Pre-optimizing tools positions for turrets-based CNC machines when facing arbitrary sequences of production ⋆

Marc-André Ménard * **Claude-Guy Quimper** *
Jonathan Gaudreault *

* *CRISI Research Consortium for Industry 4.0 Systems Engineering, Université Laval, Québec, Canada, marc-andre.menard.2@ulaval.ca, claude-guy.quimper@ift.ulaval.ca, jonathan.gaudreault@ift.ulaval.ca*

Abstract: Turret-based CNC machines can change from one tool to another by simply rotating the turret. However, each tool must be positioned at the exact same place on the turret for each batch of the same product. In industrial context, the production sequence of the different products are not known in advance. Therefore, we introduce an integer programming model that determines the best positions for each tool and for each product. It minimizes the expected setup time. Experiments are carried out in partnership with APN, a company engaged in high precision machining in the aerospace industry. We were able to solve the problem to optimality for each dataset, reducing setup time by 191 hours.

Keywords: Operations research, Optimization, Integer programming, Linear programming, CNC

1. INTRODUCTION

Computer Numerical Control machines (CNC) are used for high-precision workpiece machining. The CNC takes as input a CNC program that dictates the actions that the machine must perform to obtain the desired product.

A very common type of CNC makes use of a rotating turret that can contains up to 12 machining tools. A simple rotation of the turret allows the machine to activate a tool. Before starting a batch of a given product, one has to manually install the needed tools at the right position of the turret. The CNC program implicitly defines the right place on the turret for each tool. This program was established by a human programmer before the product entered into production for the first time. This program/positioning of the tools insures against unused tools from physically interfering with the manufactured product or with other tools during movements. Creating this program is a complicated task (carried on by a human), thus the program cannot be easily modified and the tools positions will be the same anytime the company will hereafter manufacture the product.

Although companies tries scheduling production in a way that reduces setup time, it is not always possible as it is very related to orders arrival sequence.

Thus, it would be beneficial to design the CNC programs (thus, selecting tool positions) in a way that would reduce the expected setup time of future production (altough we do not know future product. As an example, if a given tool is always at the same position in any CNC program, setup time is decreased by much.

1.1 Problem description

For each type of product T manufactured by the company, and for each CNC machine M involved in its manufacture, the company must define a *CNC program* defining the entire movements made by the machine working on this part (turret movements of the machine to alter the workpiece, rotation of the turret to change the active tool, etc.). There is therefore a CNC program for each pair $\langle T, M \rangle$.

Whenever a machine is configured to manufacture a product, one has to manually install on its turrets the tools it has to use. For each product, we have a list of tools X needed to machine the product. For each tool $x \in X$, we must determine, the position Y to which assign the tool x to manufacture product T. It is desirable that the same tool used to manufacture two products T_1 and T_2 be assigned in both cases to the same position to reduce the setup time.

To summarize, the problem is to assign the tools to the various positions of the turrets to reduce the setup time needed to move from the manufacture of a product to the manufacture of another product.

However, we do not know the production sequence. Indeed, there are several different products that can be manufactured on one machine. The best production sequence changes from one week to the next one and if one changes the position of the tools on the turret between two productions of the same product, then one has to update the CNC program.

⋆ The authors would like to thank APN inc, CRIQ and the Natural Sciences and Engineering Research Council of Canada (NSERC) who provided funding for this research.

The tools used by the CNC machines are composed of several parts partitioned into "levels". All tool parts put together form an assembly of tools. The first level is the lower part of the tool assembly called the *base* which is inserted in the turret. The second and third levels are intermediate components. The last level is the actual cutting tool that is in contact with the product.

It takes longer to change the first level of a tool than to change only the top level, etc. Indeed, changing a lower level involves changing the entire tool while it is possible to leave in place the lower levels when changing an upper level. It is therefore more important to minimize the number of changes of the lower levels than to minimize the changes in the upper levels.

Since we do not know the sequence of production, we cannot minimize the number of changes in the sequence of production. The objective of the problem is to minimize for each pair of product, the number of "avoidable" changes. An *avoidable change* is a change that could be avoided by placing the tool at the same position in two tool lists. In Table 1, the shown solution has 2 changes related to tool B. However, the change caused by the tool C (when going from product 1 to product 2, or from product 1 to product 3) is unavoidable (product 2 and product 3 do not use the tool C).

