycle with the part copy $i \geq 3$, and we obtain $W^* \geq w^0_n + w^0_3 \geq w^0_{n-1} + w^0_3$, a contradiction. Assume that $W^* < w^0_{n-2} + w^0_4$. It follows that, in the optimal sequence, the part copy $n-2$ appears on the line with a part copy from the set $\{1, 2, 3\}$ in one cycle and with another part copy from this set in the other cycle. Then the part copy n or the part copy $n-1$ appears in the same cycle with the part copy $i \geq 4$, and we obtain $W^* \geq w^0_{n-1} + w^0_4 \geq w^0_{n-2} + w^0_4$, a contradiction. Continuing in the same fashion, $W^* \geq w^0_{n-r+1} + w^0_{r+1}$ for $r = 1, \ldots, \lfloor n/2 \rfloor$, as required. ■

Observe that the Up-and-Down sequence can be represented using $O(k)$ memory units because copies of the same part appear consecutively in the LWR order. This concise representation of the Up-and-Down sequence can be constructed in $O(k \log k)$ time by sorting the parts into the non-decreasing order of their workforce requirements and employing this order in the description of the Up-and-Down sequence of part copies. Thus, both the MinMaxSum problem with $m = 2$ and $w_{ij} = w_j$ for $i = 1, 2$ and $j = 1, \ldots, k$, and the BTSP-W with costs $c_{vu} = w^0_v + w^0_u$ for all v and u can be solved in $O(k \log k)$ time.

Combining Lemma 1 and the latter result, we obtain the following corollary.

Corollary 1 *If $m = rn + 2$ and $w_{ij} = w_j$, $i = 1, \ldots, m$, $j = 1, \ldots, k$, then the* MinMaxSum *problem can be solved in $O(k \log k)$ time.*

8 Conclusions and suggestions for future research

The algorithmic and computational complexity results for the MinMaxSum problem are summarized in Table 1.

For future research, it would be interesting to establish computational complexity of the following important special cases:

- fixed number of part types, for example, $k = 2$ or $k = 3$, and variable m and n,
- $m = 3$ and $w_{ij} = w_j$ for all i and j,

Table 1: Computational complexity and algorithms for the MinMaxSum problem.

Additional characteristics	Complexity	Reference
General	$O\big(\min\big\{mnk^{n-1}, \frac{n!m}{(o_1-1)!o_2!\cdots o_k!}\big\}\big)$	Section 3
$n \geq m$	$O(n^{k+1}k^{2m+2-k})$	Section 5
$m = 3$	Strongly NP-hard	[45]
$m = 4$, $w_{ij} = w_j$	Strongly NP-hard	Theorem 1
$w_{ij} = w_i$	Any solution is optimal	Section 1
$k = 2$, $w_{i1} = w_1$, $w_{i2} = w_2$	$O(1)$	Section 7.1
$m = 2$	$O(n \log n)$	[37]
$m = 2$, agreeable w_{ij}	$O(n)$	Section 7.2 and [43]
$m = 2$, reversely agreeable w_{ij}	$O(k)$	Section 7.2 and [20]
$m = 2$, $w_{ij} = w_j$	$O(k \log k)$	Theorem 3
$m = rn + 2$, $w_{ij} = w_j$	$O(k \log k)$	Corollary 1

- $m = 2$ with respect to the ordinary NP-hardness or the existence of an algorithm polynomial in k,
- $m = 2$ and agreeable or proportional ($w_{2j} = aw_{1j}$ for $a > 0$) workforce requirements with respect to the ordinary NP-hardness or the existence of an algorithm polynomial in k.

To resolve open questions, links with the BTSP and MinMaxSum TSPs could be further investigated. The development of exact and approximate solution methods for the general MinMaxSum problem and computer experiments with them are both interesting and practically relevant topics.

References

[1] Akagi, F., Osaki, H., Kikuchi, S. (1983) A method for assembly line balancing with more than one worker in each station, International Journal of Production Research, 21(5), 755–770.

[2] Arkin, E.M., Chiang, Y., Mitchell, J.S.B., Skiena, S.S., Yang, T. (1999) On the maximum scatter traveling salesman problem, SIAM Journal on Computing, 29(2), 515-544.

