European Journal of Operational Research 297 (2022) 66–85

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

# An iterated local search procedure for the job sequencing and tool switching problem with non-identical parallel machines

Dorothea Calmels[1]

*School of Business, Economics and Information Systems, Chair of Information Systems (Information and IT Service Management), University of Passau, Innstrasse 43, 94032 Passau, Germany*

ARTICLE INFO

*Article history:*
Received 27 March 2020
Accepted 9 May 2021
Available online 21 May 2021

*Keywords:*
Flexible manufacturing systems
Scheduling
Iterated local search
Tool switching
Sequence-dependent setup times

ABSTRACT

In this paper, a new generalization of the uniform job sequencing and tool switching problem is presented. It considers non-identical parallel machines, which differ in tool magazine capacity and setup times. Applications of the problem can be found in the metal working or semiconductor manufacturing industries when the jobs require different tool sets for processing. The paper provides a Mixed Integer Programming formulation for the job sequencing and tool switching problem with machine-dependent processing and tool switching times under the consideration of three conflicting objectives (minimizing the total flowtime, minimizing the makespan and minimizing the total number of tool switches). Different efficient construction heuristics and Iterated Local Search methods are proposed and evaluated, using 640 new and publicly available instances. The results of the construction heuristics, the Iterated Local Search schemes and a Mixed Integer Programming Solver are discussed. The extensive computational experiments show the merit of the perturbation strategy in order to overcome local optima. It is shown that certain methods are more or less suitable depending on the objective function.



## 1. Introduction

Modern manufacturing systems must be flexible and adjustable in order to face product changes and complex product requirements. A manufacturing machine may be equipped with a tool magazine of limited capacity and may provide different functions such as milling, turning, or tool handling operations. In general, the number of tools that is required to process a variety of jobs on a machine is so large that it exceeds the tool magazine capacity, and hence tool switches - defined by removing a tool from its slot and inserting another tool in the free slot (Tang & Denardo, 1988) - become necessary. The tool switching time is particularly significant if it is relatively high compared to the processing time of a job and if the tools cannot be interchanged during job processing. In most real manufacturing systems, component or tool switching is the most time-consuming process (Van Hop & Nagarur, 2004), and must be avoided. The objective of the standard uniform job sequencing and tool switching problem (SSP) is, therefore, to find the best sequence of jobs for a given set of jobs that minimizes the number of tool switches on a single machine. This problem has been addressed in production and operations research literature for more than thirty years, with a sharp increase in the last five years (Calmels, 2019).

In the following, the job sequencing and tool switching problem with non-identical parallel machines (SSP-NPM) is addressed which is better adapted to challenges in the modern production environment. More precisely, the SSP-NPM is defined by three interdependent problems (see Fig. 1): (1) assigning a set of jobs to non-identical parallel machines, (2) sequencing the allocated jobs on the machines, and (3) arranging the tool loading. Non-identical machines imply that all the machines can execute the same operations but may have different magazine capacities and different tools, and the processing time of a job and the tool switching time need not be equal for all machines. Since the machines have a limited tool capacity, the number of tool switches and hence the tool setup time of a job are influenced by the job sequence. In contrast to the single machine problem, time-related objectives such as minimizing the makespan or total flowtime become more important when considering multi-machine problems. The minimization of the makespan is the most common objective in scheduling literature (Allahverdi, 2015) because it is an indicator of the system's throughput and a high throughput can be obtained by avoiding tool switching time. But this objective might lead to long waiting times for some of the jobs, thus a higher total flowtime. Therefore the total flowtime is considered to be more relevant for recent production systems (Liu & Reeves, 2001), because it leads to

*E-mail address:* dorothea.calmels@uni-passau.de
[1] http://www.wi.uni-passau.de/

https://doi.org/10.1016/j.ejor.2021.05.005
0377-2217/

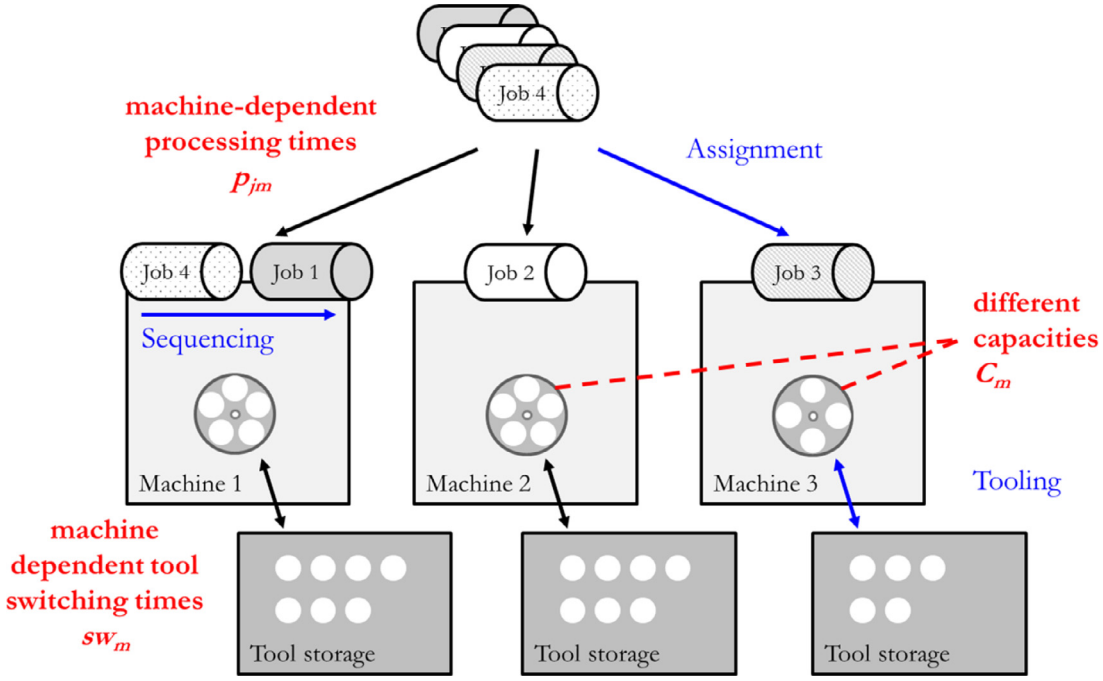


**Fig. 1.** Schematic layout of the problem environment.

the minimisation of in-process inventory and a fast turn-around of jobs (Rajendran, 1994).

The main contributions of this work include the formulation of a mixed integer linear (MIP) model that allows for time-related objectives, and the development of several heuristic solution methods. The mathematical formulation presents a more realistic and general approach to existing SSP literature. Moreover, the paper provides two new data sets for the SSP-NPM and examines the solution quality of different problem groups. The remainder of this article is as follows. Section 2 presents an overview of related work. The problem description and the mathematical formulation are provided in Section 3, followed by presenting the heuristic approaches in Section 4. The computational experiments and the analysis are discussed in Section 5. The final Section 6 concludes the study.

## 2. Related work

Scheduling parallel machines has been widely discussed in literature and most of the problems have been shown to be NP-complete for different objectives (Lenstra, Rinnooy Kan & Brucker, 1977). The interested reader is referred to Allahverdi (2015) and Kravchenko and Werner (2011). Although the SSP is directly related to machine scheduling, it developed its own research directions. The uniform SSP for a single machine with uniform tool size and equal and sequence-independent setup times was first introduced by Tang and Denardo (1988) who showed that the tool loading sub-problem can be solved to optimality by the '*keep tool needed soonest policy*' (KTNS) in polynomial time. The uniform SSP was proven to be NP-hard by Crama, Kolen, Oerlemans and Spieksma (1994). Therefore, researchers have mostly concentrated on heuristics or meta-heuristics. Only few articles consider exact algorithms or integer programming (IP) methods. A detailed review of the research on the SSP is provided by Calmels (2019).

Job sequencing with tool switches occurs in various industries (Crama, Moonen, Spieksma & Talloen, 2007; Shirazi & Frizelle, 2001), and related problems have been established in diverse areas. Analogies to the tool switching problem in the manufacturing industry occur in the electronics industry for sequencing printed circuit boards (PCBs) (Ghrayeb, Phojanamongkolkij & Finch, 2003; Hirvikorpi, Salonen, Knuutila, and Nevalainen (2006); Raduly-Baka & Nevalainen, 2015; Tzur & Altmann, 2004) or in computer systems for caching and paging or k-server problems (Djellab, Djellab & Gourgand, 2000; Ghiani, Grieco & Guerriero, 2007; Privault & Finke, 2000).

The SSP for multiple machines was first mentioned by Bard (1988) for tandem machines. Fathi and Barnette (2002) present heuristics for the SSP with identical parallel machines and conclude that the problem is NP-hard. A mixed-integer non-linear program for machines with identical working conditions but different magazine capacities is addressed by Sarmadi and Gholami (2011). Beezão, Cordeau, Laporte and Yanasse (2017) extend the IP formulations of Tang and Denardo (1988) and Laporte, Salazar-González and Semet (2004) to identical parallel machines with given processing times and the objective of minimizing makespan. Özpeynirci, Gökgür and Hnich (2016) and Gökgür, Hnich and Özpeynirci (2018) investigate the SSP for unrelated parallel machines without considering the magazine capacity. They include a limited amount of tool copies to minimize the makespan. A different approach with re-entrance of jobs, release times and due dates was recently presented by Dang, van Diessen, Martagan and Adan (2021) for the identical parallel machine problem with tooling constraints. So far, only one article addresses the SSP-NPM with time aspects and few related studies exist for sequencing PCBs. Ghrayeb et al. (2003) and Van Hop and Nagarur (2004) consider the scheduling of printed circuit packs on multiple parallel sequencers that are equipped with a number of dispensing heads that have to be loaded with the required input tapes. Recently, Calmels, Rajendran and Ziegler (2019) presented several sequencing policies for the SSP-NPM. It is shown that the SSP-NPM for different objective functions requires different strategies in order to ensure a good performance. Thus, the heuristics presented in Section 4 were adjusted to different objective functions.

## 3. Notation and problem formulation

A set of jobs $\mathcal{J} = \{1,\ldots, J\}$ has to be processed in a manufacturing environment with a set of non-identical parallel machines

**Table 1**
The mathematical notation used in the MIP model.

| | |
|---|---|
| Sets | |
| $\mathcal{J}$ | Set of all the jobs, $\mathcal{J}=\{1,\ldots,J\}$. |
| $\mathcal{M}$ | Set of all the machines, $\mathcal{M}=\{1,\ldots,M\}$. |
| $\mathcal{T}$ | Set of all the tools, $\mathcal{T}=\{1,\ldots,T\}$. |
| $\mathcal{J}_t$ | Set of jobs that require tool $t$, $\mathcal{J}_t \in \mathcal{J}$. |
| Indices | |
| $j$ | Job index, $j \in \mathcal{J}$. |
| $m$ | Machine index, $m \in \mathcal{M}$. |
| $t$ | Tool index, $t \in \mathcal{T}$. |
| $r$ | Position index, $r \in \mathcal{J}$. |
| Parameters | |
| $p_{jm}$ | Processing time of job j on machine $m$, $p_{jm} \geq 0$, $j \in \mathcal{J}$, $m \in \mathcal{M}$. |
| $sw_m$ | Switching Time per tool of machine $m$, $p_{jm} \geq 0$, $m \in \mathcal{M}$. |
| $C_m$ | Tool magazine capacity of machine $m$, $C_{\boldsymbol{m}} > 0$, $m \in \mathcal{M}$. |
| Variables | |
| $f_{jrm}$ | Completion time of job $j$ on machine $m$, when job $j$ is processed in the $r^{\text{th}}$ position on machine $m$, $j \in \mathcal{J}$, $r \in \mathcal{J}$, $m \in \mathcal{M}$. |
| $x_{jrm}$ | 1, if job $j$ is processed in the $r^{\text{th}}$ position on machine $m$, *else 0*, $j \in \mathcal{J}$, $r \in \mathcal{J}$, $m \in \mathcal{M}$. |
| $w_{trm}$ | 1, if tool $t$ is inserted in the tool magazine of machine $m$ exactly before the $r^{\text{th}}$ job is treated, else 0, $t \in \mathcal{T}$, $r \in \mathcal{J}$, $m \in \mathcal{M}$. |
| $v_{trm}$ | 1, if tool $t$ is present in machine $m$ during the processing of the job in the $r^{\text{th}}$ position, else 0, $t \in \mathcal{T}$, $r \in \mathcal{J}$, $m \in \mathcal{M}$. |
| *FMAX* | Makespan. |
| *TFT* | Total flowtime. |
| *TS* | Total number of tool switches. |

**Table 2**
Example with $J=6$, $T=9$ and $M=2$ with $C_1=4$, $C_2=3$, $sw_1=1$ and $sw_2=2$.

| Jobs | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Tools | 1 | 1 | 2 | 1 | 3 | 1 |
| | 4 | 3 | 6 | 5 | 5 | 2 |
| | 8 | 5 | 7 | 7 | 8 | 4 |
| | 9 | | 8 | 9 | | |
| $p_{j1}$ | 1 | 3 | 4 | 5 | 6 | 1 |
| $p_{j2}$ | | 4 | | | 3 | 1 |

$\mathcal{M}=\{1,\ldots,M\}$. Each job $j$ is available for processing at time zero, and no machine eligibility restrictions are imposed. Jobs are processed sequentially on each machine $m$. The processing time, $p_{jm}$, of a job $j$ on machine $m$ is machine-dependent and the machines may have different tool magazine capacities $C_m$. The time to switch one tool on a machine, $sw_m$, may be different for the various machines, i.e., depends on the machine and not on the tool. $\mathcal{T}=\{1,\ldots,T\}$ denotes the set of tools required for processing all jobs. It is assumed that each tool occupies only one slot in the tool magazine of a machine and at least one machine can hold all tools necessary for processing a job. Each job $j$ requires a subset of tools $\mathcal{T}_j$ . The processing of a job can only start if the required tools are present in the magazine of the machine on which the job $j$ is processed. $\mathcal{J}_t$ denotes the subset of jobs that require a certain tool $t \in \mathcal{T}$. The nature of the SSP requires that the tool magazine capacity $C_m$ of each machine $m$ does not provide enough tool slots for processing all jobs, so that tool switches may become necessary for two successive jobs. The initial loading does not count as a tool switch and tool switching cannot happen while the processing of a job is ongoing. The three presented conflicting objectives are: (a) minimize the makespan (*FMAX*), and (b) minimize total flowtime (*TFT*), and (c) minimizing the total number of tool switches (*TS*). Table 1 defines the notations of sets, parameters and decision variables used to formulate the MIP model.

