Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 17 (2018) 968–975

Procedia MANUFACTURING

www.elsevier.com/locate/procedia

28th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2018), June 11-14, 2018, Columbus, OH, USA

Efficient tool loading heuristic for machines with modular feeders.

Csaba Raduly-Baka[a,*], Juha Mäkilä[a], Mika Johnsson[b], Olli S. Nevalainen[a]

[a] *University of Turku, Finland*
[b] *Mentor Graphics A Siemens Business, Finland*

Abstract

Numerically controlled placement machines are often equipped with modular feeder units. These allow the group loading of a large number of component reels in one step. While the loading of individual component reels has been extensively studied in the past, there has not been any algorithmic solutions published for modular feeder loading. The problem in known to be NP-hard even for a fixed job sequence. A mathematical $0-1$ formulation, a lower bound on the optimal result, and an efficient heuristics will be given in the present study. The heuristic is evaluated against the lower bound, and the ratio of the heuristic result and lower bound value is reported. We found that for randomly generated problems, this ratio is between 2 and 4, while for some of the large industrial problems we obtain a ratio between 6 and 12.

Keywords: Printed Circuit Board; Component placement; Set-up optimization; Changeable feeder units; Control of assembly process; Tool loading

1. Introduction

In this paper we study the Modular Tool Loading (MTL) problem, a machine control problem of managing component feeder set-ups in the assembling of printed circuit boards. The problem setting consists of a fixed sequence of PCB assembly jobs, and our task is to minimize the unproductive time caused by component feeder changes. The novelty of our problem setting is in the current tactic of organizing the feeder set-ups. Unlike in most of the previous studies, it is supposed here that the feeder unit consists of a number (say at most four) of changeable *feeder modules* (also called *feeder carriages*). Each module has a fixed capacity of (say 20 to 40) storage slots for the actual component reels. Replacing a whole feeder module is a simple and relatively fast manual operation, whereas changing several individual component reels is more time-consuming.

[*] Corresponding author. Tel.: +358451167977.
E-mail address: csraba@utu.fi

10.1016/j.promfg.2018.10.108

2. Literature review

The MTL problem is related to previously studied tooling problems. In the Tool Replacement Problem (TRP, also called Tool Switching Problem, TSwP, see [8]) of metal working industry there is a flexible machine with a tool magazine (in our case feeder unit) of limited capacity, and each job uses a number of tools (components) kept in the magazine. The tools are of the same size (consume one slot in the feeder unit), and the total number of tools (needed by all jobs) exceeds the magazine capacity, but the tools that each individual job requires fit in the magazine at a time.

The objective in TRP is to find a minimum number of tool swaps required to manufacture a given set of jobs. In the case of a fixed job sequence and equal tool sizes [18] and [6] show that this can be solved optimally by the Keep Tool Needed Soonest (KTNS) policy in $O(M \cdot N)$ time. In case of tool dependent loading cost, the problem is solved optimally by [13], using a min-cost flow formulation.

If the tool sizes are not uniform, the problem becomes NP-hard even for a fixed job sequence, as shown by [7]. A number of heuristics have been proposed for these cases by [12], [19], [14]. The problem of loading tools in groups (by using a tool transporter) has been discussed by [17]. The tool transporting problem differs from the modular feeder loading problem; in the former problem the feeder magazine consists of a single module (unit) and the location of component reels does not restrict the future component reel loading decisions.

The TRP becomes harder if the job sequence is allowed to change. [6] show that finding the optimum job sequence for uniform tool sizes and uniform loading costs is NP-hard, even if the online capacity of the tool magazine is $C = 2$. Several heuristics have been proposed for the problem, see [6], [10] [19], [16]. The approximation ratio for some of these solutions was discussed in [9]. Extensions of local search have been applied by [20, 11, 1, 21]. Variations of the sequencing problem are addressed by [4, 5, 3].

