Ann Oper Res
DOI 10.1007/s10479-016-2304-3

A cloud based job sequencing with sequence-dependent setup for sheet metal manufacturing

Yashar Ahmadov[1] · **Petri Helo**[1]

Abstract This paper presents a prototype system of sheet metal processing machinery which collects production order data, passes current information to cloud based centralized job scheduling for setup time reduction and updates the production calendar accordingly. A centralized cloud service can collect and analyse production order data for machines and suggest optimized schedules. This paper explores the application of sequencing algorithms in the sheet metal forming industry, which faces sequence-dependent changeover times on single machine systems. We analyse the effectiveness of using such algorithms in the reduction of total setup times. We describe alternative models: Clustering, Nearest Neighbourhood and Travelling Salesman Problem, and then apply them to real data obtained from a manufacturing company, as well as to randomly generated data sets. Based on the prototype implementation clustering algorithm was proposed for actual implementation. Sequence-dependency increases the complexity of the scheduling problems; thus, effective approaches are required to solve them. The algorithms proposed in this paper provide efficient solutions to these types of sequencing problems.

Keywords Big data analytics · Sequence-dependent setup · Heuristics · Job scheduling

1 Introduction

Machine building companies are looking for opportunities to utilize the data stored in smart machines and provide value added services for their customers (Dubey et al. 2016; Lee et al. 2013). Technology consisting of Internet of Things (IoT) and Big data analytics enables opportunities for improving the operational performance as machinery has become connected and linked to centralized cloud services (Porter and Heppelmann 2014; Lee et al. 2014; Liu

✉ Petri Helo
phelo@uva.fi

[1] Networked Value Systems, Department of Production, University of Vaasa, PO Box 700, 65101 Vaasa, Finland

Published online: 03 September 2016

et al. 2016). Several authors such as Zhong et al. (2016) and Tao et al. (2016) have proposed ways to integrate IoT and Cloud Manufacturing.

Increasing competitiveness in business is forcing companies to decrease their costs and improve operational performance. For manufacturing companies, an important component of costs is setup time, which they want to eliminate. Reducing the setup times leads to many improvements; increased productivity is one of them (Spence and Porteus 1987). Production planning and control is one of the approaches to reduce the actual time spent in these non-value adding activities. Traditionally, production planning and control activities such as job scheduling are conducted by plant management and coordinated then with manufacturing execution systems. Job scheduling algorithms aim to find the sequence of jobs among alternative routes that satisfies certain optimisation criteria. Total tardiness, total completion time, total setup costs, makespan and number of late jobs are some of the criteria used for optimisation (Reza Hejazi* and Saghafian 2005). In this paper, we discuss ways of minimizing the total tool changeover time for each workday.

We introduce a prototype system of sheet metal processing machine collecting data from production orders, feeding the information periodically to a centralized cloud based system which performs job scheduling algorithm with sequence-dependent tool changeover times and returns an improved job sequence. The effectiveness of using such algorithms in the reduction of total setup times has been analysed. We describe alternative models, Clustering (CL), Nearest Neighbourhood (NN) and Travelling Salesman Problem (TSP), and then apply them to real data obtained from a manufacturing company, as well as to randomly generated data sets.

In the next section we introduce the problem, and then review the previous work in sequence-dependent scheduling. Alternative models are presented and compared by using both real-data collected from machinery as well as randomly generated data. Finally, the results are compared and the paper is concluded.

2 Previous work

Sequence-dependent setup time reduction in a generic form has been known for a long time. The abridged form of the problem is FS–SD, where FS refers to Flow Shop and SD to Sequence Dependency. A setup can be needed after a job or batch; the convention FS–SD-job and FS–SD-batch are used to express these two situations. Other variations are also present such as JS (Job Shop), Hybrid Flow Shop (HFS), etc.

According to Burtseva et al. (2010), the total time that a product spends in a machine consists of three parts: setup, production and removal. In the past, companies used to ignore the setup and removal times, thinking that they are negligible. Pinedo (2008) showed that ignoring setup times can decrease the efficiency of the machines by more than 20 %. Also he proved that including setups in the scheduling makes the problem NP-hard, which is more complex to solve than a traditional approach.

Often the problem has been modelled as Travelling Salesman Problem (TSP); various algorithms, MILP linear programming and dynamic programming have been employed to solve the problem. Lockett and Muhlemann's (1972) paper is the article most related to our case. They discuss a scheduling problem with sequence-dependent changeover times. The authors divide the total changeover time into two: first-order serial and higher-order serial. First-order serial setup is caused by the previous job and higher-order setups by the jobs before the previous task. Problems with size of up to 35 jobs are solved using various heuristics. Namely, Random Ordering, Travelling Salesman without Backtracking, Travelling Salesman,

Closest Unvisited City algorithms and their performance were all tested. The results show that Travelling Salesman with no backtracking dominates the other methods Lockett and Muhlemann (1972). In their case, it is assumed that jobs require tools in specific stations, i.e. the order of tools in the turret are predefined. For example, assume that we have six tools in total and the turret has four stations. Further assume that a job requires tools (2, 4, 6, 8) in order to start the manufacturing process. It is not allowed to change the order of tools in the turret; the order should be exactly as (2, 4, 6, 8). But in our problem, a tool can be in any station as long as it fits into the station. Another efficient heuristic to solve large-scale TSP problems were proposed by Westphal and Noparlik (2014). They calculate a factor which is less than or equal to 5.875 for the TSP problems. This factor is then used to approximate the optimal solutions.

Gawroński (2012) discusses a sequence-dependent setup time reduction problem for the made-to-order furniture industry. His work is interesting, because the nature of the problem is same as the one discussed in this paper. The proposed algorithms reduce the setup times by 58–70 % when compared to the single shortest processing rule. Bowers et al. (1995) offer cluster analysis in order to minimise the sequence-dependent changeover times. Their approach is to group similar jobs, find an optimal sequence within groups and then find the optimal sequence among groups. Thus, near-optimal solutions are achieved; their calculations show that the gap between optimality and heuristic result is less than 5 %. This paper prompted us to decrease the computational effort. The number of possible sequences increases exponentially and an implicit enumeration becomes unfeasible. For example, for 15 jobs, the number of possible combinations is expressed in terms of 10^{11}. The computational difficulties in sequencing problems have been researched by Rudek (2012) by focusing on the problems with position dependent job processing times. He has proved that these types of problems which are aiming the minimization of the maximum lateness with release dates are NP-hard problems.

One of the prominent works in this field belongs to Nonaka et al. (2012). The authors discuss scheduling with alternative routings in CNC workshops. Although sequence-dependency has not been directly taken into account, the paper gives novel ideas about sequencing in similar work environments. Process planning and production scheduling are integrated to increase efficiency. They combine mathematical optimisation and tabu search for finding the optimal route and assigning the operations to machines, respectively. The problem is similar to our case in the sense that there are (x!) number of different routing alternatives for x jobs. They tackle the computational difficulty by using the column generation method. The column generation method is an algorithm used for solving huge linear programming models and it has been used successfully in many cases.