[3] Bar-Noy, A., Dreizin, V., Patt-Shamir, B. (2004) Efficient algorithms for periodic scheduling, Computer Networks, 45, 155–173.

[4] Bartholdi III, J.J., Orlin, J.B., Ratliff, H.D. (1980) Cyclic scheduling via integer programs with circular ones, Operations Research, 28(5), 1074–1085.

[5] Battaïa, O., Delorme, X., Dolgui, A., Hagemann, J., Horlemann, A., Kovalev, S., Malyutin, S. (2015) Workforce minimization for a mixed-model assembly line in the automotive industry, International Journal of Production Economics, 170(B), 489–500.

[6] Battaïa, O., Dolgui, A. (2013) A taxonomy of line balancing problems and their solution approaches, International Journal of Production Economics, 142(2), 259–277.

[7] Bobrova, E.A., Servakh, V.V. (2017) Construction of cyclic schedules in presence of parallel machines, Journal of Applied and Industrial Mathematics, 11(1), 17–25.

[8] Brucker, P., Kampmeyer, T. (2008) A general model for cyclic machine scheduling problems, Discrete Applied Mathematics, 156(13), 2561–2572.

[9] Burkard, R.E., Deineko, V.G., Van Dal, R., Van der Veen, J.A.A., Woeginger, G.A. (1998) Well-solvable special cases of the traveling salesman problem: a survey, SIAM Review, 40(3), 496–546.

[10] Burkard, R.E., Sandholzer, W. (1991) Efficiently solvable special cases of bottleneck travelling salesman problems, Discrete Applied Mathematics 32(1), 61–76.

[11] Che, A., Kats, V., Levner, E. (2017) An efficient bicriteria algorithm for stable robotic flow shop scheduling, European Journal of Operational Research, 260(3), Pages 964–971.

[12] Chiang, Y.-J. (2005) New approximation results for the maximum scatter TSP, Algorithmica, 41(4) 309–341.

[13] Chretienne, P. (1991) The basic cyclic scheduling problem with deadlines, Discrete Applied Mathematics, 30, 109–123.

[14] Dalle Mura, M., Dini, G. (2016) Worker skills and equipment optimization in assembly line balancing by a genetic approach, Procedia CIRP, 44, 102–107.

[15] Dauscha, W., Modrow, H.D., Neumann, A. (1985) On cyclic sequence types for constructing cyclic schedule, Zeitschrift fur Operations Research, 29, 1–30.

[16] De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E. (2015) Workforce planning incorporating skills: state of the art, European Journal of Operational Research, 243(1), 1–16.

[17] Dolgui, A., Kovalev, S., Kovalyov, M.Y., Malyutin, S., Soukhal, A. (2018) Optimal workforce assignment to operations of a paced assembly line, European Journal of Operational Research, 264, 200–211.

[18] Garey, M.R., Johnson, D.S., 1979. Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco.

[19] Gilmore, P.C., Gomory, R.E. (1964) Sequencing a one state-variable machine: a solvable case of the traveling salesman problem, Operations Research, 12, 655–679.

[20] Gilmore, P.C., Lawler, E.L., Shmoys, D.B. Well-solved special cases, in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys, eds., The Traveling Salesman Problem (Wiley, Chichester, 1985) 87–143.

[21] Hall, N.G., Lee, T.E., Posner, M.E. (2002) The complexity of cyclic shop scheduling problems, Journal of Scheduling, 5(4), 307–327.

[22] Hanen, C., Munier, A. Cyclic scheduling on parallel processors: on overview. In P. Chretienne, E.G. Coffman, J.K. Lenstra, and Z. Liu, eds., Scheduling Theory and Its Applications, chapter 4 (John Wiley and Sons, New York, 1995) 194–226.

[23] Hitz, K.L. (1980) Scheduling of flow shops II. Report no. LIDS-R-879, Laboratory for Information and Decision Systems, MIT, Cambridge, Massachusetts.

[24] Hochbaum, D.S., Levin, A. (2006) Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms, Discrete Optimization, 3(4), 327–340.

[25] Kabadi, S.N., Punnen, A.P. The Bottleneck TSP. In G. Gutin and A.P. Punnen, eds., The Traveling Salesman Problem and Its Variations. Combinatorial Optimization, vol 12. (Springer, Boston, MA, 2007).