The problem of tool loading in the context of a modular magazine has been largely ignored by the literature. The MTL problem was introduced in [15], there the computational complexity of the general formulation and its special cases were studied. The problem was shown to be NP-hard even for a fixed job sequence, unless both the number of on-line modules and their capacity are fixed. For the case of fixed magazine capacity, [15] present a dynamic programming algorithm to solve the problem optimally. However, it is noted that this method cannot be used in practice even for small problem instances, due to the large exponent involved in size of the dynamic program table. Practical solution of the problem remains therefore open. In this study, we attempt to answer the practical aspect of the MTL problem. Namely, can practical problem instances be solved by the means of integer programs, and if not (which turns out to be the case), what kind of heuristics are suitable to find feasible and good solutions.

3. Modular tool loading problem

Suppose that a list J of N PCB assembly jobs is given, and the jobs must be processed in the order given by J. An assembly job consists of a batch of PCBs of the same type. The jobs are processed in a fixed order by a single component placement machine, and each job i ($i \in [1...N]$) presupposes the insertion of a set S_i components of different types. The components of a job are supplied by the means of a feeder unit attached to the machine. The feeder unit is capable of holding all the components of a job at a time, but the total on-line capacity C of the feeder unit is less than the total demand for all M different component types of all jobs in J. The feeder unit is of modular type, which means that a number p (typically 2 to 6) of changeable on-line modules (racks) can be installed into the machine. In addition, there are $q > 0$ off-line modules placed at the vicinity of the machine and their setup can be changed while the production of the previous job is going on, with no impact on the production delay. The capacity of all modules is the same, c feeder slots (typically 20-40). Here we assume, that *component reels* (also called *component tape holders*) are of uniform size, that is, each reel consumes one feeder slot.

The number of holders for electronic components is one factor which matters when planning the management of the feeder unit. Factories commonly want to restrict the number of the component tape holders due to their costs. However duplicate holders may be advantageous in this context; for heavily used component types one may benefit from keeping two reels for the same component type at a moment in the feeder unit, and one may save on-line feeder reorganizations due to the duplicates used for off-line settings.

We suppose that the production line is stopped each time a job (i.e. PCB type) is changed. This is done for changing the NC-program that controls the machine and for performing the necessary material handling tasks. However, one should avoid rearrangements of the component reels in the on-line feeder modules because they greatly delay the restarts. Reel removal costs are not accounted for separately, but as part of the reel change costs. A reel change instance consists of a (possible) reel removal and the insertion of a new reel. A reel change step is also referred to as a *reel switch.*

4. Integer linear programming formulation

For the integer programming formulation of the MTL problem the following notations are introduced:
Program parameters

Z	- the total cost (time) of processing the job sequence (objective function).
N	- number of jobs to be processed.
M	- total number of component reels of different types used in the jobs.
p	- number of on-line modules (in primary storage).
q	- number of off-line modules (in secondary storage).
T	- total number of modules ($T = p + q$).
c	- capacity of a module.
t_f	- cost of changing one component reel in the feeder module.
t_m	- cost for interchanging one on-line module with one off-line module.
s_i	- maximum number of component reels available for each component type i. Typically this is 1 for each component reel, but frequently used reels can be duplicated for better performance.
r_{ji}	- 1 if job j requires component reel i, 0 otherwise.

Decision variables

z_{kj}	- 1 if module k is on-line when processing job j, 0 otherwise.
w_{kji}	- 1 if module k contains component reel i when processing job j, 0 otherwise.
v_{kji}	- 1 if module k is on-line while processing job j and contains component reel i, 0 otherwise. v_{kji} is used to linearize $v_{kji} = w_{kji} \cdot z_{kj}$.
x_{kji}	- component reel i is switched in module k just prior to the starting of job j.
y_{kj}	- is 1 if module k must be loaded into the feeder unit just prior to the starting of job j (that is, module k is off-line while processing job $j-1$, and it is needed on-line for job j. y_{kj} is similar to z_{kj}, except here we count only module loading (not status). We do not count module removals (on-line to off-line), because those are included in the module loading costs.