Hwang and Sun (1998) discuss the sequencing problem with re-entrant work flows and sequence-dependent setup times. Re-entrant work flow means that jobs are performed in a specific machine more than one time during the sequence of the manufacturing process. The scheduling problem consists of n jobs that are to be produced in two machines and the objective is minimization of the makespan. The authors employ modified dynamic programming to solve the problem and find the optimal sequence. Dynamic Programming has also been used by Giglio (2015) to solve sequence dependent scheduling problems where the machine is unreliable.

An interesting approach by White and Wilson (1977) is worth mentioning. They employ a regression model to find out and classify the significant factors that affect setups for each machine. 93 setups have been collected by setup personnel for this purpose. Then the regression model is used to predict the setup times for sequence-dependent operations. The regression model unveils hidden characteristics of the setup operations, which is important for sequencing. The next stage is the sequencing of tasks using predicted changeover

time values for which different optimisation tools and heurists were employed. The authors model the problem as a Travelling Salesman problem (TSP) with no requirement to return to origin, which has NxN cost (or distance, time) matrix. The advantage of the solution heuristics offered in the paper is that they are easy and do not take much time to solve.

Gupta (1982) offers the Branch and Bound method to solve scheduling problems with sequence-dependent setup for n jobs and m machines. His objective function was minimizing the total setup cost. But again, he assumes that the setup time of switching from job A to B is the same as switching from B to A, which is not a valid assumption in our case. One other drawback of his algorithm is that it is limited to small problems, i.e. it is not efficient in solving large problems. Cheung and Zhou (2001) have combined the Genetic Algorithm and heuristic methods to solve a similar problem. Similar approach has been employed by Kalayci et al. (2014) in the context of optimization of the reverse logistics. Mirabi (2011) has proposed a modified ant colony optimisation (ACO) algorithm to solve the sequence-dependent setup problems with the objective of minimizing the total makespan. His findings show that the new algorithm performs better than the regular ACO algorithm.

3 Problem description

Setup operation is generally defined as changing the manufacturing status from producing one job to another (Zandin 2001, 95). There can be different kinds of setups such as material, tool, and operator setups. 'Tool changeover' is a self-explanatory term which is one component of setup, i.e. the operation of changing tools in order to start the manufacturing of a specific product. In this paper, we use these terms interchangeably; both referring to the tool changeover times. 'Sequence-dependent changeover' means that the time spent for changeover in the previous step affects the time for the current stage. Sometimes it is not only the previous step affecting the setup time, but the whole sequence of jobs preceding the current step which are determining the changeover time.

Sheet metal forming is a good example of industry facing sequence-dependent changeover times. The main challenge in solving this type of problem is the computational burden. The payoff between acceptable solution and solving time is an important issue to be decided (Nonaka et al. 2012). Finding the optimal solution may take such a long time that utilizing such methods may not be feasible in practice. To overcome this difficulty, different heuristic rules have been proposed by a number of authors. These algorithms provide acceptable solutions within a reasonable solving time. The number of jobs that we will be dealing with is on average 10–15 per day, but it is also assumed that the problem size can be as large as 30 jobs per day. Customers are willing to wait 1–2 min for the solution; in case the problem size is big, this can take 3–4 min at most. These are the main restrictions that we will keep in mind while searching for solutions to the problem.

In this paper we will be incorporating so-called 'accepted tool change', different tool and station sizes into our models. These issues change the approach to the solution methods. To reduce the computational difficulties, we apply Clustering, Nearest Neighbour and Travelling Salesman methods which give near-optimal results within reasonable solving time. In this problem formulation, we want to optimise the sequencing of jobs, which will minimise the total time spent on tool changeovers. That is the purpose of this work: optimizing the machine task list by finding the optimal (or near-optimal) arrangement of jobs for each machine. (Fig. 1).

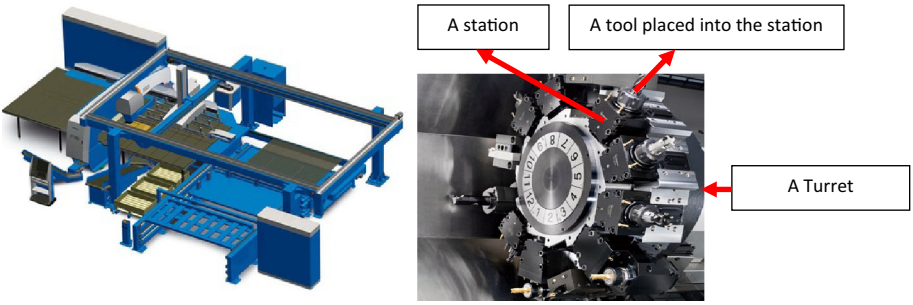

Fig. 1 A sample combi machine for shearing, punching and laser cut of sheet metal with tool turret

Table 1 Tool Requirements for a sample job

Station	Tool	Load angle	Hits	Die	Size
2	RD7.0	0	4660	0.30	B
7	SQ20.0	0	320	0.20	B i
11	RD20.0	0	180	0.20	D* i
21	SQ5.0	0	3360	0.20	MT6-A*
22	RD2.0	0	860	0.20	MT6-A*
24	RD6.0	0	340	0.20	MT6-A*

The five main elements involved in the problem are:

(1) *Machines*. Different types of punching and shearing, punching and laser cutting machines.
(2) *Turret*. A round-shaped structure that holds the tools.
(3) *Stations*. A 'nest' in the turret where the tools are installed.
(4) *Tools*. Different types of metal structures that give the specified shape to the products.
(5) *Parts (orders)*. Finished sheet metals.

After the general production plan is made, nesting is done for all tasks. The objective of the nesting is to decide which parts will be produced from one sheet metal piece by reducing the waste of raw materials (Rao et al. 2007, 439). As a result of nesting, the required tools, load angles, hits and die clearances are set. In order to manufacture a product, all the necessary tools should be loaded into the turret. Table 1. shows an example of such information for a sample task. The column station is the tool station identification number in the turret. The stations #1–20 are normal stations, #21 and greater than #21 are multi-tool stations; Tool column refers to the name and shape of the tool that is assigned to the station; Load angle refers to the initial angle of the tool; hits refers to The number of punching hits processed with each tool. Die refers to the value of the die clearance; and size to the size of the station ('i' stands for indexable).