[26] Kimbrel, T., Sviridenko, M. (2008) High-multiplicity cyclic job shop scheduling, Operations Research Letters, 36(5), 574–578.

[27] Kouvelis, P., Karabati, S. (1999) Cyclic scheduling in synchronous production lines, IIE Transactions, 31(8), 709–719.

[28] LaRusic, J., Punnen, A.P. (2014) The asymmetric bottleneck traveling salesman problem: algorithms, complexity and empirical analysis, Computers and Operations Research, 43, 20-35.

[29] Lee, C.-Y., Vairaktarakis, G.L. (1997) Workforce planning in mixed model assembly systems, Operations Research, 45(4), 553–567.

[30] Levner, E., Kats, V., Alcaide, D., Cheng, T.C.E. (2010) Complexity of cyclic scheduling problems: a state-of-the-art survey, Computers and Industrial Engineering, 59, 352-361.

[31] Lutz, C.M., Davis, K.R. (1994) Development of operator assignment schedules: a DSS approach, Omega, 22(1), 57-67.

[32] Polat, O., Kalayci, C.B., Mutlu, Ö., Gupta, S.M. (2016) A two-phase variable neighbourhood search algorithm for assembly line worker assignment and balancing problem type-II: an industrial case study, International Journal of Production Research, 54(3), 722–741.

[33] Punnen, A.P. (1992) Traveling salesman problem under categorization, Operations Research Letters, 12, 89–95.

[34] Ritt, M., Costa, A.M., Miralles, C. (2016) The assembly line worker assignment and balancing problem with stochastic worker availability, International Journal of Production Research, 54(3), 907–922.

[35] Sawik, T. (2014) A mixed integer program for cyclic scheduling of flexible flow lines, Bulletin of the Polish Academy of Sciences Technical Sciences, 62(1), 121–128.

[36] Timkovsky, V.G. Cycle shop scheduling. In J.Y.-T. Leung, ed., Handbook of scheduling: algorithms, models, and performance analysis (Chapman & Hall/CRC, Boca Raton, London, 2004), 127–148.

[37] Vairaktarakis, G.L. (2003) Simple algorithms for Gilmore-Gomory's traveling salesman and related problems, Journal of Scheduling, 6, 499–520.

[38] Vairaktarakis, G.L. (2003) On Gilmore-Gomory's open question for the bottleneck TSP, Operations Research Letters, 31(6), 483–491.

[39] Vairaktarakis, G.L., Cai, X. (2003) Complexity of workforce scheduling in transfer lines, Journal of Global Optimization, 27(2-3), 273–291.

[40] Vairaktarakis, G.L., Winch, J.K. (1999) Worker cross-training in paced assembly lines, Manufacturing & Services Operations Management, 1(2), 1999, 112–131.

[41] Vairaktarakis, G.L., Cai, X., Lee C.-Y. (2002) Workforce planning in synchronous production systems, European Journal of Operational Research, 136 (2002) 551–572.

[42] Vairaktarakis, G.L., Szmerekovsky, J.G., Xu, J. (2016) Level workforce planning for multistage transfer lines, Naval Research Logistics, 63, 577–590.

[43] van der Veen, J.A.A. (1993) An $O(n)$ algorithm to solve the Bottleneck Traveling Salesman Problem restricted to ordered product matrices, Discrete Applied Mathematics, 47, 57–75.

[44] Wilson, J.M. (1986) Formulation of a problem involving assembly lines with multiple manning of work stations, International Journal of Production Research, 24(1), 59–63.

[45] Winch, J.K., Cai, X., Vairaktarakis, G.L. (2007) Cyclic job scheduling in paced assembly lines with cross-trained workers, International Journal of Production Research, 45(4), 803–828.

[46] Woods, B., Punnen, A., Stephen, T. (2015) Linear time algorithm for the 3-neighbour traveling salesman problem on Halin graphs and extensions, arXiv: 1504.02151v3 [cs.DM] 29 Apr 2015.

[47] Yilmaz, H., Yilmaz, M. (2016) A multi-manned assembly line balancing problem with classified teams: a new approach, Assembly Automation, 36(1), 51–59.

[48] Zacharia, P.T., Nearchou, A.C. (2016) A population-based algorithm for the bi-objective assembly line worker assignment and balancing problem, Engineering Applications of Artificial Intelligence, 49(C), 1–9.