An example of the SSP-NPM is given in Table 2. The example consists of 6 jobs and 2 machines with different capacities and switching times. An optimal solution for minimizing the total flowtime is presented in Fig. 2. The tool loading shows that although *job 6* requires only three tools, *tool 8* is kept in the free slot since it is required again for *job 3*. Note that minimizing *TS*, *TFT* or *FMAX* are conflicting objectives. Consequently, the optimal solutions of this example may be different from the displayed solution under the minimization of *TS* or *FMAX*.

The formulations of Beezão et al. (2017) and Calmels (2021) provide the basis for the following mathematical formulation with the additional consideration of non-identical parallel machines and sequence and machine-dependent setup times. The variables $x_{jrm}$ are equal to one if and only if job $j$ is processed in the $r^{\text{th}}$ position on machine $m$. Binary variables $v_{trm}$ are equal to one if and only if tool $t$ is present in machine $m$ during the processing of the job in the $r^{\text{th}}$ postion. Binary variables $w_{trm}$ are equal to one if and only if tool $t$ is inserted in the tool magazine of machine $m$ exactly before the $r^{\text{th}}$ job is treated. The continuous variables $f_{jrm}$ denote the completion time of job $j$ in the $r^{\text{th}}$ position on machine $m$. $G=\sum_{j=1}^{J}\max_m(p_{jm})+(J-1)\cdot\max_m(sw_m\cdot C_m)$ is a large constant.

$$\sum_{r\in\mathcal{J}}\sum_{m\in\mathcal{M}} x_{jrm}=1 \qquad \forall j\in\mathcal{J} \tag{1}$$

$$\sum_{j\in\mathcal{J}} x_{jrm}\leq 1 \qquad \forall r\in\mathcal{J},\forall m\in\mathcal{M} \tag{2}$$

$$\sum_{j\in\mathcal{J}} x_{jrm}\leq\sum_{j\in\mathcal{J}} x_{j(r-1)m} \qquad \forall r\in\mathcal{J}\backslash\{1\},\forall m\in\mathcal{M} \tag{3}$$

$$\sum_{j\in\mathcal{J}_t} x_{jrm}\leq v_{trm} \qquad \forall t\in\mathcal{T}, \qquad \forall r\in\mathcal{J},\forall m\in\mathcal{M} \tag{4}$$

$$\sum_{t\in\mathcal{T}} v_{trm}\leq C_m \qquad \forall r\in\mathcal{J},\forall m\in\mathcal{M} \tag{5}$$

$$v_{trm}-v_{t(r-1)m}\leq w_{trm} \qquad \forall t\in\mathcal{T},\forall r\in\mathcal{J}\backslash\{1\},\forall m\in\mathcal{M} \tag{6}$$

$$f_{j1m}=p_{jm}\cdot x_{j1m} \qquad \forall j\in\mathcal{J},\forall m\in\mathcal{M} \tag{7}$$

$$f_{jrm}\geq\sum_{i\in\mathcal{J},i\neq j} f_{i(r-1)m}+sw_m\sum_{t\in\mathcal{T}} w_{trm}+p_{jm}x_{jrm} -G\left(1-x_{jrm}\right) \qquad \forall j\in\mathcal{J},\forall r\in\mathcal{J}\backslash\{1\},\forall m\in\mathcal{M} \tag{8}$$

$$v_{trm},w_{trm}\in\{0,1\} \qquad \forall t\in\mathcal{T},\forall r\in\mathcal{J},\forall m\in\mathcal{M} \tag{9}$$

$$x_{jrm}\in\{0,1\} \qquad \forall j\in\mathcal{J},\forall r\in\mathcal{J},\forall m\in\mathcal{M}. \tag{10}$$

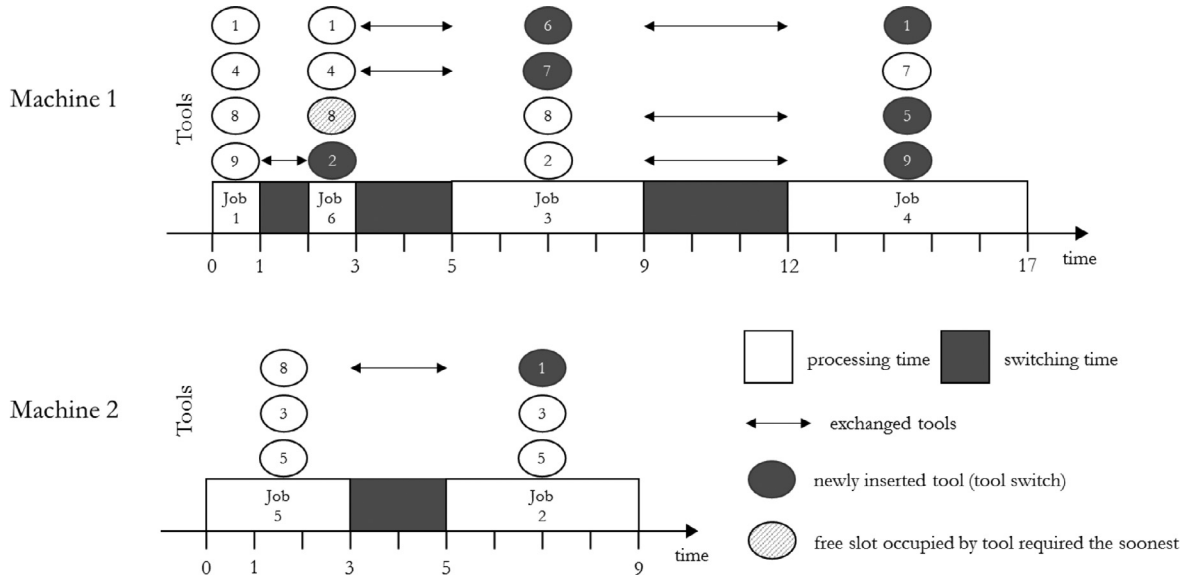


**Fig. 2.** Layout of the problem environment for optimizing TFT.

Constraints (1) denote that each job is assigned to exactly one machine and position. Constraints (2) impose that each position on machine is used by at most one job. Constraints (3) ensure that if there is a job in the $r^{\text{th}}$ position on machine $m$ then there must be a job in position $r-1$. Constraints (4) guarantee that the tools required for job $j$ are available in the tool magazine in the moment of job $j$'s processing. Constraints (5) ensure that the tool loading does not exceed the tool magazine capacity. Constraints (6) determine whether tool $t$ is inserted in the magazine of machine $m$ before the processing of a job in position $r$. The completion times of the initial jobs are calculated by the constraints (7) while the completion times of any other job are restricted by the constraints (8) as the sum of the completion time of the job processed in position $r-1$, the setup time and the processing time. Constraints (9) and (10) denote the integrality conditions.

The objective functions consider the minimization of the total flowtime (*TFT*) (11), minimization of the makespan (*FMAX*) (12) in combination with the additional constraints (13), and the minimization of the total number of tool switches (*TS*) (14).

$$\min TFT = \sum_{j\in\mathcal{J}}\sum_{r\in\mathcal{J}}\sum_{m\in\mathcal{M}} f_{jrm} \tag{11}$$

$$\min FMAX \tag{12}$$

$$FMAX \geq f_{jrm} \qquad \forall j \in \mathcal{J}, \forall r \in \mathcal{J}, \forall m \in \mathcal{M} \tag{13}$$

$$\min TS = \sum_{t\in\mathcal{T}}\sum_{r\in\mathcal{J}}\sum_{m\in\mathcal{M}} w_{trm} \tag{14}$$

## 4. Solution approaches for the ssp-npm

This section presents several solution procedures for the SSP-NPM. Three construction heuristics are introduced (Section 4.1), which build the initial solutions for the proposed iterated local search (ILS) (Section 4.2). The solution methods are evaluated by the extensive computational experiments reported in Section 5.

### 4.1. Construction heuristics

The performance of many exact and heuristic approaches depends on the quality of their initial solution. This section presents two rule-based heuristics and a Multi-Start-Random heuristic. Building a good initial solution is extremely important for the SSP-NPM because the optimal tool switches and the allocation of tools to free slots can only be determined for a given sequence. The purpose of the construction heuristics is to assign the jobs to machines and to sequence the jobs on the machines. The optimal tool loading can be determined using the KTNS-policy (Tang & Denardo, 1988): (1) no tool is inserted unless it is required by the next job, (2) if the number of tools needed for a next job is less than the magazine capacity, then tools needed the soonest by subsequent jobs are kept in the remaining free slots of the tool magazine and are not removed.

**IEACT (Iterated Earliest Artificial Completion Time)**: This heuristic is an altered version of the Iterated Earliest Completion Time heuristic (IECT) proposed by Pessoa and Andrade (2018) for the flowshop scheduling problem with delivery dates. The IEACT extends the SPT-based heuristic for the SSP-NPM of Calmels, Rajendran and Ziegler (2019) by adding the approximated tool setup time to the processing time. The calculation of the artificial completion time is an adaptation of the concept proposed by Liu and Reeves (2001) for the permutation flowshop scheduling problem with minimizing total flow time objective. A solution is built from scratch by iteratively appending a job at the end of a machine's partial solution. The machine with the minimum completion time of the last job is always selected for job assignment. The first job added is the job with the lowest processing time. After that, each unscheduled job is evaluated according to its earliest artificial completion time. The job with the lowest artificial completion time is added to the partial sequence. All ties are broken randomly. The KTNS-algorithm (Tang & Denardo, 1988) is used to subsequently determine the tool loading that minimize the number of tool switches for the given sequence. The pseudocode (Algorithm 1) is provided in the appendix.

**IGI (Iterated Greatest Intersection)**: Similar to the IEACT heuristic, this heuristic builds a solution by iteratively appending a job at the end of a machine's partial solution. The basic structure of this algorithm was proposed by Tang and Denardo (1988) and several variations exist for the single machine problem (Crama et al., 1994; Follonier, 1994). Following their ideas, a new iterative construction heuristic is proposed for the SSP-NPM. The machine with the minimum current artificial completion time of the last job is always selected for job assignment. The first job added is

the job that requires the maximum number of tools (with less than or equal to the magazine capacity of the selected machine) in order to prevent later tool switching. After that, each unscheduled job is evaluated according to the number of tools in common with the last job in sequence on the selected machine. The job with the currently greatest intersection of tools compared to the job last in sequence is added to the partial sequence. The artificial completion time of the job being added is updated afterwards. All ties are broken randomly. Note that the optimum tool loading can only be evaluated for a given sequence and the actual intersection between two jobs may actually differ from the number of tool switches in the optimal solution. The tool loading is subsequently optimized for the final constructed solution using KTNS (Tang & Denardo, 1988). The pseudocode (Algorithm 2) is provided in the appendix.

**MSR (Multi-Start-Random)**: The multi-start random heuristic generates a random permutation of all jobs. Afterwards, the jobs in the sequence of the generated permutation are allocated cyclically to the machines starting with the first machine. The idea behind is that for most real-world problems the machine load has to be balanced in order to reduce the completion time for all machines. If a job cannot be assigned to the machine under consideration, because the capacity restrictions would be violated, the job is randomly assigned to a machine with sufficient magazine slots. The tool loading is subsequently optimized by the KTNS-policy (Tang & Denardo, 1988) and the process restarts until the iteration limit is reached. The best solution for each objective is returned. The pseudocode (Algorithm 3) is provided in the appendix.

### 4.2. An iterated local search method (ILS)

Although construction heuristics can usually provide a good solution very quickly, an improvement of the solution quality may be obtained by applying local search algorithms. In this section, an iterated local search method is presented. ILS has been applied to a wide range of scheduling problems (Lourenço, Martin & Stützle, 2019) since it avoids the extensive tuning of parameters. For the general ILS algorithm and the review of some applications see, for example, Talbi (2009), and Lourenço et al. (2019). ILS applications to the SSP are provided by Bard (1988) and Paiva and Carvalho (2017) for the uniform single machine SSP, and by Hirvikorpi, Nevalainen and Knuutila (2006) under the consideration of tool wear. Algorithm 4 in the appendix shows the general ILS scheme. Starting with an initial solution, ILS iteratively applies intensification and diversification through local search and perturbation methods until a stopping criterion is reached. In the small-step phase, a swapping-based local search is performed on each machine. A large-step optimization is performed afterwards by transferring jobs between machines using the newly proposed perturbation methods.

#### 4.2.1. Local search with restart

An initial solution is constructed by one of the construction heuristics presented in Section 4.1. The local search method (see algorithm 5 in the appendix) further explores the neighbourhood of the initial solution in order to find a local minimum. Here, the neighbourhood of the initial solution is explored by an iterated swapping scheme. After swapping two jobs processed on a machine, the tool loading is optimized by the KTNS-procedure (Tang & Denardo, 1988) to obtain a new solution $\mathcal{S}'$. The new solution is only accepted if its objective value $F(\mathcal{S}')$ is better than the best-known objective value $F(\mathcal{S}^{best})$, hence if $F(\mathcal{S}') < F(\mathcal{S}^{best})$. In this case, the local search restarts on the machine where an improvement has been generated, otherwise the next pair of jobs is swapped on the machine being considered. All possible swaps on one machine are analysed before the swaps are performed on the next machine. This is repeated until all machines have been considered. The local search is limited to the neighbourhood of a solution without modifying the job assignment to the machines. Subsequently to the local search, a large step perturbation is performed by altering the job assignment by inserting jobs on different machines.