Minimize:

$$Z = t_f \cdot \sum_{k=1}^{T} \sum_{j=1}^{N} \sum_{i=1}^{M} x_{kji} + t_m \cdot \sum_{k=1}^{T} \sum_{j=1}^{N} y_{kj} \tag{1}$$

Subject to:

$$\sum_{k=1}^{T} v_{kji} \geq r_{ji}, \text{ for all } j \in \{1, ..., N\}, i \in \{1, ..., M\} \tag{2}$$

$$v_{kji} \leq z_{kj}, \text{ for all } k \in \{1, ..., T\}, j \in \{1, ..., N\}, i \in \{1, ..., M\} \tag{3}$$

$$v_{kji} \leq w_{kji}, \text{ for all } k \in \{1, ..., T\}, j \in \{1, ..., N\}, i \in \{1, ..., M\} \tag{4}$$

$$\sum_{i=1}^{M} w_{kji} \leq c, \text{ for all } k \in \{1, ..., T\}, j \in \{1, ..., N\} \tag{5}$$

$$\sum_{k=1}^{T} w_{kji} \leq s_i, \text{ for all } j \in \{1, ..., N\}, i \in \{1, ..., M\} \tag{6}$$

$$x_{kji} \geq w_{kji} - w_{kj-1i} - (1 - z_{kj-1}) - (1 - z_{kj}) \text{ for all } k \in \{1, ..., T\}, j \in \{1, ..., N-1\}, i \in \{1, ..., M\} \tag{7}$$

$$x_{kji} \geq w_{kj-1i} - w_{kji} - (1 - z_{kj-1}) - z_{kj} \text{ for all } k \in \{1, ..., T\}, j \in \{1, ..., N-1\}, i \in \{1, ..., M\} \tag{8}$$

$$y_{kj} \geq z_{kj} - z_{kj-1} \text{ for all } k \in \{1, ..., T\}, j \in \{1, ..., N-1\} \tag{9}$$

$$\sum_{k=1}^{T} z_{kj} \leq p \text{ for all } j \in \{1, ..., N\} \tag{10}$$

$$z_{kj}, w_{kji}, v_{kji}, x_{kji}, y_j \in \{0, 1\} \tag{11}$$

In the IP formulation we refer to job 0 (by $z_{k(j-1)}$ or $w_{k(j-1)i}$ when $j = 1$), that actually does not exist. Practically job 0 defines the state of the feeder unit prior to the starting of the first job. For example (in rolling horizon production planning) the feeder unit may already contain certain modules and some component reels may be loaded into these. If the feeder unit and all modules are empty prior to the starting of the first job, then $z_{k0} = 0$ and $w_{k0i} = 0$ for all modules k and component reels i.

For each job the required component reel types must be in on-line modules. In (2) v_{kji} is 1 if the module k is on-line and contains component reel i when processing job j. In (3) and (4) component reel i at a job j is active if the module containing the component reel is on-line when processing j. By constraint (5) there can be at most c component reels in any module k at any job j. By constraint (6) in any job j there can be at most s_i copies of each component reel in modules (on-line or off-line).

The cost of switching a component reel i at job j in a module k is calculated in variable x_{kji}. The cost is nonzero if module k is on-line or a reel is not available off-line to insert it into an off-line module. Otherwise the cost is zero (switching in an off-line module is free). Constraint (7) counts the number of reel insertions into the on-line modules.

A reel switch must be counted also when a reel is loaded into an off-line module from an on-line module. This can occur only while the production is stopped, thus incurring a cost. These swaps are counted by the constraint (8). Module loading y_{kj} is counted by (9) as the difference between a module's off-line and on-line status. For any given job j, there are at most p on-line modules by constraint (10). These variables are $0/1$ integers (11).

The above integer program was too slow to solve even small problem instances, when one or more off-line modules are available. If there are no off-line modules (as in the classic tool switching problem), the IP model did provide the optimal result efficiently. We tried to implement a Lagrangian relaxation, but obtained invalid results (i.e. carriages with no tools).

5. Greedy heuristics

In this section a greedy heuristic is introduced for creating feasible module assignments. In the case of no off-line modules, or with excessively expensive module swaps, the problem can be solved optimally by the means of the KTNS method. On the other hand, when at least one off-line module is available, and exchanging a module is less costly that exchanging c individual component reels, the problem becomes hard, assuming arbitrary capacity c (as shown in [15]). This is mainly because of the necessity to decide between exchanges of individual component reels or modules.