However, one can also decide to add tools that are not required by the program. Indeed, adding non required tools can contribute to minimizing tool changes. As shown in Table 1, after machining Product 1 that uses tool A, one would have to remove the tool A to machine the second product and add it back to machine the third product. If tool A was added to the tool list of Product 2, then no tool change is required for A. To add a tool, we need to add a whole tool assembly. It is not possible to add only the base of the tool assembly without adding the upper parts. Not all tools can be added to a product tool list. A company might decide that it is not possible to add a tool if it is too expensive to add or if there are not enough of that tool in the inventory.

Table 1. Example of a change at position 1

	Position 1	Position 2	Position 3
Product 1	A	B	C
Product 2		D	B
Product 3	A	D	B

1.2 Related work

A wide range of related problems can be found in the literature under the term tooling management problems (Kayayama (1994)), or more precisely job sequencing and tool switching problems (Crama et al. (1994, 2007); Zhou et al. (2005); Amaya et al. (2008)). Calmels (2018) does a literature review on job sequencing and tool switching problem. They present several variants of the problem such as taking into consideration the wear of the tools (Hirvikorpi et al. (2007); Mauergauz (2017)) or having several machines in parallel with different tools in each machine (Beezo et al. (2017); Sarmadi and Gholami (2011)). However, this literature concerns a situation opposite to ours, that is, to determine the optimal positions of the tools after having determined the sequence of the products to be manufactured (Adjiashvili et al. (2015)) or at the same time (Laporte et al. (2004); Catanzaro et al. (2015)). This approach makes sense only for certain types of CNC machines handling one tool at a time, the other tools being stored in an "external store" that allows the machine to change tools as needed during product manufacturing.

On the other hand, for the machines rather equipped with turrets (the industrial case studied in this paper), the activation of a tool during manufacturing a product is done quickly by making a rotation of the turret. If one change the position of the tools on the turret between two productions of the same product, the CNC program need to be updated. Indeed, it must be ensured that the other tools do not physically interfere with the manufactured product or a tool from another turret of the machine during the machining operations. Changing a CNC programs cannot be done automatically since the machines are equipped with several turrets, and the times required to perform the operations of each turret are not known in advance (it depends on several criteria not taken into account by existing simulators), it is impossible to ensure that there will be no snagging without doing a physical *run test* on the machine. This CNC program update takes time from an experienced technician and monopolizes the machine for part of this work, which increases setup time well beyond what repositioning can save.

In the industrial context studied (machining using CNC machines equipped with several turrets), the classical approach common in the literature (scheduling before or during the assignment of the tools to the positions) increases the setup time even more than it would save.

2. INTEGER PROGRAM

2.1 The model

We designed an integer program to solve this problem. Since it is more important to minimize the number of tool changes in the lower levels than the higher levels, we solve the problem in stages. We begin by solving the problem by considering only the lowest level of the tool assemblies. Subsequently, we consider the first two levels of tool assemblies, but by fixing the number of changes allowed at the first level to the objective value found in the previous resolution for the first level. By setting the objective value for the first level, we make sure to keep the optimality for the first level, but we allow the solver to change the position of the tools of the first level. Since there may be more than one optimal solution, it is not necessarily the solution found previously for the first level that will minimize the number of changes for the second level. Then we solve the problem for the next level until we get a solution considering all levels of tool assemblies. When the solver solves the problem considering several levels, the solver must consider the precedence of the tool assemblies. For example, consider a tool assembly t whose part of the tool at the first level is t_1 and the part of the second level is t_2. If the solver sets the tool part t_2 to a position, it must put t_1 at the same position to respect the precedence of the tool assembly.

We present below the model designed to solve the problem of tool positioning. We present the program parameters in Table 2 and present the program variables in Table 3.