#### 4.2.2. Perturbation

Three different perturbation strategies are explored. The first *objective-specific perturbation* (**P.1**) is based on the general structure of the objective function. In order to address the difficulty that the *objective-specific perturbations* (**P.1**) may result in the algorithm being stuck in local optima, *randomized procedures* (**P.2**) are presented. The *combined procedure* (**P.3**) attempts to combine both the positive effects of the systematic objective-specific perturbation (**P.1**) and the positive effects of a random variation (**P.2**) of the solution under consideration. Ties are always broken randomly.

**Objective-specific perturbations (P.1):** The paper considers three different objectives of which two objectives are time-related (total flowtime and makespan) and one objective is related to tool switches. For this reason, two different perturbation procedures are presented to account for the conflicting nature of the different objective functions.

- Time-based perturbation (P.1$_{time}$):

The makespan and the total flowtime are influenced by many factors, among them the processing and setup time of each job. These objective values can only be decreased by decreasing the completion time of at least one job. The new perturbation concept is an insertion-based method. The pseudocode (algorithm 6) is provided in the appendix. Given a current solution, select the bottleneck machine $m_1$ with the highest completion time of the last job ( $f_{m1}$), and for which there exists a job that can be relocated to another machine $m_2$. The job $j_1 = [q^*]_{m1}$, in position $q^*$ on machine $m_1$, with the highest processing plus setup time cost is selected. Note that the job is only selected if it can be relocated to any of the other machines, thus the relocation does not violate the capacity restrictions. Job $j_1$ is reinserted on a different machine $m_2$ that must provide a sufficiently large tool capacity. Job $j_1$ is inserted on a different machine $m_2$ between two jobs such that the sum of processing time and approximated tool switching times is minimized over the different machines and insertion positions. Note that in the case of removing as well as inserting a job from the first or the last position, the calculations reduce because, as per definition, tool switches do not occur before processing the first job and after processing the last job on a machine.

- Tool-Switch-based perturbation (P.1$_{switch}$):

The perturbation for minimizing the number of tool switches is similar to the previous perturbation concept, but ignores time-related aspects. The pseudocode (algorithm 7) is provided in the appendix. The bottleneck machine $m_1$ with the highest total number of tool switches is selected, and for which there exists a job that can be relocated to another machine. $ts_j$ denotes the number of tool switches required immediately before processing job $j$. The job $j_1$ processed on $m_1$ with the largest number of tool switches required immediately before its processing is selected for reinsertion on another machine. Job $j_1$ is inserted on a different machine $m_2$ such that the number of tool switches before and after processing job $j_1$ on machine $m_2$ is minimized over the different machines and insertion positions. Fig. 3 illustrates the selection and insertion. Note that in this example, both machines have a total number of tool switches of 5. Job 6 on machine 2 is selected. Together with job 3, job 6 currently requires the most tool switches but can be allocated to another machine (machine 1). It is ran-

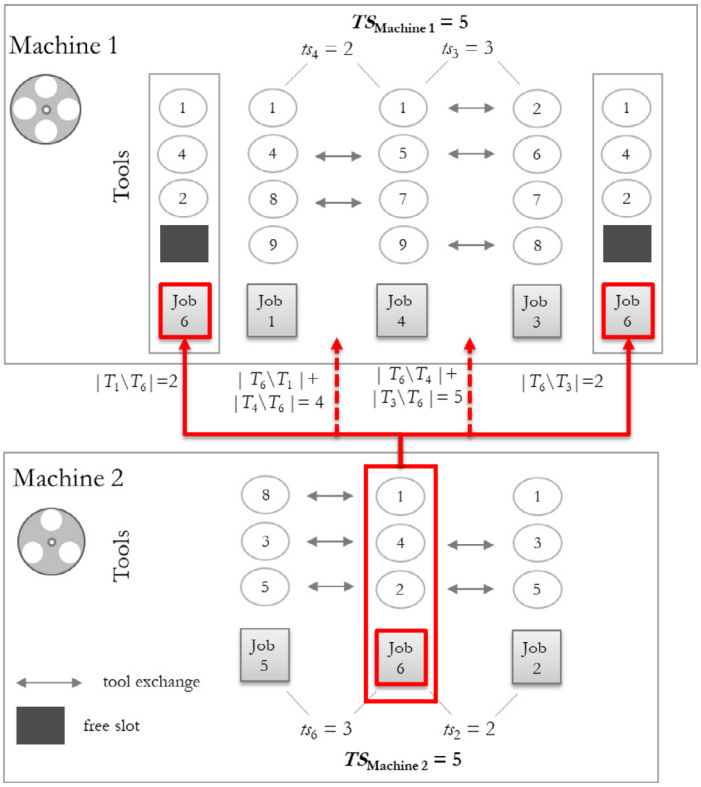


**Fig. 3.** Illustration of the removal and insertion phase for tool switching.

domly assigned to either position 1 or position 4 on machine 1 in the position with the least approximated tool switches.

**Random Perturbation (P.2):** In the random permutation, $n$ jobs of job set $\mathcal{J}$ are selected at random. The jobs are successively relocated to random positions in the sequence of a randomly selected machines with sufficient tool capacity. Based on the observations of Paiva and Carvalho (2017) $n$ is obtained by $n = \lfloor \beta \bullet J \rfloor$, with parameter $\beta$, $0 < \beta < 1$.

**Combined Perturbation (P.3):** This perturbation algorithms combines the objective-specific perturbation (P.1) and the random perturbations (P.2) in order to overcome local optima created by the systematic objective-specific perturbation. A random perturbation is performed with a fixed probability $\gamma$, with $0 < \gamma < 1$, after an iteration without improvement. For this, a random number (*rand*) is generated from a uniform distribution in the interval $0 < \gamma < 1$. If $rand < \gamma$, a random perturbation is performed, for example, $\gamma = 0.25$ would mean that a random perturbation is applied in average every fourth deterioration.

The perturbations are followed by the aforementioned local search. Note that the local search that follows the perturbation must only be performed for the machines on which the perturbation has altered the sequence.

#### 4.2.3. Acceptance and termination criteria

Accepting only improved solutions would result in the algorithm converging to a local optimum. Therefore, the acceptance criterion ensures that even an inferior solution may replace the current solution. An improved solution always replaces the current solution. The best solution obtained after the perturbation and local search step is always accepted thus allowing a large diversification until a maximum number of perturbation iterations or a time limit is reached.

## 5. Computational experiments

This section analyses the computational performance of the heuristics and the mathematical formulation. First, the characteristics of the test instances are provided (Section 5.1). Then, the tuning of the parameters is described (Section 5.2). The performance of the solution methods is summarized and discussed in Section 5.3. Experiments were conducted on a 3.00 GHz Intel Core i7–9700 processor with 64 GB of memory running under Windows 10. IBM CPLEX 12.9 on GAMS 29.1.0 was used to implement and solve the mathematical model while allowing CPLEX to use all cores in parallel. The time limit for solving the MIP models was set to 3600 s. The heuristics were implemented and run with R 3.5.1 (using RStudio 1.1.463). The codes of the mathematical model and the heuristics, as well as a brief description of the algorithms are available on https://git.io/JvLrC.

### 5.1. Description of the instances

The solution methods have been tested on two new data sets, SSP-NPM-I and SSP-NPM-II, since the problem has not been addressed in literature so far. SSP-NPM-I has been introduced in order to explore the limits of the mathematical model and provides 'small' instances. The time limit for the mathematical model is 3600 s. SSP-NPM-II is proposed in order to explore the limits of the heuristic solution methods and provides 'large' instances. Table 3 and Table 4 summarize the specifications of the problem sets considered in the experimental evaluation. 20 instances were randomly generated for each problem type. Processing times $p_{jm}$ are random integers generated from a uniform distribution over the intervals [1, 10]. The instances are publicly available on https://git.io/Jvsqp.

Data set SSP-NPM-I provides a total of 160 instances. The number of tools per job is randomly drawn from the interval [3, 5]

**Table 3**
Characteristics of SSP-NPM-I.

| $M$ | $J$ | $T$ | $C_m$ | $sw_m$ |
|---|---|---|---|---|
| 2 | {10, 15} | {10, 15} | (5, 7) | (2, 4) |
| 3 | {15, 20} | {15, 20} | (7, 10, 13) | (2, 3, 4) |

for two-machine instances and from the interval [5, 7] for three-machine instances. The required tools are randomly chosen from the tool set. Note that for both data sets the dominance criterion (Laporte et al., 2004) is met which means that any job's tool set is never a subset of any other job's tool set.

Data set SSP-NPM-II provides 480 instances. Additional instances were defined for several sub-categories in order to evaluate the impact of different switching times and different quantities of required tools per job. The tool switching times $sw_m$ were assumed to be either relatively low or high compared to the processing times. The tool switching time $sw_m$ on machine $m$ is either low, $\lfloor 0.75 \times mean(p_{jm}) \rfloor$, or high, $\lceil 1.15 \times mean(p_{jm}) \rceil$, similar to the instance generation of Beezão et al. (2017). In order to investigate the assumption of Burger, Jacobs, van Vuuren and Visagie (2015) that the difference between the magazine size and the quantity of tools required per job influences the performance of the heuristics, the job-tool combinations, depending on the number of required tools, are either sparse or dense. Sparse matrices were randomly generated based on the number of required tools per job $|\mathcal{T}_j|$ using a uniform distribution over the interval [min $\lceil \frac{C_m}{4} \rceil$, min $\lceil \frac{C_m}{2} \rceil$] while for dense matrices $|\mathcal{T}_j|$ is generated over the interval [max $\lceil \frac{C_m}{2} \rceil$, max $(C_m)$].

### 5.2. Parameter settings

Among the heuristics, only the MSR and the ILS require parameter calibrations. The iteration limit of the MSR was set to 1000. The components of the ILS include (1) the heuristic used to build the initial solution, (2) the type of perturbation move applied, (3) the acceptance threshold, and (4) the stopping criterion. The instances of data set SSP-NPM-I were used to obtain suitable parameter values. The final values of the perturbation parameters are displayed in Table 5. The parameter $\beta$ for the random permutation ($P.2_{rand}$) was analysed for $\beta \in \{0.2, 0.4, 0.6, 0.8\}$. The Wilcoxon Signed-Rank test showed that the computation time and the objective values are not significantly different ($^*p > .05$) for different levels of $\beta$. Therefore, $\beta$ was set to 0.2 for minimizing total flowtime and makespan, and 0.8 for minimizing the number of tool switches since these levels resulted in the best average relative deviations. Since the construction heuristics allocate jobs fairly evenly among machines, only a few jobs need to be shifted to improve the solution for the time-related objectives. However, the optimal solutions showed that when minimizing the number of tool switches, more jobs (i.e. a higher value for $\beta$) are allocated to the machines with a high tool capacity.

For the combined procedures (P.3), the parameter $\gamma$ was analysed for $\gamma \in \{0.05, 0.1, 0.25, 0.5\}$ since the main structure of a good solution should be retained. The parameter $\beta$ was set to $\beta \in \{0.2, 0.4, 0.6, 0.8\}$. All possible combinations of $\beta$ and $\gamma$ were tested. Since no combination was significantly better than all other combinations, the levels with the best average relative deviation values were chosen. For the combined procedure, the parameters were set to $\beta = 0.2$ and $\gamma = 0.1$ for minimizing the number of tool switches and for minimizing the makespan. For minimizing total flowtime the best average relative deviation values were obtained for $\beta = 0.2$ and $\gamma = 0.05$. The results showed that small values of $\gamma$ obtained the best results. The low values of $\gamma$ indicate that a random perturbation is only carried out if the local environment of a solution has been sufficiently investigated.

**Table 5**
Perturbation parameter values.

| Perturbation | Objective | Parameter |
|---|---|---|
| P.2 | *TS* | $\beta = 0.8$ |
| P.2 | *FMAX, TFT* | $\beta = 0.2$ |
| P.3 | *TS, FMAX* | $\beta = 0.2$, $\gamma = 0.1$ |
| P.3 | *TFT* | $\beta = 0.2$, $\gamma = 0.05$ |

Initially, an adequate time-limit for the ILS variations had been evaluated by using 1000 iterations for different problems. It was shown that the combined procedures improve the initial solution very quickly and start to converge after a few iterations. Fig. 4 and 5 confirm the convergence behavior of the individual ILS methods. They display how the average solution values of 10 runs change over 1000 iterations. The maximum computation limit was set to 3600 s for the final evaluation of the different ILS approaches. This allows for a fair comparison between the mathematical model and the heuristic approaches.

### 5.3. Experimental results

For each instance, the relative percentage deviation (RPD(%)) between the solution value found with the considered method and the best known solution value (BKS) were computed as well as the average percentage deviation (AvPD(%)) across all the instances / all the problem sets. To confirm the results, the conclusions of the comparisons were tested applying the Wilcoxon Signed-Rank test, assuming a significance level of $^*p > .05$.