A *group* of jobs is a sequence of consecutive jobs of J such that all their component reels fit into the online feeder modules at a time. In the KTNS scheme (without modules), new component reels can be loaded between any two jobs, as needed (per an optimal heuristic). When solving the MTL-problem, loading new component reels is restricted to occur between consecutive groups. This allows fast loading of several components by one or more module loads (up to 40 components per module).

5.1. Component Weights

The main (and rather involved) problem here is to decide which components should be loaded individually, and which ones will be accumulated into offline modules and loaded with a single module swap. Swapping modules also means that one or more modules are unloaded from the online feeder unit of the machine. Intuitively, components that are used in a long sequence of consecutive jobs are more likely to be needed on-line than the ones that are used less or sporadically.

The *component type weight* encodes this intuition, by counting the number of jobs that uses the component and normalizing this count with the number of gaps in the usage. Here, a gap refers to a consecutive set of jobs that do not use the component type. Let d_i be the number of gaps of component reel i, and e_i be the number of jobs using component i. Then we define the *component type weight* of reel i as the quantity: $u_i = e_i/(d_i + 1)$.

The component type weights can be calculated for the whole set of jobs, or only for a subsequence of jobs, depending on the type of heuristic employed. The component type weights can be calculated in $O(M \cdot N)$ time, as a preprocessing step.

5.2. Weighted Max Algorithm (WMA)

In the *weighted max algorithm* (WMA) the component to module assignments are done in a decreasing order of the component type weights u_i (for equal weights, ties are broken randomly). The u_i weights are calculated separately for each job group, thus reflecting the current state of component usage. The leftover space of a module is filled up by using a subset of the component reels of job $k+1$ that does not totally fit into the first group. For this, the components of job $k+1$ with the largest weights are found. This action results in module setups that contain components with similar weights.

The component type weights are also used in the group transition phase. New component reels are assigned to modules that are partially filled with components from the previous group. Further, the component reels with the largest weight are selected from job $k+1$ to complete the set for a new module. The component type weights are also used to assign the component reels to partially filled modules, so that components of similar weights end up in the same module.

6. Lower bound.

Since there are no modular component reel loading heuristics proposed yet in the literature, a lower bound on the optimal solution will be introduced here and used to evaluate the effectiveness of the proposed heuristics. Let K denote the number of component reel switches calculated by the KTNS policy [18], for a job sequence J. From [18] it is known that to process the jobs in J in the given order, at least K component reel swaps must occur in the feeder magazine.

Likewise, an optimal modular component reel loading policy must also swap K tools (or more, to take good use of the modules). Due to the use of moveable reel modules, the costs of the two different models differ; Our working assumption is that swapping one module costs significantly less than swapping a corresponding amount of reels $t_m < c \cdot t_f$. The ideal modular component reel loading policy would then be that all the component reel swaps are done by module swaps, and each module is fully used. This means at least $\lfloor \frac{K}{c} \rfloor$ module swaps, because each swap of a module would cause c reel switches.

Lemma 6.1. *Suppose that $t_m < c \cdot t_f$ and K is the number of tool switches obtained with the KTNS rule. Then a lower bound for the total change costs Z of the MTL problem is $t_m \cdot \lfloor \frac{K}{c} \rfloor$.*

Proof. By contradiction, assume that there is an optimal solution, that exchanges a modules and b component reels, and has a smaller cost than $\lfloor \frac{K}{c} \rfloor$:

The optimal component reel change cost is then: $t_m \cdot a + t_f \cdot b < t_m \cdot \lfloor \frac{K}{c} \rfloor < t_m \cdot \frac{K}{c}$

Multiplying by c/t_m, we get: $c \cdot a + \frac{t_f \cdot c}{t_m} \cdot b < K$

Our working assumption is that $t_m < c \cdot t_f$ (otherwise simple module swaps by the KTNS would suffice), therefore: $\frac{t_f \cdot c}{t_m} > 1$. Then $c \cdot a + b < c \cdot a + \frac{t_f \cdot c}{t_m} \cdot b < K$

The number of component reel swaps in the optimal solution is then $c \cdot a + b < K$, which is a contradiction, since the given job sequence cannot be processed by less than K component reel swaps. $\square$

Therefore, the value $t_m \cdot \lfloor \frac{K}{c} \rfloor$ will always be less (or equal) than the cost of the optimal component reel loading policy. We use this value as a lower bound when evaluating our heuristic.