3.1 Problem formulation

At the beginning, it is important to clarify the inputs and outputs of the problem. The inputs (of the problem) are:

a. Jobs, their tools, angle clearance requirements
b. Turret, its capacity, station sizes, initial conditions
c. All tool numbers, their sizes
d. Average time spent on tool, clearance, angle changes and adapter plugging

And the expected outputs are:

a. Optimal sequencing of jobs
b. What tools to change at which stage, where to install them in the turret (this output is optional)

The objective and constraints of the problem are given below:

Objective: Minimise total tool changeover time = (1) tool change time + (2) adapter plugging time + (3) die clearance change time + (4) angle change time

Subject to the following constraints/requirements:

- Each job has its own tool requirements. These tools should be in the turret while processing the order.
- Turret's station sizes define what tools can be fitted to them.
- Tools are also of different sizes.
- Changing tools in the turret *takes some time* (1).
- A tool can be fitted to a turret station with larger size. This needs an adapter, which *takes some time* (2) to attach to the tool.
- Clearance values (material thicknesses) should be taken into account. The tools should have appropriate die clearance; if any tool has the wrong clearance, it should be corrected. This change *takes some time* (3).
- Each task has its specific initial angle requirements, i.e., tools should be at pre-defined angles when the manufacturing process starts. Any angle change *takes time* (4).

3.2 Time calculation and tool setup

This section discusses an example of how the setup procedure is conducted in the SMF industry. Let us assume that when we start the workday tools number 1 and 4 are already available in the turret. The initial configuration is visualized in Fig. 2. Different sizes of circles represent the stations with different sizes.

During the workday, the machine has to complete seven jobs and each job requires tools with corresponding angle and clearance values as given in Table 2. The sizes of the vectors in the 2nd, 3rd and 4th columns should be same because each tool has a load angle and clearance value. The total time spent for setups is divided into four parts:

1. Time spent on tool change
2. Time spent on changing the die clearance
3. Time spent on changing the angle of the tool
4. Time spent on plugging in the adapter

Here we assumed that the tool change time is 5 min/tool, the die clearance change time is 2 min/tool, the angle change takes 1 min/tool and the adapter plugging time is 3 min/tool.

Now let us suppose that we are going to process the jobs in the following order: 2–4–1–6–3–5. This is the optimal sequence which will be explained in the upcoming sections. The following drawings (Fig. 3) exhibit the evolution of the turret during the process:

Now what happen in detail are the changes in the turret when we switch from stage 1 to stage 2.

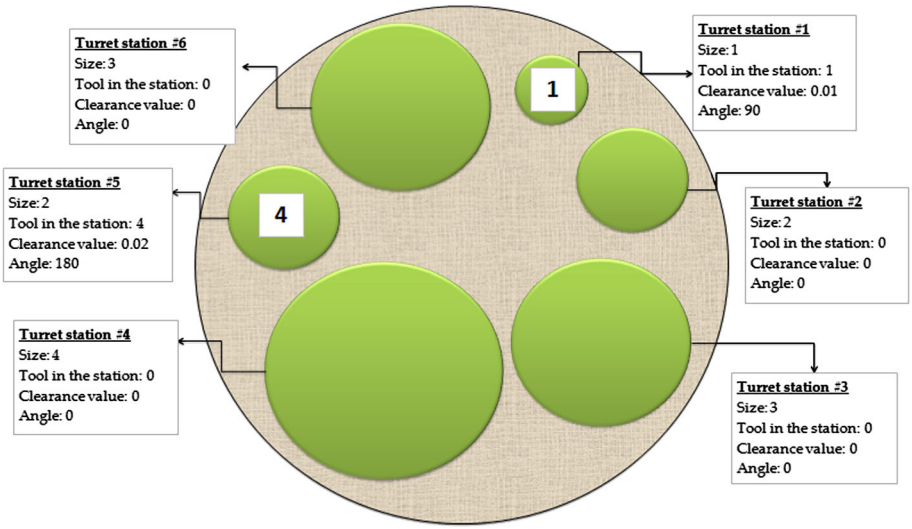

Fig. 2 A sample turret

Table 2 Tool, angle and die clearance requirements for the sample job

Job #	Tool requirements	Tool angles	Tool clearances
1	[1 2 3 5 6 7]	[90 0 0 180 270 0]	[0.01 0.05 0.02 0.08 0.03 0.04]
2	[2 3 4 8]	[0 90 180 270]	[0.02 0.04 0.05 0.03]
3	[1 2 5]	[0 90 0]	[0.02 0.07 0.02]
4	[1 2 3 5 6 7]	[90 0 0 270 180 360]	[0.01 0.02 0.03 0.04 0.03 0.02]
5	[1 2 5 8]	[0 90 0 180]	[0.02 0.08 0.09 0.01]
6	[1 3 6]	[270 90 180]	[0.02 0.08 0.09]

(a) Tools 1, 3 and 2 stay in the turret, whilst tools 5, 6 and 7 are added. This means that there are three tool changes; considering that installing each tool takes 5 min, then $3 \times 5 = 15$ min is spent on these installations.
(b) As noted above, tools 1, 3 and 2 stay in the turret when switching from stage 1 to 2. But in stage 2, tool 3's clearance value is different from stage 1 (0.04 and 0.03, respectively), that is why a die clearance change is needed. $1 \times 2 = 2$ min will be spent on this process.
(c) The same is valid for the angles. Again, there is a change in the load angle of tool 3; it should be rotated from 90° to 0°. $1 \times 1 = 1$ min is needed for this change.
(d) The last component of tool setups happens when a tool with smaller size is installed to a larger station. In our example case, tool number 7 is installed in station number 6. But the tool is smaller than the size of the station and we need to use an adapter while placing the tool. This takes $1 \times 3 = 3$ min.

Adding up all these four numbers, we get 21 min (15+2+1+3). This is the cost of processing job #4 after job #2. Our aim is to find the sequence that result in the minimum time being spent on setups. The time calculation considers all possible combinations of jobs, calculates the setup times for each of them and gives the optimal sequencing. Figure 4 depicts the Gantt chart for the optimal solution of our example case.