#### 5.3.1. Evaluation of the construction heuristics

Tables 6 and 7 present the AvPD values relative to the best-known solution values obtained by any of the three construction heuristics IEACT, IGI and MSR for a given problem type. The averages of the best objective function values are shown in columns four to six. The average runtimes ($t(s)$) in seconds are provided for each heuristic. The results show that the best starting solutions for minimizing the number of tool switches (*TS*) and minimizing the makespan (*FMAX*) were obtained by the multi-start random heuristic for small problems (Table 6). The IEACT always provides the best results for minimizing the total flowtime (*TFT*). Since the MSR was run for 1000 iterations, its performance decreased for large problems (Table 7). The MSR yielded the worst results for all objectives for large problems, while the best results were obtained by the IGI method for minimizing the total number of tool switches and by the IEACT method for minimizing total flowtime. The computation time of the MSR was significantly higher than the computation time of the IEACT and the IGI for both sets of instances. The computation times of IEACT and IGI are insignificant

**Table 4**
Characteristics of SSP-NPM-II.

| $M$ | $J$ | $T$ | $C_m$ | $sw_m$ | Job-Tool-Matrix Density |
|---|---|---|---|---|---|
| 4 | {40, 60} | 60 | $(25+(m-1) \bullet 5)$ | {low, high} | {sparse, dense} |
| 4 | {40, 60} | 120 | $(25+(m-1) \bullet 5)$ | {low, high} | {sparse, dense} |
| 6 | {80, 120} | 120 | $(45+(m-1) \bullet 5)$ | {low, high} | {sparse, dense} |

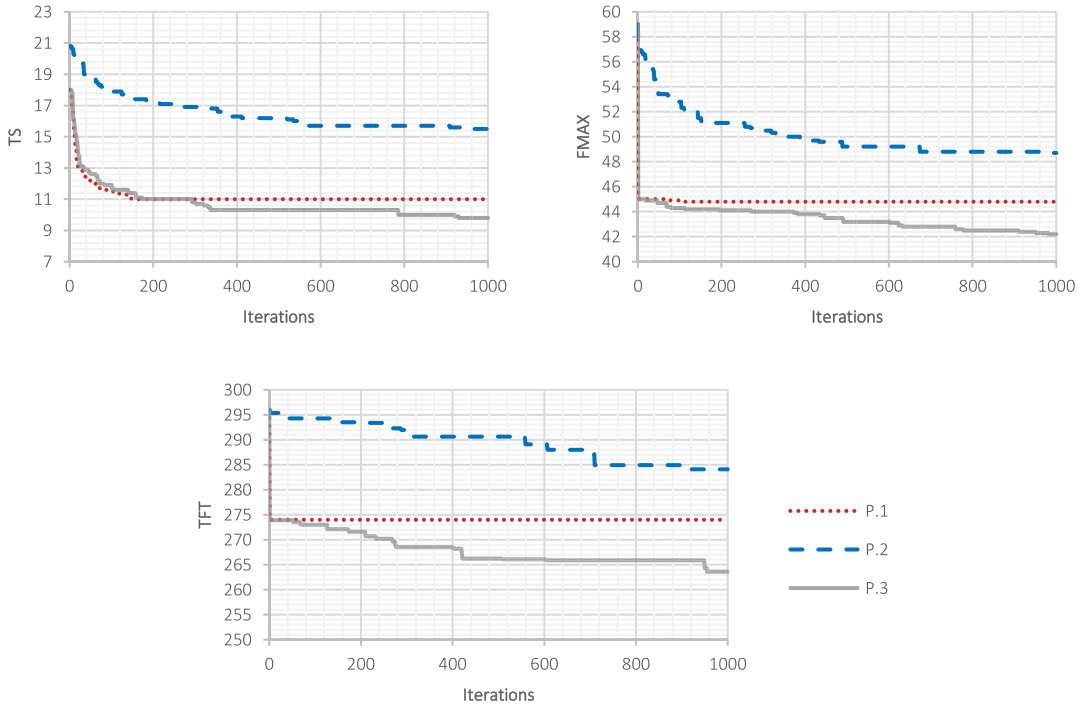


**Fig. 4.** Perturbation behavior for a 3-machines, 15-jobs, and 20-tools problem (instance 121 from SSP-NPM-I).

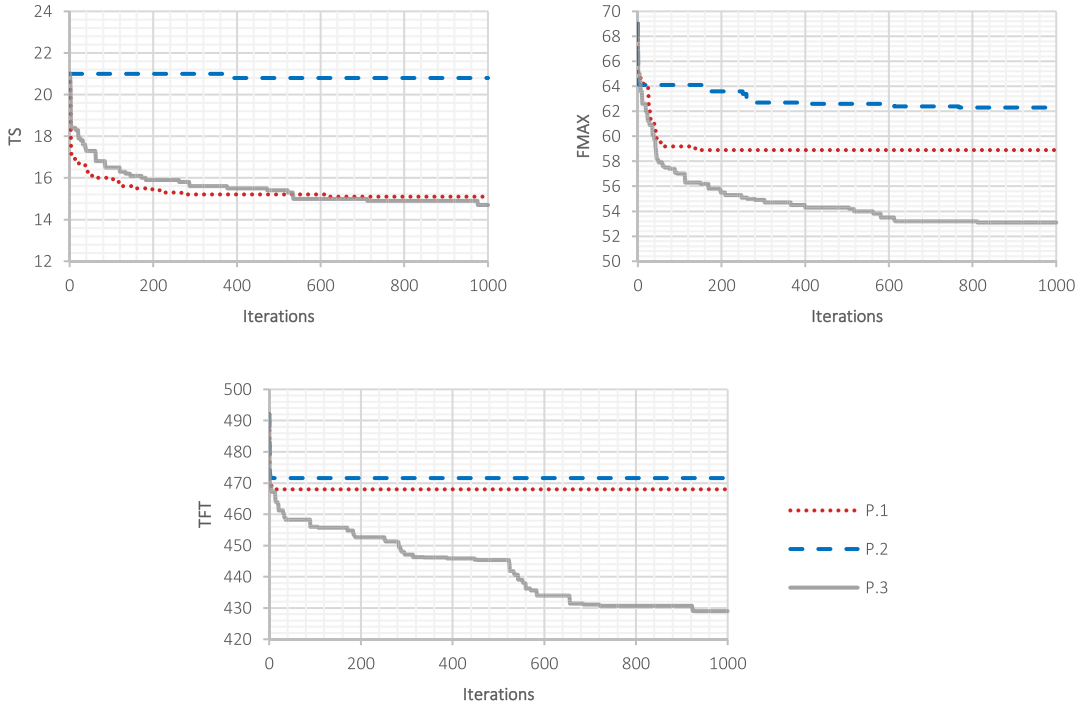


**Fig. 5.** Perturbation behavior for a 3-machines, 20-jobs, and 20-tools problem (instance 141 from SSP-NPM-I).

even for very large problems. Since a good solution should be obtained very fast, MSR cannot compete with the greedy construction methods for large problems. Therefore, IEACT was used for minimizing makespan and total flowtime and IGI was used for minimizing the total number of tool switches for the following local search and iterated local search methods. Overall, the results show that MSR provides good objective values for the makespan and the number of tool switches for small problems. While the runtime is significantly longer than for the other heuristics, a sufficient number of iterations can be performed in just a few seconds, which should be enough to allow job sequences and tool loads to be generated before a shift begins. For problems with more than 3 machines, the greedy heuristics provide an initial job sequence with the corresponding tool loading in extremely short runtime and can therefore be used in companies at any time. An improvement of the initial solutions can subsequently be carried out with the help

**Table 6**
Average percentage deviation relative to the best solution found considering the construction heuristics IEACT, IGI and MSR for each objective (SSP-NPM-I).

| | | | | | | IEACT | | | | | | | IGI | | | | | | | MSR | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Average Best | | | TS | | TFT | | FMAX | | | TS | | TFT | | FMAX | | | TS | | TFT | | FMAX | | |
| M | J | T | TS | TFT | FMAX | Mean | AvPD | Mean | AvPD | Mean | AvPD | t(s) | Mean | AvPD | Mean | AvPD | Mean | AvPD | t(s) | Mean | AvPD | Mean | AvPD | Mean | AvPD | t(s) |
| 2 | 10 | 10 | 4 | 152 | 34 | 9 | 241 | 155 | 2 | 40 | 17 | 0.02 | 8 | 178 | 218 | 45 | 42 | 25 | 0.01 | 4 | 0 | 178 | 17 | 37 | 8 | 4.81 |
| 2 | 10 | 15 | 9 | 179 | 39 | 13 | 51 | 185 | 3 | 44 | 16 | 0.02 | 12 | 37 | 248 | 39 | 52 | 35 | 0.01 | 9 | 0 | 192 | 7 | 40 | 4 | 4.30 |
| 2 | 15 | 10 | 8 | 341 | 57 | 14 | 88 | 342 | 0 | 59 | 4 | 0.04 | 12 | 60 | 462 | 37 | 67 | 18 | 0.02 | 8 | 2 | 468 | 37 | 65 | 14 | 8.36 |
| 2 | 15 | 15 | 16 | 402 | 63 | 21 | 31 | 402 | 0 | 67 | 8 | 0.04 | 18 | 12 | 510 | 27 | 74 | 18 | 0.01 | 17 | 5 | 472 | 17 | 67 | 6 | 6.5 |
| 3 | 15 | 15 | 9 | 214 | 34 | 15 | 57 | 216 | 1 | 43 | 26 | 0.04 | 13 | 39 | 327 | 54 | 52 | 52 | 0.02 | 9 | 0 | 231 | 9 | 34 | 0 | 5.44 |
| 3 | 15 | 20 | 16 | 247 | 40 | 21 | 37 | 250 | 1 | 50 | 24 | 0.04 | 19 | 23 | 353 | 43 | 58 | 44 | 0.02 | 16 | 2 | 267 | 9 | 40 | 0 | 5.76 |
| 3 | 20 | 15 | 16 | 372 | 52 | 21 | 27 | 372 | 0 | 57 | 9 | 0.07 | 18 | 10 | 549 | 49 | 68 | 31 | 0.02 | 17 | 5 | 450 | 21 | 52 | 0 | 7.21 |
| 3 | 20 | 20 | 25 | 451 | 62 | 30 | 20 | 451 | 0 | 66 | 6 | 0.07 | 26 | 3 | 643 | 43 | 80 | 29 | 0.02 | 27 | 7 | 552 | 23 | 63 | 2 | 7.67 |
| **Average** | | | **13** | **295** | **48** | **18** | **69** | **297** | **1** | **53** | **14** | **0.04** | **16** | **45** | **414** | **42** | **62** | **32** | **0.02** | **13** | **3** | **351** | **17** | **50** | **4** | **6.26** |

**Table 7**
Average percentage deviation relative to the best solution found considering the construction heuristics IEACT, IGI and MSR for each objective (SSP-NPM-II).

| | | | | | | IEACT | | | | | | | IGI | | | | | | | MSR | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Average Best | | | TS | | TFT | | FMAX | | | TS | | TFT | | FMAX | | | TS | | TFT | | FMAX | | |
| M | J | T | TS | TFT | FMAX | Mean | AvPD | Mean | AvPD | Mean | AvPD | t(s) | Mean | AvPD | Mean | AvPD | Mean | AvPD | t(s) | Mean | AvPD | Mean | AvPD | Mean | AvPD | t(s) |
| 4 | 40 | 60 | 201 | 6464 | 398 | 221 | 9 | 6466 | 2 | 412 | 4 | 0.26 | 214 | 4 | 6925 | 16 | 417 | 10 | 0.08 | 204 | 4 | 9089 | 34 | 625 | 37 | 36.26 |
| 4 | 40 | 120 | 386 | 11,087 | 750 | 398 | 4 | 11,150 | 34 | 775 | 3 | 0.26 | 390 | 1 | 11,499 | 9 | 780 | 6 | 0.08 | 393 | 4 | 15,888 | 32 | 1159 | 35 | 37.44 |
| 4 | 60 | 60 | 326 | 15,071 | 595 | 342 | 5 | 15,073 | 1 | 605 | 2 | 0.58 | 332 | 2 | 16,198 | 15 | 625 | 8 | 0.16 | 335 | 9 | 21,280 | 36 | 959 | 43 | 55.31 |
| 4 | 60 | 120 | 599 | 25,611 | 1095 | 609 | 3 | 25,723 | 22 | 1124 | 2 | 0.57 | 601 | 0 | 26,638 | 8 | 1150 | 6 | 0.16 | 638 | 9 | 37,171 | 35 | 1749 | 40 | 57.41 |
| 6 | 80 | 120 | 943 | 35,705 | 1059 | 971 | 4 | 35,705 | 0 | 1079 | 3 | 1.05 | 945 | 0 | 37,560 | 8 | 1100 | 5 | 0.31 | 979 | 9 | 50,698 | 33 | 1969 | 57 | 110.48 |
| 6 | 120 | 120 | 1461 | 85,749 | 1635 | 1500 | 3 | 85,786 | 8 | 1648 | 1 | 2.38 | 1463 | 0 | 89,415 | 6 | 1724 | 6 | 0.66 | 1568 | 16 | 118,401 | 34 | 2973 | 58 | 177.75 |
| **Average** | | | **653** | **29,948** | **922** | **674** | **5** | **29,984** | **11** | **941** | **3** | **0.85** | **658** | **1** | **31,372** | **10** | **966** | **7** | **0.24** | **686** | **8** | **42,088** | **34** | **1572** | **45** | **79.11** |

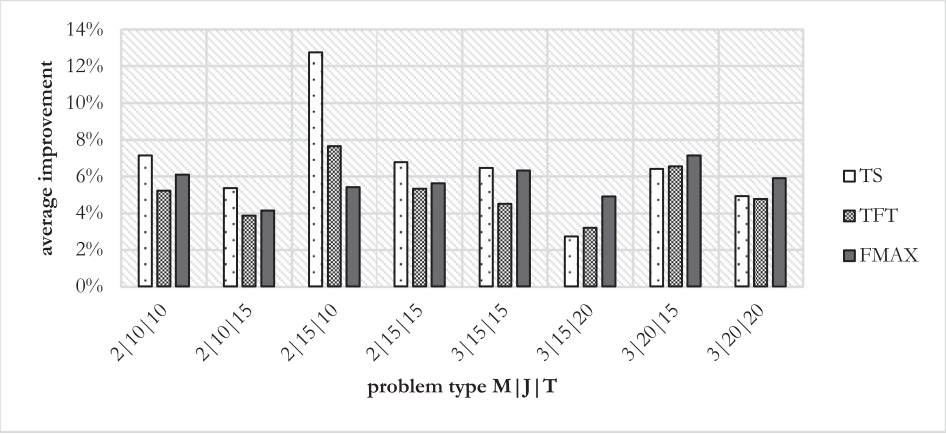


**Fig. 6.** Average percentage improvement of the initial solution per problem type using local search for SSP-NPM-I.

of local search methods. The results of the local search algorithm used in this paper are presented in the following section.

*5.3.2. Evaluation of the local search*

The local search (Algorithm 5) has the advantage that it is rather simple and easy to implement, and it requires no parameter tuning. The results show that applying an initial LS to the solution of the construction heuristics significantly improves the solution quality. The average percentage improvement for different problem types in SSP-NPM-I lies between 3% and 13% improvement of the construction heuristics for minimizing TS, between 3% and 8% for minimizing TFT and between 4% and 7% for minimizing FMAX (Fig. 6). For SSP-NPM-II, the average percentage improvement lies between 1% and 6% improvement of the construction heuristics for minimizing TS, between 6% and 13% for minimizing TFT and between 1% and 6% for minimizing FMAX (Fig. 7). The results show no significant difference in the improvement for instances with different tool switching times but a significant difference for sparse and dense matrices (Fig. 8). The improvement for sparse matrices is significantly higher than for dense matrices but requires higher computation times (Fig. 9). Dense problems require more tool switches and have thus higher objective values. Overall, a local search should always be performed, since it leads to a significant improvement of the objective function values with only little computational effort.