7. Results

The heuristic WMA is evaluated on a data set of 341 PCB instances (denoted D341), that are extracted from real jobs used in the industry. The jobs include a total of 410 different component types. The processing order of PCBs was fixed. Tests on the data set were performed using $p = 4$ on-line modules, each having a capacity of 30 component reels. This allows 120 on-line component reels at a time, which is sufficient to process the largest jobs from the data set, but insufficient to process all jobs without component reel swaps. For the D341 test set, the number of off-line modules q was varied from 0 to 4 in different tests, where $q = 0$ represents the classical tool loading problem (KTNS).

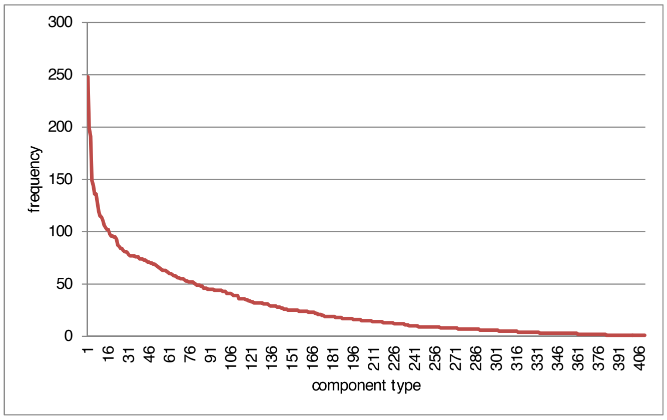

Figure 1: Frequencies of using components in different PCB types (D341).

In addition to the industrial data set, a number of artificial problem instances were generated in a similar way as in [6]. Since the present problem setting considers only a fixed job sequence, the tests were made on larger data sets (100-300 components with 100-200 jobs).

The frequency of using a particular component type in different PCB types is very skewed for D341, see Fig. 1. The maximum frequency is 248, and there are several component types that are only rarely used.

7.1. *Results with the data sets.*

In the following tests we assume that $t_m = 10$ and $t_f = 4$ (typical in practice). The tests were run on both the randomly generated instances and the industrial data sets. The random instances are generated as in [6], with the difference that here, we consider larger instances. A random instance type is denoted by $(M, N, min, max, c, p, q)$, where *min* and *max* are the minimum and maximum number of components in a job. For test with the random data sets, a hundred instances of each type were generated of each type and the results were averaged.

The tests were run on a computer equipped with a 2.3GHz Intel processor. The time it takes to run the module loading heuristics was under one second. Running all the random test instances (100 of each type) took about half a minute, therefore we omit the time from the results.

The results are reported in Table 1. The results are compared to the lower bound described earlier. The sixth row contains the ratio between the results of WMA and the lower bound.

We call an instance sparse if $|J_i|/(p \cdot c)$ is small (cases $2, 3, 6, 7, 12, 15, 17$), and dense if it is large (cases $4, 5, 8, 9, 10, 13, 16$). In sparse instances a job typically occupies a smaller part of the online feeder capacity, while in dense instances a job occupies most of the capacity, causing more frequent tool swaps. The random data sets contain both sparse and dense instances. We observe that the ratio to the lower bound is smaller for dense instances, when the data sets are larger. That is, WMA performs well for dense instances, but not so well for sparse instances (larger ratio). One reason for this can be that the lower bound may be farther from the optimum for sparse instances. Also the effect of additional off-line modules is clearly visible from the results of the runs. In test cases $1-5$, the number of off-line modules q was increased from 0 (resulting in KTNS) to 4.

7.2. *Number of duplicate reels.*

For the job list D341 with $p = 4$, $q = 2$, $t_m = 10$, $t_f = 4$ (typical values in practice), and module capacity 30, we varied the number of component types for which we have allocated a duplicate reel. It is supposed here that the manufacturer has an access to a number of extra reels and wants to allocate them to some of the component types of the current production program. In what follows, these reels are reserved to the most frequently used component types, see Fig. 1.