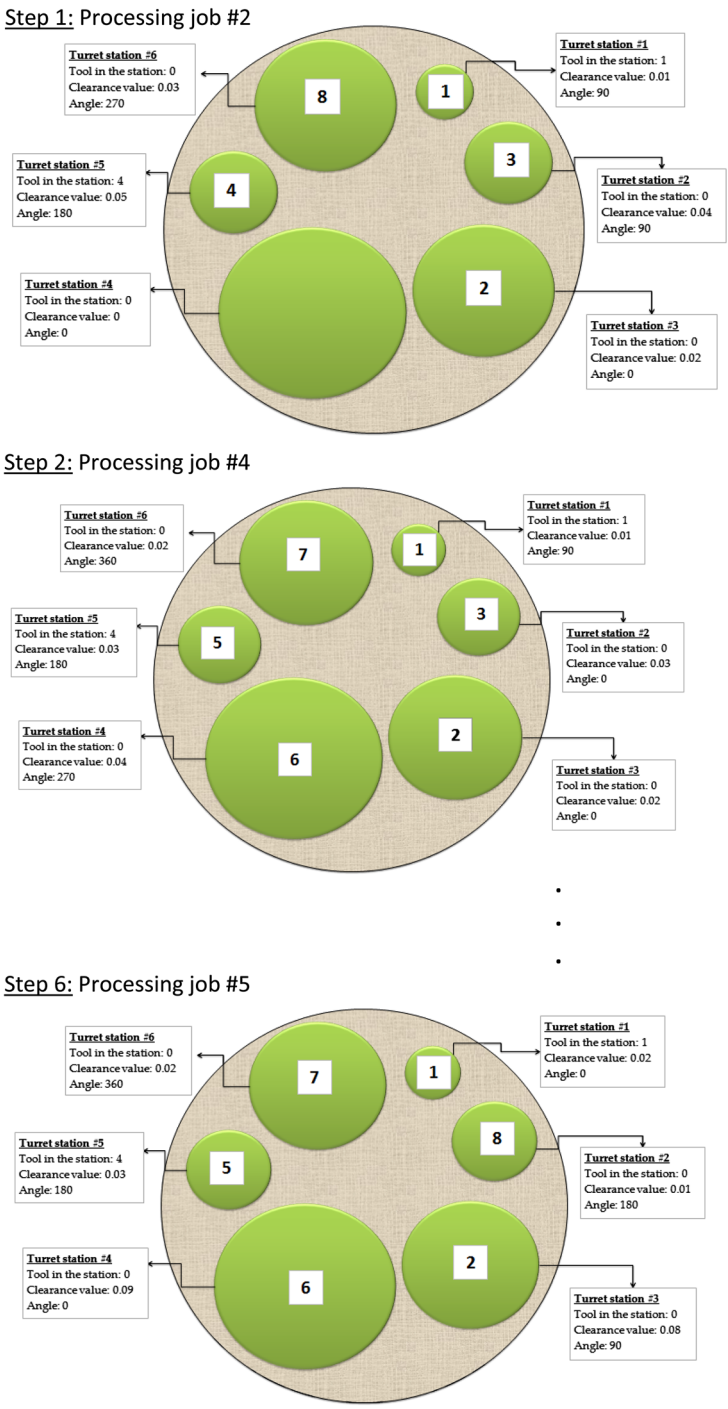

Fig. 3 Changes in the turret

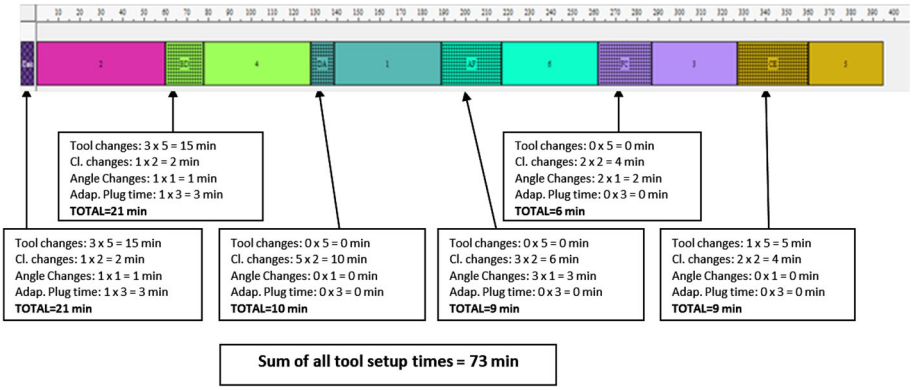

Fig. 4 Gantt chart of the production process

The distribution of the different components of the tool changeovers was as follows: tool changes—45 %, die clearance changes—38 %, angle changes—9 %, and adapter installing time—8 %. As can be seen from the chart, pure tool changes take only half of the total time. The rest of the time is devoted to angle, clearance and die changes. This work is also important from this aspect; we also take into consideration other factors that affect the total tool changeover time.

3.3 Work sequence and tool dependencies

Now that the problem is modelled, we can start searching for the solution. The first function programmed gives the total changeover time for a sequence given as input. Since finding the optimal point in such combinatorial problems is practically unfeasible from a solution time point of view, we need to use some heuristics. Before elaborating on advanced methods, we can visualise the jobs versus the tools required to have some initial ideas.

As can be seen from Fig. 5, some jobs use the same tools, others do not. For example, jobs 6 and 7, 11 and 12 require totally the same tools, so, these jobs can be processed sequentially. If we start with job number 15, for the next jobs we will have to change the tools only a few times because it contains the majority of the tools needed for processing all the jobs. This data package consists of 19 jobs, which is far above the average, but for data sets with less than 6–7 jobs it is fairly easy to guess.

4 Comparison

Three potential methods are compared for solving the tool changeover problem. These are: Nearest Neighbour, Nearest Neighbour Clustering, and Travelling Salesman Problem. Each of the approach can provide an improved solution, but based on data analysis the actual implementation will be suggested.

4.1 Solution algorithms

Based on the literature we came up with three different approaches to the solution of our case problem. First, the pseudo codes were prepared and then were programmed in the actual implementation. The following signs and functions are used in the pseudo-codes:

Jobs\Tools	2	3	4	5	6	7	8	9	12	13	14	15	16	17	19	22	24	25	27	28
1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1
2	0	0	0	0	0	1	1	1	0	1	0	0	1	0	0	0	1	0	0	1
3	1	0	0	0	0	1	1	1	0	1	0	0	1	0	0	0	1	0	0	1
4	1	1	1	0	0	1	1	1	0	1	0	0	1	1	0	0	1	0	1	1
5	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1	0	1	1
6	1	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	1	0	1	1
7	1	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	1	0	1	1
8	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	0	1
9	0	0	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1	0	1	1
10	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	0	1	1
11	1	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1	0	1	1
12	1	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1	0	1	1
13	1	0	1	0	0	1	1	1	0	1	0	0	1	1	0	0	1	0	0	1
14	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	0	1	0	1	1
15	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	1
16	1	0	0	0	0	1	1	1	0	1	1	0	1	0	0	0	1	0	1	1
17	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
18	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1
19	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0	1

Fig. 5 Jobs vs tool requirements

**length(a): The length of array a.*
**horzcat(a, b): a and b.*
**a\b: The set difference of a and b.*
** ∅: Empty set.*
**min(x): The smallest element of the vector x.*
**last(a): The last element of the vector a.*

4.2 Nearest neighbour algorithm

Nearest Neighbour (NN) algorithm is one of the earliest methods for solving several problems in operations research, including the TSP-model problems. We start with one data point, then find the Nearest Neighbour and continue this process until all points have been covered. This algorithm does not provide an optimal solution, but the main advantage is that it is much faster compared to the optimisation methods. If the solution is within an acceptable range, then NN algorithm can be preferred due to its speed. The pseudo code for this algorithm can be described as follows:

1. Pick one data point, X
2. Find the nearest unvisited data point, Y
3. Set current point to Y
4. Mark Y as visited
5. Stop if all data points have been visited, otherwise, go to step 2.

At this point, the important question is how the 'nearest' distance is measured. There are different measures proposed in the literature about this issue. For our problem, there is one criterion, i.e. the distance is one dimensional. We are interested in only the total setup time for a set of tasks and the 'nearness' will be measured according to the setup time between operations.