The first parameters of Table 2 are the set of positions given by $P \in \{1, \ldots, 12\}$ and the set of levels $N \in \{1, \ldots, 4\}$. T is the set of tools, $T^+ \subseteq T$ are the tools that can be added to a tool list, and $T_n \subseteq T$ are the tools that take place on level $n \in N$. A tool in $t_2 \in T_{n+1}$ must be installed on a tool $t_1 \in T_n$. The set Q contains the pairs (t_1, t_2) of possible assemblies. A *tool list* is a multiset of T required for a product. The set of tool list indices is L. For each tool list $l \in L$, let f_l be its frequency indicating how many times, in a one-year period, the program is expected to be executed. A tool $t \in T$ occurs z_{tl} times in the tool list $l \in L$ and occurs $\text{occ}_t = \sum_{l \in L} f_l \cdot z_{tl}$ times in all tool list. The parameter m_{ln} indicates the maximum number of tools it is possible to have for the tool list l and the level n. Initially, m_{ln} is always $|P| = 12$, but an improvement to the model that will be shown below changes this value. Let $S = \sum_{l \in L} f_l$ be the total number of products produced in a year. The parameter $L^{\text{fixed}} \in L$ is the tool list that requires the most distinct tools, ties are broken arbitrarily. The parameter $L_n^{\text{same}} \subseteq 2^L$ contains sets of tool lists that require the same tools at level n. Indeed, it is possible, especially considering only the first levels of the assemblies of tools, to have two lists of identical tools for two different products.

In the first stage, we solve the model for $N = \{1\}$ and store the objective value in O_1. In the second stage, we solve for $N = \{1, 2\}$ and store the objective value in O_2 and so on for $N = \{1, 2, 3\}$ and $N = \{1, 2, 3, 4\}$.

Table 2. Parameters

Parameters	Description
P	Set of positions.
N	Set of levels.
T	Set of tools.
T^+	Set of tools that can be added even if the tool list does not need them ($T^+ \subseteq T$).
T_n	Set of tools of level n.
Q	Set of tool parts precedences. $(t_1, t_2) \in Q$ indicates that tool t_2 must be on the tool t_1.
L	Set of tool lists.
z_{tl}	The number of occurrences of the tool t needed for the tool list l.
occ_t	Number of times the tool t appears in a tool list, i.e. $\text{occ}_t = \sum_{l \in L} f_l \cdot z_{tl}$.
m_{ln}	Maximum number of tools in tool list l for the level n.
f_l	Frequency of the tool list l for a period of one year.
S	Sum of all tool list frequencies over a period of one year i.e. $S = \sum_{l \in L} f_l$.
L^{fixed}	The tool list with the most distinct tools.
L_n^{same}	Sets of tool lists that require the same tools at level n.
O_n	Optimal objective value for the the level n.

There are two types of binary variables in the model. The variable a_{tlp} indicates whether the tool t is at position p for the tools list l. The variable b_{tpk} equals 1 if the tool t appears k times at the position p.

Table 3. Variables

Variables	Domain	Description
a_{tlp}	$\{0, 1\}$	$a_{tlp} = 1$ if tool t in tool list l is assigned to position p and $a_{tlp} = 0$ otherwise.
b_{tpk}	$\{0, 1\}$	$b_{tpk} = 1$ if and only if tool $t \in T$ occurs $k \in \{0, ..., S\}$ times at position $p \in P$.

$$\text{minimize} \quad \frac{1}{2} \sum_{t \in T_{max(N)} \setminus T^+} \sum_{p \in P} \sum_{k=0}^{S} k \cdot (\text{occ}_t - k) \cdot b_{tpk} \quad (1)$$
$$+ \sum_{t \in T^+_{max(N)}} \sum_{p \in P} \sum_{k=0}^{S} k \cdot (S - k) \cdot b_{tpk}$$

Basic model

$$\sum_{t \in T_n} a_{tlp} \leq 1 \qquad \forall n \in N, l \in L, p \in P \quad (2)$$

$$\sum_{k=0}^{S} b_{tpk} = 1 \qquad \forall t \in T, p \in P \quad (3)$$

$$\sum_{l \in L} f_l \cdot a_{tlp} = \sum_{k=0}^{S} k \cdot b_{tpk} \qquad \forall t \in T, p \in P \quad (4)$$

$$\sum_{p \in P} a_{tlp} = z_{tl} \qquad \forall l \in L, t \in T \setminus T^+ \quad (5)$$

$$\sum_{p \in P} a_{tlp} \geq z_{tl} \qquad \forall l \in L, t \in T^+ \quad (6)$$

$$a_{t_1lp} \geq a_{t_2lp} \qquad \forall l \in L, p \in P, (t_1, t_2) \in Q, \text{ such that } t_2 \notin T^+ \quad (7)$$

$$a_{t_1lp} \leq \sum_{t_2 \in T | (t_1, t_2) \in Q} a_{t_2lp} \qquad \forall t_1 \in T, l \in L, p \in P \quad (8)$$

$$O_n = \frac{1}{2} \sum_{t \in T_n \setminus T^+} \sum_{p \in P} \sum_{k=0}^{S} k \cdot (\text{occ}_t - k) \cdot b_{tpk} \quad (9)$$
$$+ \sum_{t \in T_n^+} \sum_{p \in P} \sum_{k=0}^{S} k \cdot (S - k) \cdot b_{tpk} \qquad \forall n < \max(N)$$