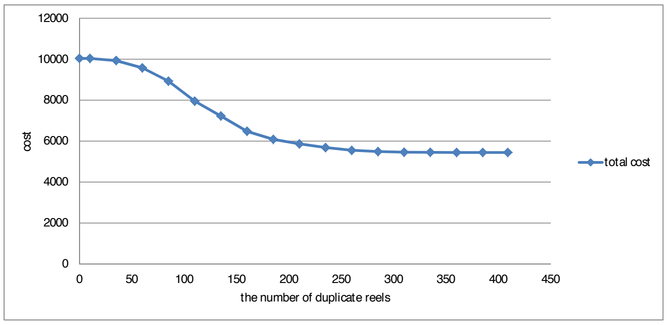

Figure 2: WMA: total job change cost of D341 as a function of the number of duplicate reels for WMA, with $p = 4$, $q = 2$, $t_m/t_f = 10/4$.

Figure 2 shows the total job change costs as a function of the number of duplicate reels for WMA. The effect of duplication is remarkable for this data. When all components are duplicated, the costs drop to nearly half of the original. On the other hand the slope of the graph approaches rapidly zero when more than half of the components have been duplicated.

8. Conclusions

In this work we discussed the tooling problem of a component placement machine equipped with a modular feeder unit and moveable component reel modules. The problem is an extension of the classical tool switching problem, in that it allows the tools to be changed in groups (a module at a time), thus reducing the tool switching cost. This is a practical problem that presently appears in various manufacturing environments.

Table 1: Results compared to the lower bound.

Case	Data Set	KTNS	WMA	LB	Ratio
1	$(D341, 30, 4, 0)$	2862.0	2862.0	238.5	12.0
2	$(D341, 30, 4, 1)$	2701.0	1736.5	225.1	7.7
3	$(D341, 30, 4, 2)$	2735.0	1410.5	227.9	6.2
4	$(D341, 30, 4, 3)$	2840.0	864.5	236.7	3.7
5	$(D341, 30, 4, 4)$	2831.0	754.5	235.9	3.2
6	(160, 100, 10, 20, 10, 4, 4)	748.0	414.5	187.0	2.2
7	(160, 100, 10, 20, 20, 2, 2)	674.0	185.0	84.2	2.2
8	(160, 100, 20, 40, 10, 4, 4)	1781.0	932.0	445.2	2.1
9	(160, 100, 20, 40, 20, 2, 2)	1797.0	480.0	224.6	2.1
10	(160, 200, 10, 20, 10, 4, 4)	1421.0	808.5	355.2	2.3
11	(160, 200, 10, 20, 20, 2, 2)	1342.0	375.0	167.8	2.2
12	(160, 200, 20, 40, 10, 4, 4)	3654.0	1881.5	913.5	2.1
13	(160, 200, 20, 40, 20, 2, 2)	3720.0	989.0	465.0	2.1
14	(300, 200, 30, 60, 20, 4, 4)	4445.0	1394.5	555.6	2.5
15	(300, 200, 30, 60, 30, 4, 4)	2993.0	730.0	249.4	2.9
16	(300, 200, 30, 60, 40, 4, 4)	1984.0	460.0	124.0	3.7
17	(300, 200, 60, 80, 20, 4, 4)	8825.0	2000.0	1103.1	1.8
18	(300, 200, 60, 80, 30, 4, 4)	5237.0	1445.0	436.4	3.3
19	(300, 200, 60, 80, 40, 4, 4)	3394.0	870.0	212.1	4.1
20	(300, 200, 40, 120, 30, 4, 4)	7206.0	1672.5	600.5	2.8
21	(300, 200, 40, 120, 40, 4, 4)	4068.0	1009.0	254.2	4.0

We introduced a mathematical formulation of the problem, in the form of a $0-1$ integer program. The $0-1$ program was tested on a solver, but only very small (non-practical) instances could be solved in this. Even for these small instances (10 jobs with 20 tools and 2 modules), it took several hours for the solver to find an optimal solution. This motivated the search for heuristic solutions that give feasible tool switching and module assignments, in reasonable time.