In this code, lines 0 and 1 serve for taking inputs and initialization. Parts 2, 3 and 4 calculate the number of tool changes and the time for switching from the current job to all of the unprocessed jobs. For example, if we are now manufacturing the first product, that part of the code will calculate the switching cost from 1 to 2, from 1 to 3, from 1 to 4, and so on. The fifth line chooses the minimum of those costs. Let us assume that switching from job #1 to #3 takes the lowest time, then the vector *visited* will be {1, 3}, and *unvisited* will be {2, 4, 5, . . .}. This loop continues until *unvisited* becomes an empty set. The final order of set elements *visited* is the solution of the Nearest Neighbour algorithm.

0: Take inputs

1: Set current point = 1, visited = {1}, unvisited = {2,3,…n}

2: ***for*** *q=1: length(unvisited)*

Calculate the following values for the [current_point, unvisited(q)]

- $\#\ of\ tool\ changes$
- $\#\ of\ clearance\ changes$
- $\#\ of\ angle\ changes$
- $\#\ of\ adapter\ pluggings$

3: ***end for***

4: Calculate total time for each sequence q:

$$
\begin{aligned}
Total\_time(q) \\
&= \ \#\ of\ tool\ changes(q) * tool\_change\_time \\
&+ \#\ of\ clearance\ changes(q) * clearance\_change\_time \\
&+ \#\ of\ angle\ changes(q) * angle\_change\_time \\
&+ \#\ of\ adapter\ plugging(q) * adap\_plug\_time
\end{aligned}
$$

5: Find the next point x that has min (total_time)

6: Set current_point = x, visited=horzcat(visited,x) , unvisited=unvisited\X

7: If unvisited= Ø, stop; the solution is the vector 'visited'. Otherwise, go to step 2.

4.3 Nearest neighbour clustering

Since the implementation of the Complete Enumeration method for sample sizes larger than six is practically unfeasible, a heuristic was developed that combines clustering and CE. Many authors have used this approach in the literature; for example, von Luxburg et al. (1981) have explained the applications of such algorithms for discrete optimisation problems. Clustering is grouping data points that share certain characteristics and there are different types and ways of Clustering. Some Clustering approaches allow a data point to be included in several clusters; others do not. Some Clustering approaches use probability, i.e. the data points belong to some group with certain probability (Witten and Frank 2005).

k-Nearest Neighbours says that a data point belongs to the same group with its k closest points. In simple terms it means '*do what your neighbours do*' (Adriaans and Zantinge 1996, 56–57). Again, the 'closeness' can be defined in many ways; in our case this is the time spent for tool changeovers. The algorithm works as follows:

1. Find the 5 closest points to the current point.
2. Optimise the sequence for those 5 jobs.

3. Mark the last point as the current point.
4. If all points are covered, then stop. Otherwise, go to step 1.

The first part of this algorithm calculates the distance matrix; the second part initialises the variables. The next parts choose the 5 jobs that will need minimum tool change from the current point. Section 5 applies the CE algorithm to find the optimal sequence of jobs for those 5 jobs. The columns of the *distance_matrix* corresponding to the scheduled jobs are changed to 1000. This ensures that already visited points will not be selected again. The algorithm continues this process until all points are visited. Once completed, the vector *visited* contains the near-optimal route of the manufacturing.

0: Take inputs
1: Calculate the distance_matrix
The distance_matrix is nxn matrix that contains the tool changeover times for switching from one job to another. The diagonals are zero, since processing the same job sequentially would not need any tool changes.
2: Set current_point = 1, visited = {1}, unvisited = {2, 3, . . .n}
3: Find the five closest points X according to the distance_matrix
4: Apply Complete Enumeration algorithm and find the optimal sequence for those five jobs, label the output as X′
5: Set visited = horzcat(visited, X′), current_point = last(visited)
6: Set unvisited = unvisited\X
7: Update the distance matrix; place 1000 to the columns that represent the already visited points.
8: If unvisited = ∅, stop. The solution is the vector visited. Otherwise, go to step 3.

4.4 Travelling Salesman problem (open tour)

Travelling Salesman Problem is an important part of operations research methodology and this approach may be used in sequence dependent setup modelling. Given a set of nodes and the distances between them, TSP aims to find the shortest route. These types of problems are NP (non-deterministic polynomial-time) hard and become challenging to solve when there are dozens of cities in the data set (Papadimitriou 1977). The TSP problem is formulated as follows:

$$\text{Min} \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij}$$

$$0 \leq x_{ij} \leq 1 \quad i, j = 0, 1, \ldots, n \tag{1}$$

$$\sum_{i=0, i \neq j}^{n} x_{ij} = 1 \quad j = 0, 1, \ldots, n \tag{2}$$

$$\sum_{j=0, j \neq i}^{n} x_{ij} = 1 \quad i = 0, 1, \ldots, n \tag{3}$$

$$u_i \in Z \quad i = 0, 1, \ldots, n \tag{4}$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad 1 \leq i \neq j \leq n \tag{5}$$

The first constraint defines the binary variables x_{ij}. The second constraint ensures that each node is visited from only one node, and the third constraint ensures that the path goes to only one node. The last constraint is for preventing sub-tours, i.e. the optimal solution consists of

one complete tour. In our case, the 'nodes' will be jobs and 'distance' will be the setup time required to switch from one job to another.

In order to apply the TSP model, several simplifications were introduced: (1) among different types of changeover components, only the number of tool changes is considered, (2) only the effect of the preceding job on the current job is considered. The higher order changeovers are ignored.

We build a distance matrix and continue the calculations based on that. In fact, this is a modified version of the CL approach; the difference is that the sequencing of jobs is done using TSP linear model. More precisely, the algorithm works as follows:

0: Take inputs
1: Calculate the distance_matrix
2: Set current_point = 1, visited = {1}, unvisited = {2,3,…n}
3: Find the five closest points X according to the distance_matrix
4: Apply TSP approach and find the optimal sequence for those five jobs, let us say the output is the set X′
5: Set visited = horzcat(visited, X′), current_point = last element of 'visited'
6: Set unvisited = unvisited\X
7: Update the distance matrix; place 1000 to the columns that represent the already visited points. (This ensures that the visited points will not be visited again.)
8: If unvisited = ∅, stop. The solution is the vector visited. Otherwise, go to step 3.

5 Evaluation

Algorithms were applied to the data sets that were explained in the methodology part. The prototype was implemented to connect with SMF machinery and provide solution for the task. The results are grouped according to the data sets. First, we start with data set 1, and then apply similar procedures to Datasets 2, 3 and 4 in order to replicate the findings.

Data set 1 is real data obtained from the case company. It includes a 13-day task list for the shearing punching machine with LSR6 robot, which is connected to automatic sheet storage. LPE is a combi machine with laser and punching features.
Data set 2 consists of 200 day's orders and was randomly generated based on actual orders.
Data set 3 was retrieved from a real factory operating a combined shearing/punching machine. It contains all the orders for the first quarter of the year 2015.
Data set 4 consists of 50 day's orders and was randomly generated based on actual orders.