*5.3.3. Evaluation of the ILS methods*

The results of the ILS methods had been compared to the mathematical formulation using the data-set SSP-NPM-I. Table 8 summarizes the performance of the mathematical formulation. The fourth, seventh and tenth column provide the average total tool switches (avTS), the average total flowtime (avTFT) and the average makespan (avFMAX) per problem type as additional information. Moreover, the average computation time in seconds (time(s)) and then number of instances solved to optimality (#Opt) per 20 problem type instances are provided. It is shown that only few instances can be solved to optimality given the time-limit of 3600 s and heuristics have to be used for larger-sized problems.

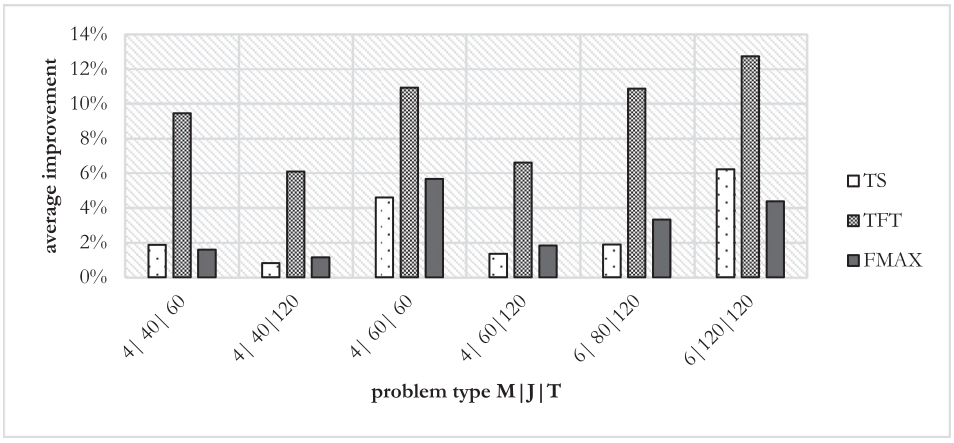


**Fig. 7.** Average percentage improvement of the initial solution per problem type using local search for SSP-NPM-II.

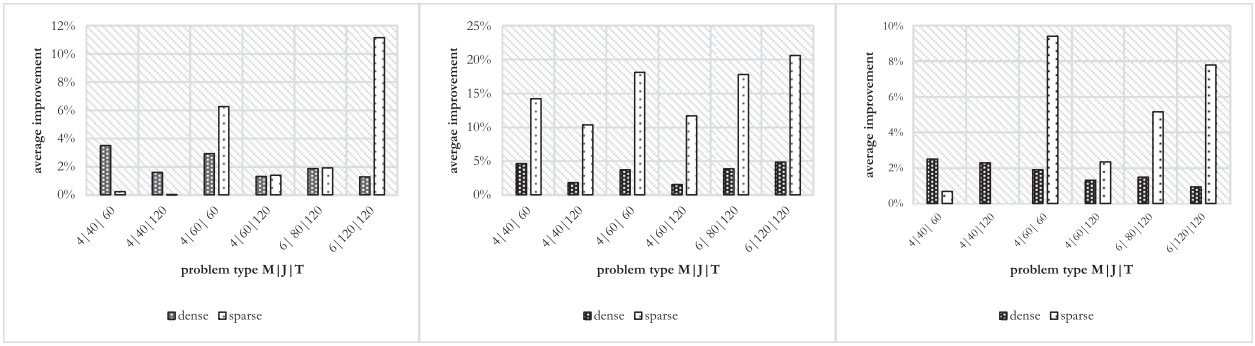


**Fig. 8.** Average percentage improvement of the initial solution per problem type using local search for sparse and dense matrices of SSP-NPM-II.

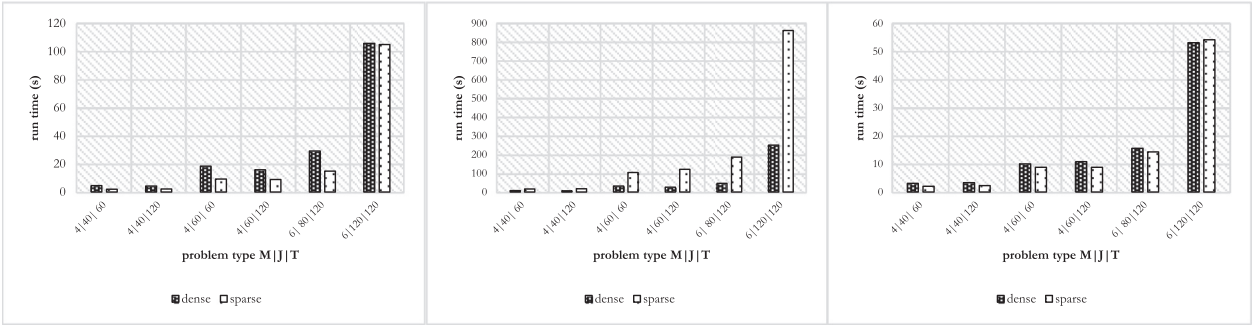


**Fig. 9.** Comparison of the average computation times per problem type of the local search for sparse and dense matrices of SSP-NPM-II.

**Table 8**
Performance of the mathematical formulation for SSP-NPM-I.

| *M* | *J* | *T* | TS | | | TFT | | | FMAX | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | avTS | time(s) | #Opt | avTFT | time(s) | #Opt | avFMAX | time(s) | #Opt |
| 2 | 10 | 10 | 3 | 12 | 20 | 129 | 535 | 20 | 29 | 306 | 20 |
| 2 | 10 | 15 | 6 | 64 | 20 | 163 | 1308 | 20 | 37 | 1317 | 18 |
| 2 | 15 | 10 | 4 | 621 | 19 | 276 | limit | 0 | 44 | limit | 0 |
| 2 | 15 | 15 | 9 | limit | 0 | 344 | limit | 0 | 54 | limit | 0 |
| 3 | 15 | 15 | 1 | 387 | 20 | 164 | limit | 0 | 26 | limit | 0 |
| 3 | 15 | 20 | 5 | limit | 0 | 196 | limit | 0 | 33 | limit | 0 |
| 3 | 20 | 15 | 2 | 2411 | 8 | 290 | limit | 0 | 37 | limit | 0 |
| 3 | 20 | 20 | 7 | limit | 0 | 396 | limit | 0 | 49 | limit | 0 |

**Table 9**
Average Percentage Deviation relative to the best solution found considering the MIP and the different ILS methods for SSP-NPM-I.

| *M* | *J* | *T* | AvPD TS | | | | AvPD TFT | | | | AvPD FMAX | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MIP | P.1$_{switch}$ | P.2 | P.3 | MIP | P.1$_{time}$ | P.2 | P.3 | MIP | P.1$_{time}$ | P.2 | P.3 |
| 2 | 10 | 10 | 0.00 | 42.08 | 6.67 | 3.33 | 0.00 | 4.01 | 0.84 | 0.30 | 0.00 | 6.03 | 0.48 | 0.00 |
| 2 | 10 | 15 | 0.00 | 33.86 | 6.56 | 6.23 | 0.20 | 4.34 | 0.71 | 0.31 | 0.00 | 5.85 | 0.85 | 0.00 |
| 2 | 15 | 10 | 0.00 | 57.08 | 41.67 | 33.08 | 1.80 | 5.46 | 5.24 | 0.95 | 1.86 | 6.42 | 10.53 | 0.70 |
| 2 | 15 | 15 | 0.00 | 25.33 | 24.85 | 18.82 | 1.35 | 6.47 | 5.30 | 1.24 | 3.72 | 6.54 | 10.21 | 0.18 |
| 3 | 15 | 15 | 0.00 | 168.42 | 236.84 | 5.26 | 1.04 | 11.04 | 18.47 | 1.72 | 2.23 | 17.54 | 22.05 | 1.03 |
| 3 | 15 | 20 | 0.00 | 92.81 | 74.37 | 30.27 | 1.20 | 12.50 | 16.48 | 1.88 | 4.66 | 18.38 | 22.19 | 0.82 |
| 3 | 20 | 15 | 0.00 | 185.00 | 382.50 | 57.50 | 3.79 | 10.18 | 22.41 | 1.73 | 8.20 | 18.77 | 29.68 | 0.37 |
| 3 | 20 | 20 | 0.00 | 66.26 | 103.89 | 47.97 | 6.77 | 5.20 | 13.18 | 0.78 | 13.98 | 9.91 | 24.10 | 0.12 |
| **Average** | | | **0.00** | **83.86** | **109.67** | **25.31** | **2.02** | **7.40** | **10.33** | **1.11** | **4.33** | **11.18** | **15.01** | **0.40** |

**Table 10**
Average percentage deviation relative to the best solution found considering the ILS schemes (P.1, P.2, P.3) for each objective (SSP-NPM-I).

| | | | | | | P.1 | | | | | | P.2 | | | | | | P.3 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *Average Best* | | | *TS* | | *TFT* | | *FMAX* | | *TS* | | *TFT* | | *FMAX* | | *TS* | | *TFT* | | *FMAX* | |
| *M* | *J* | *T* | *TS* | *TFT* | *FMAX* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* |
| 2 | 10 | 10 | 3 | 130 | 29 | 4 | 38 | 135 | 4 | 31 | 6 | 3 | 3 | 130 | 1 | 29 | 0 | 3 | 0 | 130 | 0 | 29 | 0 |
| 2 | 10 | 15 | 7 | 164 | 37 | 8 | 31 | 170 | 4 | 39 | 6 | 7 | 4 | 165 | 1 | 37 | 1 | 7 | 4 | 164 | 0 | 37 | 0 |
| 2 | 15 | 10 | 5 | 278 | 44 | 6 | 25 | 290 | 5 | 47 | 6 | 6 | 12 | 289 | 4 | 48 | 10 | 5 | 5 | 278 | 0 | 44 | 0 |
| 2 | 15 | 15 | 11 | 345 | 53 | 12 | 8 | 363 | 5 | 57 | 7 | 12 | 8 | 359 | 4 | 59 | 10 | 11 | 3 | 345 | 0 | 53 | 0 |
| 3 | 15 | 15 | 1 | 163 | 26 | 3 | 163 | 178 | 9 | 30 | 16 | 3 | 229 | 190 | 17 | 31 | 21 | 1 | 0 | 163 | 0 | 26 | 0 |
| 3 | 15 | 20 | 7 | 198 | 32 | 10 | 47 | 217 | 10 | 38 | 17 | 9 | 37 | 226 | 14 | 39 | 21 | 7 | 0 | 198 | 0 | 32 | 0 |
| 3 | 20 | 15 | 2 | 289 | 35 | 4 | 78 | 312 | 8 | 41 | 18 | 7 | 213 | 346 | 21 | 45 | 29 | 2 | 0 | 289 | 0 | 35 | 0 |
| 3 | 20 | 20 | 12 | 377 | 45 | 13 | 13 | 393 | 5 | 49 | 10 | 16 | 40 | 423 | 13 | 55 | 24 | 12 | 1 | 377 | 0 | 45 | 0 |
| **Average** | | | **6** | **243** | **38** | **7** | **50** | **257** | **6** | **41** | **11** | **8** | **68** | **266** | **9** | **43** | **15** | **6** | **2** | **243** | **0** | **38** | **0** |

**Table 11**
Average percentage deviation relative to the best solution found considering the ILS schemes (P.1, P.2, P.3) for each objective (SSP-NPM-II).

| | | | | | | P.1 | | | | | | P.2 | | | | | | P.3 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | *Average Best* | | | *TS* | | *TFT* | | *FMAX* | | *TS* | | *TFT* | | *FMAX* | | *TS* | | *TFT* | | *FMAX* | |
| *M* | *J* | *T* | *TS* | *TFT* | *FMAX* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* | *Mean* | *AvPD* |
| 4 | 40 | 60 | 171 | 5845 | 371 | 196 | 13 | 5845 | 1 | 371 | 1 | 171 | 14 | 6151 | 13 | 401 | 16 | 180 | 8 | 5855 | 1 | 374 | 2 |
| 4 | 40 | 120 | 362 | 10,672 | 717 | 380 | 5 | 10,672 | 0 | 718 | 1 | 362 | 4 | 11,022 | 8 | 766 | 11 | 369 | 3 | 10,691 | 1 | 717 | 1 |
| 4 | 60 | 60 | 275 | 13,718 | 547 | 310 | 9 | 13,718 | 0 | 547 | 0 | 275 | 12 | 14,397 | 11 | 583 | 14 | 300 | 12 | 13,760 | 1 | 556 | 3 |
| 4 | 60 | 120 | 568 | 24,687 | 1054 | 590 | 3 | 24,687 | 0 | 1054 | 0 | 568 | 3 | 25,401 | 6 | 1115 | 8 | 584 | 4 | 24,738 | 0 | 1061 | 1 |
| 6 | 80 | 120 | 857 | 33,163 | 1016 | 918 | 5 | 33,163 | 0 | 1016 | 0 | 857 | 4 | 34,421 | 9 | 1063 | 10 | 912 | 6 | 33,177 | 0 | 1021 | 2 |
| 6 | 120 | 120 | 1358 | 78,492 | 1543 | 1416 | 4 | 78,492 | 0 | 1543 | 0 | 1358 | 6 | 81,060 | 7 | 1637 | 13 | 1414 | 4 | 78,515 | 0 | 1555 | 2 |
| **Average** | | | **599** | **27,763** | **875** | **635** | **6** | **27,763** | **0** | **875** | **1** | **598** | **7** | **28,742** | **9** | **928** | **12** | **627** | **6** | **27,789** | **1** | **881** | **2** |

The mathematical model is obviously not suitable for large problems. The model especially struggles with the time-related objectives where only instances with 10 jobs can be solved to optimality. Therefore, from a managerial perspective, heuristic solution approaches provide better results in less computation time and should be favoured for most realistic problems with more than 10 jobs.