We introduced a heuristic method to solve the module assignment problem, and evaluated the method on generated random problems and on practical problem instances provided by industry. The results show that using modular feeders can significantly reduce tool loading costs. As a relevant topic of further studies, we plan to find effective ways to identify a sub-set of component reels which can be fixed into modules during a larger period of production.

References

[1] Amaya J.E., Cotta C., Fernandez A. (2008). A Memetic Algorithm for the Tool Switching Problem. *Hybrid metaheuristics, Springer Berlin Heidelberg*, pp. 190-202.

[2] Ayob M., Kendall K. (2008). A survey of surface mount machine optimization: Machine classification. *European Journal of Operational Research*, Vol. 186, pp. 893-914.

[3] Beez£o, 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.

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

[5] Chaves, A. A., Lorena, L. A. N., Senne, E. L. F., and Resende, M. G. (2016). Hybrid method with CS and BRKGA applied to the minimization of tool switches problem. *Computers & Operations Research*, Vol. 67, 174-183.

[6] Crama Y., Kolen A.W.J., Oerlemans A.G. (1994). Minimizing the number of tool switches on a flexible machine. *The International Journal of Flexible Manufacturing Systems*, Vol. 6, pp. 33-54.

[7] Crama, Y., Moonen. L.S., Spieskma, F.C.R., Talloen, E. (2007). The tool switching problem revisited. *European Journal of Operational Research*, Vol. 182, pp. 952-957.

[8] Crama Y., van de Klundert J., Spieskma F.C.R. (2002). Production planning problems in printed circuit board assembly. *Discrete Applied Mathematics*, Vol. 123, No 1-3, pp. 339-361.

[9] Crama, Y., van de Klundert, J. (1999). Worst-case performance of approximation algorithms for tool management problems. *Naval research logistics*, Vol. 46, pp. 445-462.

[10] Djellab H., Djellab K., Gourgand M. (2000). A new heuristic based on a hypergraph representation for the tool switching problem. *International Journal on Production Economics*, Vol. 64, pp. 165-176.

[11] Konak A., Konak S.K. (2007). An Ant Colony Optimization Approach to the Minimum Tool Switching Instant Problem in Flexible Manufacturing System. *IEEE Symposium on Computational Intelligence in Scheduling*.

[12] Matzliach B., Tzur M. (2000). Storage Management of Items in Two Levels of Availability. *European Journal of Operational Research*, Vol 121, pp. 363-379.

[13] Privault C., Finke G., (1995). Modeling a tool switching problem on a single NC-machine. *Journal of Intelligent Manufacturing*, Vol. 6, pp. 87-94.

[14] Raduly-Baka Cs., Knuutila T., Nevalainen O. (2005). Minimizing the Number of Tool Switches with Tools of Different Sizes. *5th International Conference on Technology and Automation*.

[15] Raduly-Baka Cs., Nevalainen O.S. (2015). The modular tool switching problem. *European Journal of Operations Research*, Vol. 242, pp. 100-106.

[16] Salonen, K., Raduly-Baka, Cs., Nevalainen, O.S. (2006). A note on the tool switching problem for a flexible machine. *Computers & Industrial Engineering*, Vol. 50, pp. 458-465.

[17] Song C.Y., Hwang H. (2002). Optimal tooling policy for a tool switching problem of a flexible machine with automatic tool transporter. *International Journal of Production Research*, Vol. 40, pp. 873-883.

[18] Tang C.S., Denardo E.V. (1988). Models arising from a flexible manufacturing machine, Part I: Minimization of the number of tool switches. *Operations Research*, Vol. 36, No. 5, pp. 767-777.

[19] Tzur M., Altman A. (2004). Minimization of tool switches for a flexible manufacturing machine with slot assignment of different sizes. *IIE Transactions*, Vol 36, pp. 95-110.

[20] Zhou G.H., Xi L.F., Cao Y.S. (2005). A beam-search-based algorithm for the tool switching problem on a flexible machine. *International Journal of Advanced Manufacturing Technology*, Vol. 25, pp. 876-882.

[21] Zhou G.H., Xi L.F., Cao Y.S. (2009). Beam Search Algorithm for Minimizing Tool Switches on a Flexible Manufacturing System. *International Conference on Mathematical and Computational Methods of Science and Engineering*, Vol 9. pp. 68-72.