5.1 Data set 1–testing the algorithms on a small data

We first give some descriptive information about the data set. In general, 16 different tools were utilised for the jobs altogether. The multi-tool MT8Ri is needed for the majority of the jobs. Some others, like NEL40 are used only a few times. The turret has 20 stations and the initial configuration is as shown in Fig. 6.

Here, ABCDE are normal stations and F, G, …, R are different types of multitools. Four out of twenty stations are multi-tool stations. Two of the multi-tool stations (numbers 15 and 17) are indexable, the other two (5 and 19) are non-indexable (fixed). For example, MT8_24 is a fixed multitool in the station 19; it is bolted to the turret permanently. Only

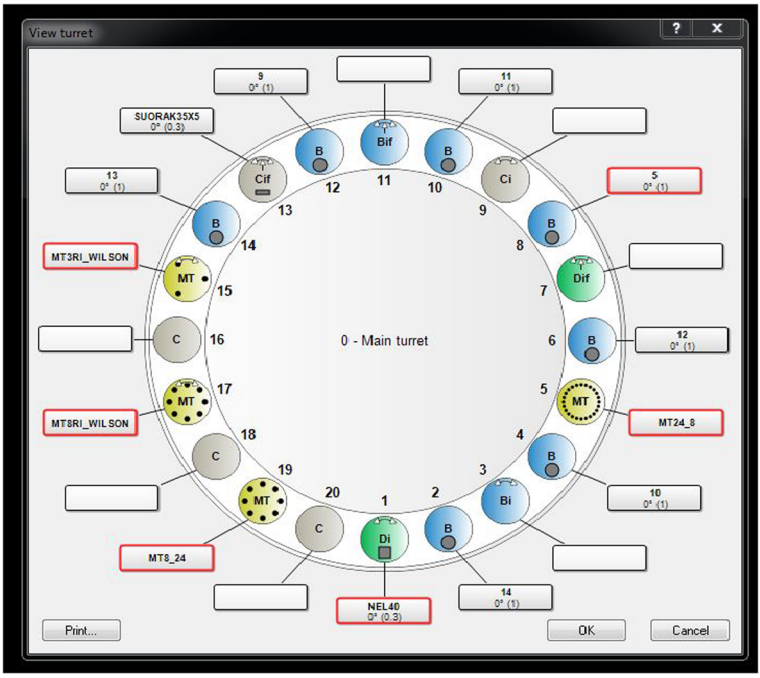

Fig. 6 Initial turret configuration on software user interface

multitool stations that include the letter 'i' are drop-in multitools and they can be installed to Di-stations. Multitools which include R (rotation) are indexable.

Fixed multi-tools are hard to disassemble and load new tools. Drop-in multi tools always have to be in indexable stations because they need to be rotated to select the correct tool inside them. Indexable means that a tool can rotate to the programmed angle freely. The rotation coordinate comes from NC-coordinate with C-axis command.

An extract of the outcomes of the algorithms are given below:

Day	# of Jobs	NN sequence	NN time	CL sequence	CL time	TSP sequence	TSP time
1	6	1–6–3–4–5–2	23	1–6–3–4–5–2	23	6–2–1–5–3–4	25

Based on these figures, the clustering algorithm performs better than the others. We used the worst possible sequence for benchmarking. The results show that the optimisation algorithms give better results as the number of jobs increase. The days with 2–6 jobs are simple; they can even be sequenced by the operator. As the complexity increases, the optimisation algorithms help more. In three cases, the Clustering algorithm decreases the setup time by 11, 25 and 23 % respectively.

With 2–3 jobs, the algorithms do not help much. To overcome this issue, we offer two ways:

1. The jobs can be combined as long as they meet the due date. For example, on day 1, there are 2 jobs and on the next day, we have 4 jobs. They could be combined (as long as this does not violate the due date constraint) so that we have 6 jobs. Then we can apply optimisation methods.

2. The second way is that when there are fewer than 5 jobs, we do not apply these optimisation methods. Instead, the workers can decide themselves easily with the help of the visualisation discussed on page 15.

The solution times of the algorithms are all within a reasonable range. TSP takes significantly more time than the others, but even that does not exceed 30 s.

One last point is that in this real data case, we had 16 tools in total and the turret had 20 stations. Thus, the number of tools is less than the number of stations in the turret. But by definition, the setup time reduction algorithms are more helpful in cases when there are far more tools than turret stations.

5.2 Data set 2–comparing the algorithms on the generated data

For this part of the work, a random sample with orders for 200 days was generated. Then the generated samples were solved using the Nearest Neighbour, Clustering and TSP algorithms. An extract from the Excel file is given below:

# of Jobs	NN	CL	TSP	DEF	NN–CL	CL–TSP	NN–TSP
11	192	207	196	209	−15	11	−4

The first column shows the number of jobs for each day, the second column is the total setup time if NN is used. The following two columns show the total setup time if clustering and TSP approaches are employed for solving the same problem. The column 'DEF' is the total setup time if the jobs are performed in the order of 1–2–3…n. This is our benchmarking value, because currently the company mainly uses this default order to process the orders. The last three columns display the differences NN–CL, CL–TSP and NN–TSP, respectively. For example, the first row is Day 1 and we have 11 different orders to be fulfilled. If we employ the sequence generated by NN approach, 192 min will be spent on setup. On the other hand, if CL algorithm is used, the setups will take 207 min. The difference is 15 min; i.e., by using the NN algorithm, we will save 15 min.

A simple analysis shows that CL performs better, because, for 126 out of 200 days, CL gives better solutions than NN. They perform equally for 11 days and for the remaining 63 days, NN's solutions are better. But this is not enough to draw conclusions about the performances of the two algorithms. The most preferred statistical approach is using paired *t*-test to check whether the samples are different or not. The main condition of the test is that the data should be normally distributed. As can be seen from the histograms below (Fig. 7), the data does not meet this condition.

Thus, we need to search for alternative tests. In such cases, **sign test** can be used; it does not require normality assumption. The null hypothesis of the sign test is that the median of the difference (NN–CL) is zero: $\mathbf{H}_0$: Median of the difference NN–CL = 0, $\mathbf{H}_1$: Median of the difference NN–CL $\neq 0$. After applying the test to our data set, the resulting p-value is 1.72×10^{-6}, so we reject the null hypothesis that the median of the difference is zero. This simply means that Clustering and Nearest Neighbour algorithms perform differently. To decide which of these two approaches yields a better result, we test the following hypothesis: $\mathbf{H}_0$: Median of the difference NN–CL = 0, $\mathbf{H}_1$: Median of the difference NN–CL > 0. The p-value of this test is 4.22×10^{-18}, which is why the null hypothesis is rejected. The conclusion is that the difference NN–CL is greater than 0, which simply means that applying CL algorithm results in shorter setup times.