Removing unique tools

$$\sum_{t \in T_n} \sum_{p \in P} a_{tlp} \leq m_{ln} \qquad \forall n \in N, l \in L \quad (10)$$

Breaking symmetries

$$a_{tL^{\text{fixed}}p} = 1 \qquad \forall t \text{ is the } p\text{th tool in } L^{\text{fixed}} \quad (11)$$

Cuts

$$a_{tl_1p} = a_{tl_2p} \qquad \forall t, l_1, l_2, \exists a \in L_n^{\text{same}}, \{l_1, l_2\} \subseteq a \quad (12)$$

$$\sum_{p \in P} \sum_{k=0}^{S} k \cdot b_{tpk} = \sum_{l \in L} f_l \cdot z_{tl} \qquad \forall t \in T \setminus T^+ \quad (13)$$

Forbiding changes

$$r_t := \text{occ}_t - \min_{l \in L | z_{tl} > 0} f_l \qquad \forall t \in T \quad (14)$$

$$a_{tl_1p} \leq a_{tl_2p} \qquad \forall l_1 \in L, l_2 \in L, \forall p \in P, n < \max(N), t \in T_n, r_t > O_n \wedge z_{tl_1} \leq z_{tl_2} \quad (15)$$

2.2 The objective function

The objective function (1) minimizes the total number of avoidable tool changes between each pair of products. The first part of the function counts the changes for the tools $T \setminus T^+$ that cannot be added to the tool lists. It sums the number of pairs of tool lists for which the tool is not placed at the same position. For instance, if a tool t occurs k times at position p, then it occurs $\text{occ}_t - k$ times at another position than p. This leads to $k \cdot (\text{occ}_t - k)$ pairs of tool lists for which the tool appears in different positions and could be avoided. Since we consider each position p, the pairs are counted twice: one for the pair (l_1, l_2) and once for the pair (l_2, l_1). This is why we divide the final result by 2.

$$\text{minimize} \quad \frac{1}{2} \sum_{t \in T_n \setminus T^+} \sum_{p \in P} \sum_{k=0}^{S} k \cdot (\text{occ}_t - k) \cdot b_{tpk} \tag{16}$$

Notice that some tool changes are counted when they should not be counted. For example, if a tool occurs twice in the tool lists l_1 and l_2 and that this tool takes the positions p_1 and p_2 for both lists, the function still counts 2 tool changes. Indeed, the tool appears at position p_1 in list l_1 and position p_2 in list l_2 which counts for one change and the tool appears at position p_2 in list l_1 and position p_1 in list l_2 which counts for another change. This even occurs when $l_1 = l_2$. In order not to count these changes, one needs to subtract the term in (17) from the objective function. Since this term is a constant and has no impact on the optimality of the solution, we omit it from the model for sake of simplicity.

$$\frac{1}{2} \sum_{t \in T_n \setminus T^+} \sum_{l_1 \in L} \sum_{\substack{l_2 \in L \\ z_{tl_1} = z_{tl_2}}} f_{l_2} \cdot z_{tl_2} \cdot f_{l_1} \cdot (z_{tl_1} - 1) \tag{17}$$

A similar situation occurs when a tool occurs more often in list l_1 than in l_2, i.e. $z_{tl_1} > z_{tl_2}$. The correction constant is given in (18). Since this case is not symmetric, we do not have to divide by two as it is done for (17). Once again, since this term is a constant, we omit it from the model for sake of simplicity.

$$\sum_{t \in T_n \setminus T^+} \sum_{l_1 \in L} \sum_{\substack{l_2 \in L \\ z_{tl_1} > z_{tl_2}}} f_{l_2} \cdot z_{tl_2} \cdot f_{l_1} \cdot (z_{tl_1} - 1) \tag{18}$$

Overall, the constant c should be subtracted from the objective function in order to obtain the exact number of avoidable changes.