Table 9 shows the performance comparison of the mathematical formulation and the ILS methods. Underlined values indicate the best results. The AvPD is calculated based on the best solution found considering the different ILS methods and the mathematical formulation. For the objective of minimizing the number of tool switches (TS) the mathematical formulation significantly outperforms the ILS heuristics. It has to be noted that the high deviation derives from the low tool switching values, so that even a solution which is very close to the solution of the solver yields a high relative deviation. However, as mentioned in the introduction, in many real-world manufacturing scenarios, minimizing the makespan or total flowtime may be more appropriate than minimizing the number of tool switches, for which the ILS heuristics provide good and robust results. Table 10 provides a direct comparison of the ILS versions for small instances and shows that the P.3 outperforms the other ILS methods for small instances. For small problems (SSP-NPM-I), the combination of objective-specific and randomized perturbation steps (P.3) yields the best results out of the ILS methods for each objective and can compete even with the performance of the mathematical model. The ILS heuristic even outperforms the mathematical formulation for time-related objectives.

For larger problems (SSP-NPM-II) the random component is responsible for a deterioration of the performance of the random perturbation (P.2) and the combined perturbation (P.3). The results are summarized in Table 11. Although P.1 yields the best results for SSP-NPM-II in many of the problem instances it has to be mentioned that the combined perturbation (P.3) overall provides good and robust results for any problem type, even for sparse and dense matrices (see Table 12). Different switching times did not provide a significant difference in the performance of the heuristics. For sparse and dense problems, however, it could be observed that the different methods yield different results for each objective. This means different strategies have to be applied in real-world scenarios when the number of required tools is either close to the tool magazine capacities or when the number of required tools is low compared to the tool magazine capacities. The objective-specific perturbation (P.1) outperforms the other perturbation schemes for sparse problems when the number of required tools is low compared to the tool capacity of the machines. The scheme P.2 only

**Table 12**
Average percentage deviation relative to the best solution found considering the different ILS methods for SSP-NPM-II.

| Density | *M* | *J* | *T* | AvPD TS | | | AvPD TFT | | | AvPD FMAX | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | P.1$_{switch}$ | P.2 | P.3 | P.1$_{time}$ | P.2 | P.3 | P.1$_{time}$ | P.2 | P.3 |
| sparse | 4 | 40 | 60 | 5.02 | 28.60 | 7.56 | 1.19 | 21.44 | 1.91 | 2.01 | 26.05 | 2.80 |
| sparse | 4 | 40 | 120 | 1.97 | 7.01 | 3.17 | 0.31 | 14.26 | 1.11 | 2.07 | 15.06 | 1.25 |
| sparse | 4 | 60 | 60 | 0.22 | 23.61 | 10.96 | 0.16 | 18.34 | 1.35 | 0.63 | 23.49 | 5.32 |
| sparse | 4 | 60 | 120 | 0.33 | 6.01 | 3.60 | 0.12 | 10.79 | 0.28 | 0.32 | 11.90 | 2.54 |
| sparse | 6 | 80 | 120 | 0.24 | 7.80 | 3.83 | 0.03 | 16.29 | 0.24 | 0.67 | 17.47 | 2.92 |
| sparse | 6 | 120 | 120 | 0.18 | 12.60 | 1.01 | 0.06 | 11.08 | 0.22 | 0.25 | 24.57 | 3.61 |
| **Average** | | | | **1.33** | **14.27** | **5.02** | **0.31** | **15.37** | **0.85** | **0.99** | **19.76** | **3.07** |
| dense | 4 | 40 | 60 | 20.51 | 0.00 | 9.36 | 0.39 | 3.75 | 0.52 | 0.44 | 6.16 | 1.32 |
| dense | 4 | 40 | 120 | 7.05 | 0.00 | 3.19 | 0.09 | 1.79 | 0.20 | 0.51 | 6.15 | 0.53 |
| dense | 4 | 60 | 60 | 18.56 | 0.00 | 12.53 | 0.27 | 3.21 | 0.44 | 0.00 | 3.53 | 0.94 |
| dense | 4 | 60 | 120 | 5.75 | 0.01 | 3.99 | 0.01 | 1.54 | 0.22 | 0.16 | 4.45 | 0.46 |
| dense | 6 | 80 | 120 | 9.61 | 0.00 | 8.19 | 0.15 | 2.21 | 0.18 | 0.04 | 2.54 | 0.18 |
| dense | 6 | 120 | 120 | 7.14 | 0.00 | 6.88 | 0.18 | 2.16 | 0.17 | 0.00 | 2.09 | 0.23 |
| **Average** | | | | **11.43** | **0.00** | **7.36** | **0.18** | **2.44** | **0.29** | **0.19** | **4.15** | **0.61** |
| **Total Average** | | | | **6.38** | **7.14** | **6.19** | **0.25** | **8.90** | **0.57** | **0.59** | **11.96** | **1.84** |

outperforms the other perturbation schemes for the minimisation of the number of tool switches for dense problems, where a random component is required in order to overcome local optima. For sparse problems and the minimisation of the number of tool switches the objective-specific perturbation method (P.1$_{switch}$) even outperforms the other since for sparse problems there exist many more possibilities to fill empty tool slots. For dense matrices, however, with less empty slots, the objective-specific method seems to get stuck in local optima which leads to worse results. Here, the random perturbation (P.2) yields extremely good results.

The objective-specific perturbation (P.1$_{time}$) and the combined perturbation (P.3) outperform the random perturbation (P.2) for sparse and dense problems for minimizing the total flowtime. On a closer look, it can be observed that P.1$_{time}$ always yields a lower AvPD than P.3 for sparse problems. The results of minimizing the makespan are similar to the results of minimizing the total flowtime. The random perturbation (P.2) always yields the worst results. For each method, the AvPD-values for dense problems are lower than the values for sparse problems showing that sparse problems are more difficult to solve because of the many tool optimization possibilities.

## 6. Conclusion

In this paper, the challenging job sequencing and tool switching problem in the context of non-identical parallel machines is investigated. A MIP formulation is presented that can be applied to different objective functions. The computational experiments show that only small problems can be solved to optimality. New ILS methods are proposed since large problems cannot be solved to optimality within reasonable run time. Several construction heuristics and perturbation schemes are evaluated. It is shown that the local search is indispensable and that the proposed ILS methods are able to provide good solutions even for large problem instances. In general, the ILS method with objective-specific and randomized perturbation moves outperforms the objective-specific ILS method as well as the randomized ILS method for small problems. Although the objective-specific heuristics tend to converge to local optima, they are able to find good solutions faster than the ILS heuristics with a random component. The randomized heuristic requires more time to find promising areas of the solution space but may overcome local optima which proves useful for minimizing the number of tool switches of dense problems.

The focus of this work are different perturbation strategies for three different objectives, and thus only one local search method had been included in the experimental evaluation. Further research may investigate the impact of different local search techniques on the performance of the ILS or meta-heuristics. Moreover, examining different acceptance and stopping criteria may prove useful. In the presented objective-specific ILS, any iteration could return to a previously examined solution. Implementing a tabu-list may improve the performance of the ILS by prohibiting the objective-specific perturbation to get stuck in a local optimum.

Extensions of the SSP-NPM may consider tool wear or different tool sizes and tool-dependent setup times. The SSP with different tool sizes has already been studied for a single-machine (see, for example, Hirvikorpi et al., 2006; Raduly-Baka, Knuutila & Nevalainen, 2005; Van Hop, 2005). Different tool sizes appear for example in the metal-working industry where different metal-working operations require tools of different sizes. Some tools may also require special care or handling so that the tool switching time may not be equal for each tool. Another interesting extension of the SSP-NPM may consider that the number of available tools may be limited and that some tools could be handled by only specific machines.

In this paper, the three conflicting objectives are considered individually. By using multi-criteria approaches, further insights could be derived about either the structure of the problem as well as the interrelation of the objectives. An attempt to obtain non-dominated solutions can easily be made by adapting the provided MIP. Overall, the job-sequencing and tool switching problem with non-identical parallel machines demonstrated to be a challenging problem which offers many possibilities for future research.

## Appendix

**Algorithm 1**
Iterated Earlieast Artifical Completion Time Heuristic (IEACT).

Let $[\,r\,]_m$ denote the job processed in position $r$ on machine $m$ and let $\boldsymbol{\mathcal{S}}_m = (\,[\,1\,]_m, [\,2\,]_m, \ldots, [\,l\,]_m)$ be the job sequence on machine $m$, and $\boldsymbol{\mathcal{T}}_m$ the corresponding optimal tool loading on machine $m$, $m = 1, 2, \ldots, M$. Let $\boldsymbol{\mathcal{AC}}_j$ be the Artificial Completion time of job $j$, $j = 1, 2, \ldots, J$.

**Initialize**

$\boldsymbol{\mathcal{L}} \leftarrow \{1, \ldots, J\}$ ; // *set of jobs to be scheduled* //

$\boldsymbol{\mathcal{S}}_m \leftarrow \varnothing$ , $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *job sequence on machine m* //

$\boldsymbol{\mathcal{T}}_m \leftarrow \varnothing$ , $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *optimal tool loading on machine m* //

$f_m \leftarrow 0$, $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *artificial completion time of the last job on machine m* //

**while** $\boldsymbol{\mathcal{L}} \neq \varnothing$ **do**

  $m' \leftarrow argmin\ \{ f_m \mid \exists j \in \boldsymbol{\mathcal{L}} : (|\boldsymbol{\mathcal{T}}_j| \leq C_m), m \in \boldsymbol{\mathcal{M}} \}$ ;
  // *select the machine m' with the minimum completion time of the last job ($f_{m'}$) and for which there exists at least one remaining job that can be processed on that machine without violating capacity restrictions* //

  **if** $f_{m'} = 0$

    $j' \leftarrow argmin\ \{ p_{jm'} \mid j \in \boldsymbol{\mathcal{L}} : (|\boldsymbol{\mathcal{T}}_j| \leq C_{m'}) \}$ ;
    // *select the job (j') with the minimum processing time ($p_{j'm'}$) on the selected machine which requires less tools than the capacity of that machine ($C_{m'}$)* //

    $f_{m'} \leftarrow p_{j'm'}$ ;
    // *update the artifical completion time of the last job on machine m'* //

  **else**

    **for each** $j \in \boldsymbol{\mathcal{L}} : (|\boldsymbol{\mathcal{T}}_j| \leq C_{m'})$

      $\boldsymbol{\mathcal{AC}}_j = p_{jm'} + \mathrm{sw}_{m'} \cdot |\{t \in \boldsymbol{\mathcal{T}}_j \mid t \notin \boldsymbol{\mathcal{T}}_{[\,l\,]_{m'}}\}|$ ;
      // *calculate the artificial completion time ($\boldsymbol{\mathcal{AC}}_j$) for each remaining job* //

    **end for each**

    $j' \leftarrow argmin\ \{\boldsymbol{\mathcal{AC}}_j \mid j \in \boldsymbol{\mathcal{L}} : (|\boldsymbol{\mathcal{T}}_j| \leq C_{m'})\}$ ;
    // *select the job j' with the minimum earliest artificial completion time ($\boldsymbol{\mathcal{AC}}_j$) which requires less tools than the tool magazine capacity of the selected machine ($C_{m'}$)* //

    $f_{m'} \leftarrow f_{m'} + \boldsymbol{\mathcal{AC}}_{j'}$ ;
    // *update the artifical completion time of the last job on machine m'* //

  **end if**

  $\boldsymbol{\mathcal{S}}_{m'} \leftarrow \boldsymbol{\mathcal{S}}_{m'}$ , $\{j'\}$ ;
  // *append the job j' to the job sequence $\boldsymbol{\mathcal{S}}_{m'}$ of machine m'* //

  $\boldsymbol{\mathcal{L}} \leftarrow \boldsymbol{\mathcal{L}} \backslash \{j'\}$ ; // *remove j' from $\boldsymbol{\mathcal{L}}$* //

**end while**

**for each** $m \in \boldsymbol{\mathcal{M}}$

  $\boldsymbol{\mathcal{T}}_m \leftarrow \mathrm{KTNS}\ (\boldsymbol{\mathcal{S}}_m)$ ;
  // *apply the "Keep Tool Needed Soonest"-policy using the obtained sequence $\boldsymbol{\mathcal{S}}_m$ in order to optimize the tool loading on machine m* //

**end for each**

**return** $\boldsymbol{\mathcal{S}}^0 = (\boldsymbol{\mathcal{S}}_1, \boldsymbol{\mathcal{S}}_2, \ldots, \boldsymbol{\mathcal{S}}_M)$ ; $\boldsymbol{\mathcal{T}}^0 = (\boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2, \ldots, \boldsymbol{\mathcal{T}}_M)$ ;
// *return the job sequences $\boldsymbol{\mathcal{S}}^0$ and the corresponding optimal tool loadings $\boldsymbol{\mathcal{T}}^0$* //

**Algorithm 2**
Iterated Greatest Intersection Heuristic (IGI).

Let $[r]_m$ denote the job processed in position $r$ on machine $m$ and let $\boldsymbol{\mathcal{S}}_m = ([1]_m, [2]_m, \ldots, [l]_m)$ be the job sequence on machine $m$, and $\boldsymbol{\mathcal{T}}_m$ the corresponding optimal tool loading on machine $m$, $m = 1, 2, \ldots, M$.