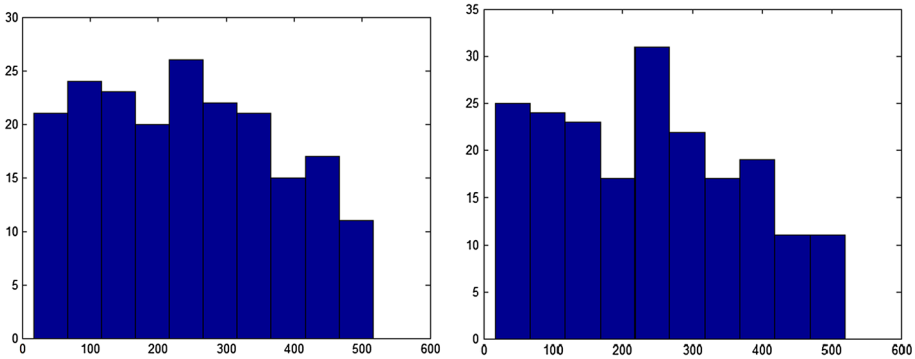

Fig. 7 Histograms of the total setup times for NN and CL, respectively

This can be confirmed by analysing the descriptive statistics of the 'NN–CL' column. The mean of the difference is 6.355 and median is 6. Thus, since the NN–CL difference is on average more than zero and the confidence interval is on the positive side, we conclude that **Clustering** yields better results than Nearest Neighbour. This is because, for example, if NN–CL = 5, it means that the total setup time of the route generated by CL is 5 min less than NN. That is how we decide whether one solution is better than another or not.

Following the same procedures for the TSP–CL difference, we get the p-value of 1.08×10^{-24}, meaning that CL also performs better than TSP.

While analysing the first data set, we saw that with up to 14 jobs per day, clustering algorithm gave better results. There were doubts about whether there is a threshold value that sets the border between two algorithms. It is not statistically correct to decide by analysing only one data set. Now that data from 200 random days is available, we can test the hypothesis. The test is built as follows: $\mathbf{H}_0$: For days with more than 14 jobs, median of the difference NN–CL = 0; $\mathbf{H}_1$: For days with more than 14 jobs, median of the difference NN–CL > 0. The p-value of the test is 2.74×10^{-8}, which means that we reject the null hypothesis. Again, we conclude that Clustering performs better than Nearest Neighbour approach. Testing for other possible threshold values did not give consistent results either; CL dominates NN in all cases.

5.3 Data set 3–comparing the algorithms on a real data

After testing the proposed algorithms on a randomly generated data set, the results were validated using the real data set 4 collected from a factory operating a sub-contracting service of sheet metal processing. Production mix is typically high in this environment and production volume relatively low. Columns *NN* and *CL* represent the total changeover times for NN and CL algorithms, respectively. *DIFF* is the difference of the columns NN and CL. *DEF* is the total changeover time if the jobs are processed using the default sequence, i.e. in the order of Job 1, Job2, …, Job n. The results are summarised in Table 3.

The average of the column DIFF is positive, proving our findings in the previous section. On average, CL sequencing saves 10.4 % of the setup time compared to the default sequencing that is currently used.

5.4 Sensitivity analyses

Sensitivity analysis is an important part of any optimisation problem. In our case, we discuss how the algorithms behave when (i) tool changes, (ii) angle changes and (iii) clearance

Table 3 Comparison of the algorithms

Day	No. of jobs	NN	CL	DIFF	DEF	CL % better than DEF
1	14	41	30	11	34	11.8
2	25	50	52	−2	58	10.3
3	17	67	63	4	75	16.0
4	12	50	56	−6	59	5.1
5	11	51	52	−1	54	3.7
6	18	40	43	−3	53	18.9
7	16	39	45	−6	49	8.2
8	16	84	81	3	82	1.2
9	17	84	90	−6	107	15.9
10	14	87	89	−2	105	15.2
11	11	76	85	−9	96	11.5
12	11	79	79	0	87	9.2
13	14	63	54	9	64	15.6
14	19	86	76	10	104	26.9
15	13	73	70	3	73	4.1
16	13	29	45	−16	45	0.0
17	23	85	76	9	82	7.3
18	21	102	92	10	98	6.1
					Avg	10.4

changes dominate. The idea is to test what happens when there are a lot of tool, angle or clearance changes. In practice, this means that some companies might receive such orders that they use the same set of tools to manufacture the parts. But the majority of setup time might be devoted to angle changes, so the rest might be ignored.

(i) When tool changes dominate—The hypothesis test is built as follows: $\mathbf{H}_0$: Median of the difference NN–CL = 0, $\mathbf{H}_1$: Median of the difference NN–CL > 0. The p-value is 4.22×10^{-28}, the null hypothesis is rejected. To test which of these two algorithms is better, we need to test the following: $\mathbf{H}_0$: Median of the difference TSP–CL = 0, $\mathbf{H}_1$: Median of the difference TSP–CL > 0. The p-value is ≈ 0; again, there is no evidence to support the null hypothesis. Clustering works better than NN and TSP models.

(ii) When angle changes dominate: To test what would happen if the production process faces a lot of angle changes, we modified the first data set such that all jobs require almost the same tools, but there are lots of angle changes. The total changeover time is 125 for NN, 136 for CL and 149 for TSP. Again, Nearest Neighbour algorithm provides a better solution than the others. This was also proven by using the large sample data: $\mathbf{H}_0$: Median of the difference NN–CL = 0, $\mathbf{H}_1$: Median of the difference NN–CL > 0. p-value is 5.75×10^{-20}, the null hypothesis is rejected. Testing the hypothesis $\mathbf{H}_0$: Median of the difference TSP–CL = 0, $\mathbf{H}_1$: Median of the difference TSP–CL > 0 yields p-value of ≈ 0; again, there is no evidence to support the null hypothesis. CL dominates the NN and TSP algorithms, and gives better scheduling in the case of many tool changes.

(iii) When die clearance changes dominate: The hypotheses are the same as before. Testing the hypothesis NN–CL=0 versus NN–CL>0 yields a p-value of 0.99; we fail to reject

Table 4 Sensitivity analysis results

Null hyp.	Tool changes		Angle changes		Die clearance changes	
	NN–CL = 0	TSP–CL = 0	NN–CL = 0	TSP–CL = 0	NN–CL = 0	TSP–CL = 0
p-value	4.22×10^{-28}	0	5.75×10^{-20}	0	0.99	0
Better alg.	CL	CL	CL	CL	–	CL

the null hypothesis. Testing the hypothesis TSP–CL = 0 versus TSP–CL > 0 results in a *p*-value of 0; thus, there is no evidence to support the null hypothesis. The conclusion is that CL performs better than TSP, but there is not enough statistical evidence that it is better than NN. The findings of the sensitivity analyses are summarised in Table 4. In five out of six cases, CL yields a better processing sequence when compared to others.