$$\begin{aligned} c = \frac{1}{2} & \sum_{t \in T_n \setminus T^+} \sum_{l_1 \in L} \sum_{\substack{l_2 \in L \\ z_{tl_1} = z_{tl_2}}} f_{l_2} \cdot z_{tl_2} \cdot f_{l_1} \cdot (z_{tl_1} - 1) \\ + & \sum_{t \in T_n \setminus T^+} \sum_{l_1 \in L} \sum_{\substack{l_2 \in L \\ z_{tl_1} > z_{tl_2}}} f_{l_2} \cdot z_{tl_2} \cdot f_{l_1} \cdot (z_{tl_1} - 1) \end{aligned} \tag{19}$$

The second part of the objective function (20) counts the avoidable changes for tools T^+ that can be added. Since such a tool can be added to any tool list, any change that occurs is avoidable. We therefore consider that if a tool occurs k times at position p, it could have occurred S times at such a position.

$$\text{minimize} \quad \sum_{t \in T_n^+} \sum_{p \in P} \sum_{k=0}^{S} k \cdot (S - k) \cdot b_{tpk} \tag{20}$$

2.3 The basic model

The basic model, formed with constraints (2) to (9), is sufficient to find a feasible solution to the problem. Other constraints are cuts that help the solver to find a solution faster. Constraints (2) ensure there is at most one tool part for a given level, a given position, and a given tool list. Constraints (3) and (4) ensure $b_{tpk} = 1$ if and only if the tool t is installed at position p exactly k times a year. Constraints (5) ensure that all the necessary tools for machining a product have a position. For tools which can be added ($\forall t \in T^+$), constraints (6) ensure there are no fewer tools than necessary for machining a product. Constraints (7) and (8) ensure that the tool assembly is respected. If the upper part of the tool assembly is in one position, the lower part of that assembly is at the same position. Moreover, if a lower part of the tool assembly is in one position, then there is an upper part at the same position. Constraints (9) ensure the number of avoidable changes remains optimal for lower levels. Only the objective value is fixed which makes it possible to change the position of the tools of the lower levels while maintaining the optimality.

2.4 Improvements

A possible improvement to reduce the size of an instance is to remove from the problem the tools that occur in a single tool list. Unique tools can only cause inevitable changes with other tools. By removing these tools, we remove a constant from the objective value. However, we need to make sure to have an available position for the removed tool after solving the problem. By decrementing m_{ln} each time a unique tool t is removed from the tool list l, the constraints (10) ensure that a position remains available to install the tool t.

The problem has symmetries. Indeed, tools at position p_1 can be swapped with tools at position p_2 to obtain a new, but symmetric, solution. To break these symmetries and hence reduce the search space, constraint (11) selects the tool list L^{fixed} with the most distinct tools and arbitrarily affects a position to every tool in this tool list.

Constraints (12) add simple cuts that force two identical tool lists, for a given level, to have the same tool placement. This is particularly useful in lower levels where such tool lists are common. Optimal solutions necessarily satisfy these constraints.

Constraints (13) are redundant, but they nevertheless help to solve the model by further constraining the variables b without pruning valid solutions.

Constraints (14) define the constant r_t that is the minimum number of changes that occurs when the tool $t \in T_n$ is not always assigned to the same position. Since O_n is the optimal value when solving the problem for the first n levels, if $r_t > O_n$, this means no avoidable changes can occur for tool t. The first $\min(z_{tl_1}, z_{tl_2})$ occurrences of tool t in list l_1 and l_2 must be assigned to exactly the same positions. This is enforced by constraints (15).

2.5 Analysis

The complexity of the model depends on the number of tool lists $|L|$, the number of tools $|T|$, the number of positions $|P|$, and the number of levels $|N|$. The model has $O(|P||T||L| + S|P||T|)$ binary variables and $O(|L|^2|T||P||N| + |L||P||Q|)$ constraints.

3. EXPERIMENTS

APN inc. is a US and Canada based company machining high precision metal parts mainly for the aerospace and military industries. They contacted us to discuss the problem and provided us with real datasets for their production over a period of one year on 6 different CNC machines. Each machine has 2 turrets with 12 tool slots. With 6 machines and 2 magazines per machine, we have 12 different instances.

We do not know the production frequency of each products for instances 3 to 12. So, for experiments using those instances, we take a random number between 1 and 10 as the frequency of the product.