**Initialize**

$\boldsymbol{\mathcal{L}} \leftarrow \{1, \ldots, J\}$ ; // *set of jobs to be scheduled* //

$\boldsymbol{\mathcal{S}}_m \leftarrow \emptyset$ , $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *job sequence on machine m* //

$\boldsymbol{\mathcal{T}}_m \leftarrow \emptyset$ , $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *optimal tool loading on machine m* //

$f_m \leftarrow 0$, $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *artificial completion time of the last job on machine m* //

**while** $\boldsymbol{\mathcal{L}} \neq \emptyset$ **do**

  $m' \leftarrow argmin \{ f_m \mid \exists j \in \boldsymbol{\mathcal{L}} : (|\boldsymbol{\mathcal{T}}_j| \leq C_m), m \in \boldsymbol{\mathcal{M}} \}$ ;

  // *select the machine m' with the minimum completion time of the last job ($f_{m'}$) and for which there exists at least one remaining job that can be processed on that machine without violating capacity restrictions* //

  **if** $f_{m'} = 0$

    $j' \leftarrow argmax \{ |\boldsymbol{\mathcal{T}}_j| \mid j \in \boldsymbol{\mathcal{L}} : (|\boldsymbol{\mathcal{T}}_j| \leq C_{m'}) \}$ ;

    // *select the job (j') with the maximum number of required tools ($|\boldsymbol{\mathcal{T}}_{j'}|$) on the selected machine, and which requires less tools than the capacity of that machine ($C_{m'}$)* //

    $f_{m'} \leftarrow p_{j'm'}$ ;

    // *update the artifical completion time of the last job on machine m'* //

  **else**

    $j' \leftarrow argmax \{ |\boldsymbol{\mathcal{T}}_j \cap \boldsymbol{\mathcal{T}}_{[l]_{m'}}| \mid j \in \boldsymbol{\mathcal{L}} : (|\boldsymbol{\mathcal{T}}_j| \leq C_{m'}) \}$ ;

    // *select the job j' with the maximum number of tool intersections with the currently last job in the sequence on machine m', and which requires less tools than the tool magazine capacity of the selected machine ($C_{m'}$)* //

    $f_{m'} \leftarrow f_{m'} + p_{j'm'} + \mathrm{sw}_{m'} \cdot |\{ t \in \boldsymbol{\mathcal{T}}_{j'} \mid t \notin \boldsymbol{\mathcal{T}}_{[l]_{m'}} \}|$ ;

    // *update the artifical completion time of the last job on machine m'* //

  **end if**

  $\boldsymbol{\mathcal{S}}_{m'} \leftarrow \boldsymbol{\mathcal{S}}_{m'}$ , $\{ j' \}$ ;

  // *append the job j' to the job sequence $\boldsymbol{\mathcal{S}}_{m'}$ of machine m'* //

  $\boldsymbol{\mathcal{L}} \leftarrow \boldsymbol{\mathcal{L}} \backslash \{ j' \}$ ; // *remove j' from* $\boldsymbol{\mathcal{L}}$ //

**end while**

**for each** $m \in \boldsymbol{\mathcal{M}}$

  $\boldsymbol{\mathcal{T}}_m \leftarrow \mathrm{KTNS}\ (\boldsymbol{\mathcal{S}}_m)$ ;

  // *apply the "Keep Tool Needed Soonest"-policy using the obtained sequence $\boldsymbol{\mathcal{S}}_m$ in order to optimize the tool loading on machine m* //

**end for each**

**return** $\boldsymbol{\mathcal{S}}^0 = (\boldsymbol{\mathcal{S}}_1, \boldsymbol{\mathcal{S}}_2, \ldots, \boldsymbol{\mathcal{S}}_M)$ ; $\boldsymbol{\mathcal{T}}^0 = (\boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2, \ldots, \boldsymbol{\mathcal{T}}_M)$ ;
// *return the job sequences $\boldsymbol{\mathcal{S}}^0$ and the corresponding optimal tool loadings $\boldsymbol{\mathcal{T}}^0$* //

**Algorithm 3**
Multi-Star-Random Heuristic (MSR).

Let $[\,r\,]_m$ denote the job processed in position $r$ on machine $m$ and let $\boldsymbol{\mathcal{S}}_m = (\,[\,1\,]_m, [\,2\,]_m, \ldots, [\,l\,]_m)$ be the job sequence on machine $m$, and $\boldsymbol{\mathcal{T}}_m$ the corresponding optimal tool loading on machine $m$, $m = 1, 2, \ldots, M$.
**Initialize**
$\boldsymbol{\mathcal{S}}_m \leftarrow \emptyset$ , $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *job sequence on machine m* //
$\boldsymbol{\mathcal{T}}_m \leftarrow \emptyset$ , $\forall m \in \boldsymbol{\mathcal{M}}$ ; // *optimal tool loading on machine m* //
I ; // *iteration limit* //
**for** $i = 1, 2, \ldots, I$ **do**
  $\boldsymbol{\mathcal{S}}_{rand} \leftarrow$ *random.permutation*$\{1,\ldots, J\}$ ;
  // *generate a random job permutation, i.e. a "giant-sequence"* //
  $m' \leftarrow 1$ ; // *selected machine* //
  $r' \leftarrow 1$ ; // *job* $[\,r'\,]$ *in position* $r'$ *in* $\boldsymbol{\mathcal{S}}_{rand}$ //
  **while** $r' \leq J$ **do**
    **if** $|\boldsymbol{\mathcal{T}}_{[\,r'\,]}| \leq C_{m'}$
      $\boldsymbol{\mathcal{S}}_{m'} \leftarrow \boldsymbol{\mathcal{S}}_{m'}$ , $\boldsymbol{\mathcal{S}}_{rand}\,[\,r'\,]$ ;
      // *append the job in position* $r'$ *in* $\boldsymbol{\mathcal{S}}_{rand}$ *to the job sequence* $\boldsymbol{\mathcal{S}}_{m'}$ *of machine* $m'$ //
      $m' \leftarrow m' + 1$ ; // *select next machine* //
      **if** $m' > M$
        $m' \leftarrow 1$ ; // *if there is no next machine, restart with machine 1* //
      **end if**
    **else**
      $m' \leftarrow$ *random.machine*$\{m \in \boldsymbol{\mathcal{M}} \mid (|\boldsymbol{\mathcal{T}}_{[\,r'\,]}| \leq C_m)\}$ ;
      // *randomly chose a machine with sufficient tool magazine capacity* //
      $\boldsymbol{\mathcal{S}}_{m'} \leftarrow \{\,\boldsymbol{\mathcal{S}}_{m'}$ , $\boldsymbol{\mathcal{S}}_{rand}\,[\,r'\,]\}$ ;
      // *append the job in position* $r'$ *in* $\boldsymbol{\mathcal{S}}_{rand}$ *to the job sequence* $\boldsymbol{\mathcal{S}}_{m'}$ *of machine* $m'$ //
    **end if**
    $r' \leftarrow r' + 1$ ; // *select the next job* //
  **end while**
**end for**
**for each** $m \in \boldsymbol{\mathcal{M}}$
  $\boldsymbol{\mathcal{T}}_m \leftarrow$ KTNS $(\boldsymbol{\mathcal{S}}_m)$ ;
  // *apply the "Keep Tool Needed Soonest"-policy using the obtained sequence* $\boldsymbol{\mathcal{S}}_m$ *in order to optimize the tool loading on machine m* //
**end for each**
**return** $\boldsymbol{\mathcal{S}}^0 = (\boldsymbol{\mathcal{S}}_1, \boldsymbol{\mathcal{S}}_2, \ldots, \boldsymbol{\mathcal{S}}_M)$ ; $\boldsymbol{\mathcal{T}}^0 = (\boldsymbol{\mathcal{T}}_1, \boldsymbol{\mathcal{T}}_2, \ldots, \boldsymbol{\mathcal{T}}_M)$ ;
// *return the job sequences* $\boldsymbol{\mathcal{S}}^0$ *and the corresponding optimal tool loadings* $\boldsymbol{\mathcal{T}}^0$ //

**Algorithm 4**
General Iterated Local Search Scheme.

```
1  Let S_0 be an initial solution obtained by a construction heuristic;
2  S* ← LocalSearch (S_0);
3  while termination criterion not satisfied do
4      S' ← Perturbation (S*, search history);
5      S*' ← LocalSearch (S' );
6      S* ← AcceptanceCriterion (S*, S*', search memory);
7  end while
8  return S*;
```

**Algorithm 5**
Local Search (LS).

1 Let $[\,r\,]_m$ denote the job processed in position $r$ on machine $m$ and let $\boldsymbol{S}_m = (\,[\,1\,]_m, [\,2\,]_m, \ldots, [\,l\,]_m)$ be the job sequence on machine $m$, and $\boldsymbol{T}_m$ the corresponding optimal tool loading on machine $m$, $m = 1, 2, \ldots, M$. Let $\boldsymbol{S}^0 = (\boldsymbol{S}^0_1, \boldsymbol{S}^0_2, \ldots, \boldsymbol{S}^0_M)$ and $\boldsymbol{T}^0 = (\boldsymbol{T}^0_1, \boldsymbol{T}^0_2, \ldots, \boldsymbol{T}^0_J)$ be the initial solution and the corresponding optimal tool loading obtained by a construction heuristic. Let $\boldsymbol{S}^{best}$ denote the best known sequence and $F(\boldsymbol{S}^{best})$ its objective value.
2 **Initialize**
3 $\boldsymbol{S}^{best} \leftarrow \boldsymbol{S}^0$
4 $\boldsymbol{T}^{best} \leftarrow \boldsymbol{T}^0$
5 **for** $m = 1, 2, \ldots, M$ **do**
6 $improved \leftarrow$ TRUE ;
7 **while** $improved$ = TRUE **do**
8 $improved \leftarrow$ FALSE ;
9 $\boldsymbol{S}' \leftarrow \boldsymbol{S}^0$
10 **for** $r_1 = 1, 2, \ldots, l-1$
11 **for** $r_2 = r_1 + 1, \ldots, l$
12 $\boldsymbol{S}'_m [\,[\,r_1\,]_m\,] \leftarrow \boldsymbol{S}^0_m [\,[\,r_2\,]_m\,]$ ;
13 $\boldsymbol{S}'_m [\,[\,r_2\,]_m\,] \leftarrow \boldsymbol{S}^0_m [\,[\,r_1\,]_m\,]$ ;
14 $\boldsymbol{T}' \leftarrow$ KTNS $(\boldsymbol{S}')$ ;
*// apply the "Keep Tool Needed Soonest"-policy using the obtained sequence $\boldsymbol{S}'$ in order to optimize the tool loading on machine m //*
15 **if** $F(\boldsymbol{S}') < F(\boldsymbol{S}^{best})$
16 $\boldsymbol{S}^{best} \leftarrow \boldsymbol{S}'$ ; *// update best known sequence //*
17 $\boldsymbol{T}^{best} \leftarrow \boldsymbol{T}'$; *// update the optimal tool loading for the best known sequence //*
18 $\boldsymbol{S}^0 \leftarrow \boldsymbol{S}^{best}$ ; *// the best known sequence replaces the current sequence //*
19 $improved \leftarrow$ TRUE ;
20 **go to** line 7 ; *// restart the swapping on the current machine //*
21 **else**
22 $\boldsymbol{S}' \leftarrow \boldsymbol{S}^0$ ; *// maintain the old current solution //*
23 **end if**
24 **end for**
25 **end for**
26 **end while**
27 **end for**
28 **return** $\boldsymbol{S}^{best}$ ; $\boldsymbol{T}^{best}$ ;

**Algorithm 6**
Time-based Perturbation (P.1$_{time}$).

Let $[r]_m$ denote the job processed in position $r$ on machine $m$ and let $\boldsymbol{\mathcal{S}}_m = ([1]_m, [2]_m, \ldots, [l]_m)$ be the job sequence on machine $m$, and $\boldsymbol{\mathcal{T}}_m$ the corresponding optimal tool loading on machine $m$, $m = 1, 2, \ldots, M$. Let $\boldsymbol{\mathcal{S}}^0 = (\boldsymbol{\mathcal{S}}^0_1, \boldsymbol{\mathcal{S}}^0_2, \ldots, \boldsymbol{\mathcal{S}}^0_M)$ and $\boldsymbol{\mathcal{T}}^0 = (\boldsymbol{\mathcal{T}}^0_1, \boldsymbol{\mathcal{T}}^0_2, \ldots, \boldsymbol{\mathcal{T}}^0_j)$ be the initial solution and the corresponding optimal tool loading obtained by the Local Search. Let $\boldsymbol{\mathcal{S}}^{best}$ denote the best known sequence and $F(\boldsymbol{\mathcal{S}}^{best})$ its objective value. Let $ts_{[r]m}$ denote the number of tool switches required immediately before processing the job in the $r^{th}$ position on machine $m$. Let $\mathcal{C}_{j,[r]m2}$ be the relocation cost when job $j$ is relocated to position $r$ on machine $m_2$.