6 Conclusions

Setup time reduction is a challenge that many manufacturing companies face during their daily operations. The job sequencing problem has been known for a long time and there are several alternatives for solution. The information retrieved from actual users showed that FIFO sequence is very often used and there is potential to improve capacity utilization by considering tool setups. In this paper we analysed cases where setup times are sequence-dependent and proposed several algorithms.

We received real data from a machinery and also generated similar sample data from these orders to have better statistical reliability. Three different approaches, Nearest Neighbour, Clustering and Travelling Salesman Problem were employed to solve the problem under examination. TSP approach was used under several assumptions, but it did not give the expected results and was dominated by CL and NN. Comparative analyses of CL and NN approaches showed that in the majority of cases, CL algorithm performs better. Then, sensitivity analysis was done to test the performance of the algorithms in various situations.

Based on the data analysis, the sequences generated by CL algorithm saved more than 10 % of the setup times when compared to the default FIFO sequencing. One main aspect of this problem is the solution time and the discussed algorithms solved the sequencing problems in less than 30 s on average. This is very important from a practicality point of view. The developed prototype system and the analysis conducted from data collected from operational machines showed that embedded automatic scheduling systems have great potential as companies tend to schedule work order without analysis of setups. By applying these principles, smart connected machines may request improved operation sequence by utilizing information of production orders in the queue as well as tool information. Embedded big data analytics can improve operational efficiency.

The limitation of this study is that, we analysed the data received from a equipment manufacturer company, which represents example of sub-contracting type of operation. Other operational profiles such as high volume–low product mix might represent different type of order patterns. In the future, the we will attempt to replicate this work with more data from other companies beyond the prototype setup. Also, this work could be replicated in other industries than SMF as these algorithms are applicable not only in the SMF industry, but also in similar situations where setups are sequence-dependent and typical industry standard is to

run FIFO with due date adjustments. That would provide more information about the setup time savings of the proposed algorithms.

References

Adriaans, P., & Zantinge, D. (1996). *Data mining. Harlow*. England: Addison-Wesley.

Bowers, M. R., Groom, K., Ng, W. M., & Zhang, G. (1995). Cluster analysis to minimize sequence dependent changeover times. *Mathematical and Computer Modelling*, *21*(11), 89–95.

Burtseva, L., Parra, R. R., & Yaurima, V. (2010). *Scheduling methods for hybrid flow shops with setup times*. Rijeka: INTECH Open Access Publisher.

Cheung, W., & Zhou, H. (2001). Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times. *Annals of Operations Research*, *107*(1–4), 65–81.

Dubey, R., Gunasekaran, A., Childe, S. J., Wamba, S. F., & Papadopoulos, T. (2016). The impact of big data on world-class sustainable manufacturing. *The International Journal of Advanced Manufacturing Technology*, *84*(1–4), 631–645.

Gawroński, T. (2012). Optimization of setup times in the furniture industry. *Annals of Operations Research*, *201*(1), 169–182.

Giglio, D. (2015). Optimal control strategies for single-machine family scheduling with sequence-dependent batch setup and controllable processing times. *Journal of Scheduling*, *18*(5), 525–543.

Gupta, Sushil K. (1982). N jobs and m machines job-shop problems with sequence-dependent set-up times. *The International Journal of Production Research*, *20*(5), 643–656.

Hwang, H., & Sun, J. U. (1998). Production sequencing problem with re-entrant work flows and sequence dependent setup times. *International Journal of Production Research*, *36*(9), 2435–2450.

Kalayci, C. B., Polat, O., & Gupta, S. M. (2014). A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. *Annals of Operations Research*, 1–34.

Lee, J., Kao, H. A., & Yang, S. (2014). Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia CIRP*, *16*, 3–8.

Lee, J., Lapira, E., Bagheri, B., & Kao, H. A. (2013). Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, *1*(1), 38–41.

Liu, Z., Wang, Y., Cai, L., Cheng, Q., & Zhang, H. (2016). Design and manufacturing model of customized hydrostatic bearing system based on cloud and big data technology. *The International Journal of Advanced Manufacturing Technology*, *84*(1–4), 261–273.

Lockett, A. G., & Muhlemann, A. P. (1972). Technical note–a scheduling problem involving sequence dependent changeover times. *Operations Research*, *20*(4), 895–902.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., & Barton, D. (2012). Big data. *The Management Revolution. Harvard Business Review*, *90*(10), 61–67.

Mirabi, M. (2011). Ant colony optimization technique for the sequence-dependent flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, *55*(1–4), 317–326.

Nonaka, Y., Erdős, G., Tamás, K., Nakano, T., & Váncza, J. (2012). Scheduling with alternative routings in CNC workshops. *CIRP Annals-Manufacturing Technology*, *61*(1), 449–454.

Papadimitriou, Christos H. (1977). The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, *4*(3), 237–244.

Pinedo, Michael L. (2008). *Scheduling: Theory, algorithms, and systems*. New Jersey: Prentice-Hall.

Porter, M. E., & Heppelmann, J. E. (2014). How smart, connected products are transforming competition. *Harvard Business Review*, *92*(11), 64–88.

Rao, Y., Huang, G., Li, P., Shao, X., & Daoyuan, Y. (2007). An integrated manufacturing information system for mass sheet metal cutting. *The International Journal of Advanced Manufacturing Technology*, *33*(5–6), 436–448.

Reza Hejazi*, S., & Saghafian, S. (2005). Flowshop–scheduling problems with makespan criterion: A review. *International Journal of Production Research*, *43*(14), 2895–2929.

Rudek, R. (2012). Scheduling problems with position dependent job processing times: Computational complexity results. *Annals of Operations Research*, *196*(1), 491–516.

Spence, Anne M., & Porteus, Evan L. (1987). Setup reduction and increased effective capacity. *Management Science*, *33*(10), 1291–1301.

Tao, F., Zhang, L., Nee, A. Y. C., & Pickl, S. W. (2016). Editorial for the special issue on big data and cloud technology for manufacturing. *The International Journal of Advanced Manufacturing Technology*, *84*(1–4), 1–3.

von Luxburg, U., Bubeck, S., Jegelka, S., & Kaufmann, M. (1981). Nearest neighbor clustering. *Annals of Statistics*, *9*(1), 135–140.

Westphal, S., & Noparlik, K. (2014). A 5.875-approximation for the traveling tournament problem. *Annals of Operations Research*, *218*(1), 347–360.

White, Charles H., & Wilson, Richard C. (1977). Sequence dependent set-up times and job sequencing. *The International Journal of Production Research*, *15*(2), 191–202.

Witten, Ian H., & Frank, Eibe. (2005). *Data mining: Practical machine learning tools and techniques*. Burlington: Morgan Kaufmann.

Zandin, K. B. (Ed.). (2001). *Maynard's industrial engineering handbook*. New York: McGraw-Hill.

Zhong, R. Y., Lan, S., Xu, C., Dai, Q., & Huang, G. Q. (2016). Visualization of RFID-enabled shopfloor logistics Big Data in Cloud Manufacturing. *The International Journal of Advanced Manufacturing Technology*, *84*(1–4), 5–16.