The company only allowed us to add a tools on the tool list of a given product if the entire tool assembly appears in more than half of the tool lists. We defined the set T^+ as such.

The experiments were executed on an Intel Core i7 4.0GHz with 16 GB of memory. We use CPLEX version 12.6.1 to solve the linear program.

Table 4 compares the time taken by the solver to find the optimal solution to the different instances of the problem using the different improvements of the model. The header of each column indicates which constraints were used to solve the problem. A "−" as a result indicates that the solver did not find a solution or proved the optimality of the solution within one hour of computation.

Table 4 shows that the basic model, on its own, only solves two instances. Therefore, it is necessary to use improvements to the model to solve the other instances of the problem.

Removing the unique tools makes it possible to find the optimal solution for two more instances. This improvement also reduced the time to find the optimal solution for the instances that could be solved by the basic model. By reducing the number of tools, we reduce the number of variables and constraints which makes it easier for the solver to find the solution.

Optimal solutions can be found for two additional instances when we break symmetries. For the other instances that were already possible to solve, removing the symmetries allowed to reduce the time by 82% on average.

Removing the identical lists allowed improving the time spent by the solver finding the optimal solution of 4 more instances. However, there is only instance 8 that takes longer to solve with this improvement.

Adding the cut (13) solved 2 more instances. This improvement also reduced the time to solve instances by 37% on average.

Forbidding changes makes it possible to solve all the instances of the problem to optimality. For all instances of the problem except instances 1, 6, and 10, the solver finds the optimal solution within one minute. For instances 1, 6, and 10, the solver takes less than 3 and a half minutes which is short enough to be used by our industrial partner.

The results presented in Table 5 show the contribution of having optimal solutions compared to what is currently used by the company APN. The results are approximated. Indeed, we do not know in advance the production sequence for next year. We therefore generate a random production sequence and compare, for this sequence, the difference between the number of changes with the solution of the model and the positioning of the tools currently used by the company. We do this experiment 5 times and take the average. The number of times a product is machined depends on its frequency. We do not allow to have a product appearing twice in a row in the sequence.

The first four columns of Table 5 show the percentage of the number of changes saved during the year at a given level. For all instances and all levels, the optimized solutions reduce the number of tool changes with respect to what is currently used by the company. The percentage of gain is higher for the first level. This is a desirable behavior since tools on the first level take more time to change than upper levels. By optimizing the first level and fixing the number of changes for that level, we prioritize the minimization of changes at level one.

The last column of Table 5 shows the setup time saved using the solver solution instead of the solution currently used by the company. This saved time is again an approximation since the sequence of productions were randomly generated. Moreover, there is an approximation of the time taken to remove a tool and add a tool. For each level, the company has determined an average time to add a tool and to remove a tool. This time includes the time the machine is stopped, the time to prepare the assembly of tools, and to clean the tools assembly. It does not include the calibration of the tools assembly. In the context of high-precision machining, tool calibration can take most of the setup time. If we do not need to change the tool, we save more time by avoiding calibration of the tool assembly. The average time used does not include extra times that occurs when there are errors in a tool assembly. It takes an average of 2 and a half minutes to add a tool and 3 and a half minutes to remove the tool for a change of the first level. A change at the second level takes 1 minute and a half to add the tool and 2 minutes to remove the tool. A change at the third level takes 1 minute to add the tool and 1 minute and a quarter to remove the tool. A change at the fourth level takes 30 seconds to add the tool and 30 seconds to remove the tool.

The last column of Table 5 shows that even without taking all the time into consideration for the change of a tool such as tool calibration, it is possible to save about 191 hours of setup time, which is more than significative. This represents 21.5 % of setup times for the jobs we considered.

It would be possible to get better results if we allow the solver to add more tools to the tool lists. Indeed, the company only allows adding a tool assembly if it appears

Table 4. Time in seconds needed to solve the instances of the problem.

	Basic model (2) to (9)	Removing unique tools (2) to (10)	Breaking symmetries (2) to (11)	Removing identical lists (2) to (12)	Cuts on b (2) to (13)	Forbiding changes (2) to (15)
Instance 1	-	-	-	-	369	205
Instance 2	-	-	-	-	230	7
Instance 3	-	3001	267	111	12	3
Instance 4	163	80	5	5	4	3
Instance 5	-	-	-	-	-	50
Instance 6	-	-	-	-	-	82
Instance 7	-	-	1149	195	156	36
Instance 8	-	-	24	50	50	53
Instance 9	3563	132	17	17	14	14
Instance 10	-	-	-	-	-	156
Instance 11	-	1288	557	209	49	5
Instance 12	-	-	-	-	-	34

Table 5. Percentage of changes saved per level and total time (minutes) saved.

Instance	Level 1	Level 2	Level 3	Level 4	Time saved (minutes)
1	10 %	5 %	1 %	1 %	192
2	17 %	9 %	2 %	2 %	242
3	18 %	1 %	0 %	0 %	118
4	24 %	13 %	11 %	11 %	314
5	21 %	18 %	5 %	5 %	723
6	27 %	17 %	7 %	5 %	1093
7	59 %	52 %	47 %	46 %	3224
8	48 %	33 %	33 %	33 %	2178
9	27 %	23 %	12 %	12 %	1109
10	22 %	16 %	9 %	7 %	1164
11	24 %	11 %	7 %	6 %	410
12	28 %	17 %	10 %	9 %	699

in more than half of the tool lists. However, to add more tools, the company would need a larger tool inventory.

4. CONCLUSION

We presented an integer program to solve the problem of tool positioning for turrets-based CNC machines for which we do not know the production sequence. This corresponds to a huge part of the machining business right now. The integer program presented allows to find the optimal solution in a reasonable time for the 12 industrial instances of the problem. Solving this problem optimally allows the company to save approximately 191 hours of set up during the year which increases its productivity. The CNC machines being the bottleneck of the workshop, this should have a major impact on productivity and profitability. Moreover, by reducing the number of changes, we standardize the tool lists and reduce the number of manipulations that can cause errors.

ACKNOWLEDGEMENTS

The authors would like to thank Stéphane Agnard, Joël Lessard, Mathieu Béliveau, and Yves Proteau from APN inc. Funding was also provided by CRIQ and NSERC.

REFERENCES

Adjiashvili, D., Bosio, S., and Zemmer, K. (2015). Minimizing the number of switch instances on a flexible machine in polynomial time. *Operations Research Letters*, 43(3), 317–322.

Amaya, J.E., Cotta, C., and Fernández, A.J. (2008). *A Memetic Algorithm for the Tool Switching Problem*, 190–202. Springer Berlin Heidelberg, Berlin, Heidelberg.

Beezo, A.C., Cordeau, J.F., Laporte, G., and Yanasse, H.H. (2017). Scheduling identical parallel machines with tooling constraints. *European Journal of Operational Research*, 257(3), 834 – 844.

Calmels, D. (2018). The job sequencing and tool switching problem: state-of-the-art literature review, classification, and trends. *International Journal of Production Research*, 1–21.

Catanzaro, D., Gouveia, L., and Labbé, M. (2015). Improved integer linear programming formulations for the job sequencing and tool switching problem. *European journal of operational research*, 244(3), 766–777.

Crama, Y., Kolen, A.W.J., Oerlemans, A.G., and Spieksma, F.C.R. (1994). Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6(1), 33–54.

Crama, Y., Moonen, L.S., Spieksma, F.C., and Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, 182(2), 952–957.

Hirvikorpi, M., Knuutila, T., Leipälä, T., and Nevalainen, O.S. (2007). Job scheduling and management of wearing tools with stochastic tool lifetimes. *International Journal of Flexible Manufacturing Systems*, 19(4), 443–462.

Kayayama, H. (1994). FMS tool change schemes and their characteristics. *Computers & Industrial Engineering*, 27(1), 75–80.

Laporte, G., Salazar-Gonzalez, J.J., and Semet, F. (2004). Exact algorithms for the job sequencing and tool switching problem. *IIE Transactions*, 36(1), 37–45.

Mauergauz, Y. (2017). Job and tool group scheduling for a machining center. *International Journal of Management Science and Engineering Management*, 12(4), 280–287.

Sarmadi, H. and Gholami, S. (2011). Modeling of tool switching problem in a flexible manufacturing cell: with two or more machines. In *International Conference on Mechanical and Electrical Technology, 3rd,(ICMET-China 2011), Volumes 1–3*. ASME Press.

Zhou, B.H., Xi, L.F., and Cao, Y.S. (2005). A beam-search-based algorithm for the tool switching problem on a flexible machine. *The International Journal of Advanced Manufacturing Technology*, 25(9), 876–882.