**Initialize**

$\boldsymbol{\mathcal{S}}^{best} \leftarrow \boldsymbol{\mathcal{S}}^0$ ;

$\boldsymbol{\mathcal{T}}^{best} \leftarrow \boldsymbol{\mathcal{T}}^0$ ;

**while** *termination criterion not reached* **do**

  $\boldsymbol{\mathcal{S}}' \leftarrow \boldsymbol{\mathcal{S}}^0$ ;

  $m_1 \leftarrow argmax \{ f_m \mid m \in \mathcal{M} \wedge (\exists j \in \boldsymbol{\mathcal{S}}'_m : (|\boldsymbol{\mathcal{T}}_j| \leq C_{m_2}), m_2 \in \mathcal{M}, m_2 \neq m) \}$ ;

  *// select the machine $m_1$ with the highest completion time of the last job ($f_{m_1}$), and for which there exists at least one job that can be processed on another machine $m_2$ without violating capacity restrictions //*

  $j_1 \leftarrow argmax \{ p_{[q]m1} + sw_{m_1} \cdot ts_{[q]m1} \mid q = \{1, \ldots, |\boldsymbol{\mathcal{S}}_{m_1}|\} \wedge (|\boldsymbol{\mathcal{T}}_{[q]m1}| \leq C_{m_2}), m_2 \in \mathcal{M}, m_2 \neq m_1) \}\}$ ;

  *// select the job in position $q^*$ on machine $m_1$ with the highest processing time plus setup time, and which can be relocated to another machine $m_2$ without violating capacity restrictions //*

  **for** $m \in \mathcal{M} \mid \{m \neq m_1 \wedge (|\boldsymbol{\mathcal{T}}_{j_1}| \leq C_m)\}$

    **for** $r = (1, \ldots, |\boldsymbol{\mathcal{S}}_m|+1)$

      **if** $r = 1$ *// assigned to the first position //*

        $\mathcal{C}_{j_1,[r]m} = p_{j_1 m} + sw_m \cdot |\{t \in \boldsymbol{\mathcal{T}}_{[r]m} : t \notin \boldsymbol{\mathcal{T}}_{j_1}\}|$ ;

      **else if** $r = (|\boldsymbol{\mathcal{S}}_m|+1)$ *// assigned to the last position //*

        $\mathcal{C}_{j_1,[r]m} = p_{j_1 m} + sw_m \cdot |\{t \in \boldsymbol{\mathcal{T}}_{j_1} : t \notin \boldsymbol{\mathcal{T}}_{[r-1]m}\}|$ ;

      **else** *// assigned to any other position //*

        $\mathcal{C}_{j_1,[r]m} = p_{j_1 m} + sw_m \cdot |\{t \in \boldsymbol{\mathcal{T}}_{[r]m} : t \notin \boldsymbol{\mathcal{T}}_{j_1}\}| + sw_m \cdot |\{t \in \boldsymbol{\mathcal{T}}_{j_1} : t \notin \boldsymbol{\mathcal{T}}_{[r-1]m}\}|$ ;

      **end if**

    **end for**

  **end for**

  $(r^*, m_2) \leftarrow argmin \{\mathcal{C}_{j_1,[r]m} \mid m \in \mathcal{M} : \{m \neq m_1 \wedge (|\boldsymbol{\mathcal{T}}_{j_1}| \leq C_m)\} \wedge r = (1, \ldots, |\boldsymbol{\mathcal{S}}'_m|+1)\}$ ;

  $\boldsymbol{\mathcal{S}}'_{m_1} \leftarrow \boldsymbol{\mathcal{S}}'_{m_1} \setminus \{j_1\}$ ; *// update sequence on machine $m_1$ //*

  $\boldsymbol{\mathcal{S}}'_{m_2} \leftarrow job.insert\ (j_1, [r^*]_{m_2}, \boldsymbol{\mathcal{S}}'_{m_2})$ ; *// insert job $j_1$ in position $r^*$ on machine $m_2$ //*

  $\boldsymbol{\mathcal{T}}'_{m_1} \leftarrow \text{KTNS}\ (\boldsymbol{\mathcal{S}}'_{m_1})$ ; $\boldsymbol{\mathcal{T}}'_{m_2} \leftarrow \text{KTNS}\ (\boldsymbol{\mathcal{S}}'_{m_2})$ ;

  *// apply the "Keep Tool Needed Soonest"-policy to optimize the tool loadings on the machines $m_1$ and $m_2$ //*

  **if** $F(\boldsymbol{\mathcal{S}}') < F(\boldsymbol{\mathcal{S}}^{best})$

    $\boldsymbol{\mathcal{S}}^{best} \leftarrow \boldsymbol{\mathcal{S}}'$ ; *// update best known sequence //*

    $\boldsymbol{\mathcal{T}}^{best} \leftarrow \boldsymbol{\mathcal{T}}'$ ; *// update the optimal tool loading for the best known sequence //*

  **end if**

  $(\boldsymbol{\mathcal{S}}^{*\prime}, \boldsymbol{\mathcal{T}}^{*\prime}) \leftarrow Local.Search\ (\boldsymbol{\mathcal{S}}', \boldsymbol{\mathcal{S}}'_{m_1}, \boldsymbol{\mathcal{S}}'_{m_2})$

  *// apply the local search to the machines $m_1$ and $m_2$ of the perturbed solution //*

  $\boldsymbol{\mathcal{S}}^0 \leftarrow \boldsymbol{\mathcal{S}}^{*\prime}$ *// best solution from local search becomes new current solution //*

**end while**

**return** $\boldsymbol{\mathcal{S}}^{best}$ ; $\boldsymbol{\mathcal{T}}^{best}$ ;

**Algorithm 7**
Tool-Switch-based Perturbation ($P.1_{switch}$).

Let $[r]_m$ denote the job processed in position $r$ on machine $m$ and let $\boldsymbol{\mathcal{S}}_m = ([1]_m, [2]_m, \ldots, [l]_m)$ be the job sequence on machine $m$, and $\boldsymbol{\mathcal{T}}_m$ the corresponding optimal tool loading on machine $m$, $m = 1, 2, \ldots, M$. Let $\boldsymbol{\mathcal{S}}^0 = (\boldsymbol{\mathcal{S}}^0_1, \boldsymbol{\mathcal{S}}^0_2, \ldots, \boldsymbol{\mathcal{S}}^0_M)$ and $\boldsymbol{\mathcal{T}}^0 = (\boldsymbol{\mathcal{T}}^0_1, \boldsymbol{\mathcal{T}}^0_2, \ldots, \boldsymbol{\mathcal{T}}^0_J)$ be the initial solution and the corresponding optimal tool loading obtained by the Local Search. Let $\boldsymbol{\mathcal{S}}^{best}$ denote the best known sequence and $F(\boldsymbol{\mathcal{S}}^{best})$ its objective value. Let $TS_m$ denote the total number of tool switches on machine $m$ and $ts_{[r]_m}$ the number of tool switches required immediately before processing job in the $r^{th}$ position on machine $m$. Let $\boldsymbol{\mathcal{C}}_{j,[r]m2}$ be the relocation cost when job $j$ is relocated to position $r$ on machine $m_2$.

**Initialize**

$\boldsymbol{\mathcal{S}}^{best} \leftarrow \boldsymbol{\mathcal{S}}^0$ ;

$\boldsymbol{\mathcal{T}}^{best} \leftarrow \boldsymbol{\mathcal{T}}^0$ ;

**while** *termination criterion not reached* **do**

  $\boldsymbol{\mathcal{S}}' \leftarrow \boldsymbol{\mathcal{S}}^0$ ;

  $m_1 \leftarrow argmax \{ TS_m \mid m \in \boldsymbol{\mathcal{M}} \wedge (\exists j \in \boldsymbol{\mathcal{S}}'_m : (|T_j| \leq C_{m2}), m_2 \in \boldsymbol{\mathcal{M}},\ m_2 \neq m) \}$ ;

  *// select the machine $m_1$ with the highest number of tool switches ($TS_{m_1}$), and for which there exists at least one job that can be processed on another machine $m_2$ without violating capacity restrictions //*

  $j_1 \leftarrow argmax \{ ts_{[q]m1} \mid q = \{1, \ldots, k\} \wedge (|T_{[q]m1}| \leq C_{m2}), m_2 \in \boldsymbol{\mathcal{M}},\ m_2 \neq m_1) \}$ ;

  *// select the job in position $q^*$ on machine $m_1$ with the largest number of tool switches, and which can be relocated to another machine $m_2$ without violating capacity restrictions //*

  **for** $m \in M \mid \{m \neq m_1 \wedge (|T_{j1}| \leq C_m)\}$

    **for** $r = (1, \ldots, |\boldsymbol{\mathcal{S}}_m| + 1)$

      **if** $r = 1$ *// assigned to the first position //*

        $\boldsymbol{\mathcal{C}}_{j1,[r]m} = |\{t \in \boldsymbol{\mathcal{T}}_{[r]m} : t \notin T_{j1}\}|$ ;

      **else if** $r = (|\boldsymbol{\mathcal{S}}_m| + 1)$ *// assigned to the last position //*

        $\boldsymbol{\mathcal{C}}_{j1,[r]m} = |\{t \in \boldsymbol{\mathcal{T}}_{j1} : t \notin T_{[r-1]m}\}|$ ;

      **else** *// assigned to any other position //*

        $\boldsymbol{\mathcal{C}}_{j1,[r]m} = |\{t \in \boldsymbol{\mathcal{T}}_{[r]m} : t \notin T_{j1}\}| + |\{t \in \boldsymbol{\mathcal{T}}_{j1} : t \notin \boldsymbol{\mathcal{T}}_{[r-1]m}\}|$ ;

      **end if**

    **end for**

  **end for**

  $(r^*, m_2) \leftarrow argmin \{\boldsymbol{\mathcal{C}}_{j1,[r]m} \mid m \in \boldsymbol{\mathcal{M}} : \{m \neq m_1 \wedge (|\boldsymbol{\mathcal{T}}_{j1}| \leq C_m)\} \wedge r = (1, \ldots, |\boldsymbol{\mathcal{S}}'_m| + 1)\}$ ;

  $\boldsymbol{\mathcal{S}}'_{m1} \leftarrow \boldsymbol{\mathcal{S}}'_{m1} \setminus \{j_1\}$ ; *// update sequence on machine $m_1$ //*

  $\boldsymbol{\mathcal{S}}'_{m2} \leftarrow job.insert\ (j_1, [r^*]_{m2}, \boldsymbol{\mathcal{S}}'_{m2})$ ; *// insert job $j_1$ in position $r^*$ on machine $m_2$ //*

  $\boldsymbol{\mathcal{T}}'_{m1} \leftarrow \text{KTNS}\ (\boldsymbol{\mathcal{S}}'_{m1})$ ; $\boldsymbol{\mathcal{T}}'_{m2} \leftarrow \text{KTNS}\ (\boldsymbol{\mathcal{S}}'_{m2})$ ;

  *// apply the "Keep Tool Needed Soonest"-policy to optimize the tool loadings on the machines $m_1$ and $m_2$ //*

  **if** $F(\boldsymbol{\mathcal{S}}') < F(\boldsymbol{\mathcal{S}}^{best})$

    $\boldsymbol{\mathcal{S}}^{best} \leftarrow \boldsymbol{\mathcal{S}}'$ ; *// update best known sequence //*

    $\boldsymbol{\mathcal{T}}^{best} \leftarrow \boldsymbol{\mathcal{T}}'$ ; *// update the optimal tool loading for the best known sequence //*

  **end if**

  $(\boldsymbol{\mathcal{S}}^{*\prime}, \boldsymbol{\mathcal{T}}^{*\prime}) \leftarrow Local.Search\ (\boldsymbol{\mathcal{S}}', \boldsymbol{\mathcal{S}}'_{m1}, \boldsymbol{\mathcal{S}}'_{m2})$

  *// apply the local search to the machines $m_1$ and $m_2$ of the perturbed solution //*

  $\boldsymbol{\mathcal{S}}^0 \leftarrow \boldsymbol{\mathcal{S}}^{*\prime}$ *// best solution from local search becomes new current solution //*

**end while**

**return** $\boldsymbol{\mathcal{S}}^{best}$ ; $\boldsymbol{\mathcal{T}}^{best}$ ;

## References


Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*, 345–378.

Bard, J. F. (1988). A Heuristic for Minimizing the Number of Tool Switches on a Flexible Machine. *IIE Transactions, 20*, 382–391.

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

Burger, A. P., Jacobs, C. G., van Vuuren, J. H., & Visagie, S. E. (2015). Scheduling multi-colour print jobs with sequence-dependent setup times. *Journal of Scheduling, 18*, 131–145.

Calmels, D., Rajendran, C., & Ziegler, H. (2019). Heuristics for Solving the Job Sequencing and Tool Switching Problem with Non-identical Parallel Machines. In B. Fortz, & M. Labbé (Eds.), *Operations research proceedings 2018, operations research proceedings* (pp. 459–465). Cham: Springer International Publishing.

Calmels, D. (2019). The job sequencing and tool switching problem: State-of-the-art literature review, classification, and trends. *International Journal of Production Research, 57*, 5005–5025.

Calmels, D. (2021). A comparison of different mathematical models for the job sequencing and tool switching problem with non-identical parallel machines. *International Journal of Operational Research* Advance online publication:. https://doi.org/10.1504/IJOR.2021.10034075.

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

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

Dang, Q.-. V., van Diessen, T., Martagan, T., & Adan, I. (2021). A matheuristic for parallel machine scheduling with tool replacements. *European Journal of Operational Research, 291*, 640–660.

Djellab, H., Djellab, K., & Gourgand, M. (2000). A new heuristic based on a hypergraph representation for the tool switching problem. *International Journal of Production Economics, 64*, 165–176.

Fathi, Y., & Barnette, K. W. (2002). Heuristic procedures for the parallel machine problem with tool switches. *International Journal of Production Research, 40*, 151–164.

Follonier, J.-. P. (1994). Minimization of the number of tool switches on a flexible manufacturing machine. *Belgian Journal of Operations Research, Statistics and Computer Science, 34*, 55–72.

Ghiani, G., Grieco, A., & Guerriero, E. (2007). An exact solution to the TLP problem in an NC machine. *Robotics and Computer-Integrated Manufacturing, 23*, 645–649.

Ghrayeb, O. A., Phojanamongkolkij, N., & Finch, P. R. (2003). A mathematical model and heuristic procedure to schedule printed circuit packs on sequencers. *International Journal of Production Research, 41*, 3849–3860.

Gökgür, B., Hnich, B., & Özpeynirci, S. (2018). Parallel machine scheduling with tool loading: A constraint programming approach. *International Journal of Production Research, 54*, 1–17.

Hirvikorpi, M., Nevalainen, O. S., & Knuutila, T. (2006). Job ordering and management of wearing tools. *Engineering Optimization, 38*, 227–244.

Hirvikorpi, M., Salonen, K., Knuutila, T., & Nevalainen, O. S. (2006). The general two-level storage management problem: A reconsideration of the KTNS-rule. *European Journal of Operational Research, 171*, 189–207.

Kravchenko, S. A., & Werner, F. (2011). Parallel machine problems with equal processing times. A survey. *Journal of Scheduling, 14*, 435–444.

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

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. In P. Hammer, E. Johnson, & B. Korte (Eds.), *Annals of discrete mathematics, studies in integer programming* (pp. 343–362). Elsevier.

Liu, J., & Reeves, C. R. (2001). Constructive and composite heuristic solutions to the P//ΣCi scheduling problem. *European Journal of Operational Research, 132*, 439–452.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2019). Iterated Local Search: Framework and Applications. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of metaheuristics, international series in operations research & management science* (pp. 129–168). Cham: Springer International Publishing.

Özpeynirci, S., Gökgür, B., & Hnich, B. (2016). Parallel machine scheduling with tool loading. *Applied Mathematical Modelling, 40*, 5660–5671.

Paiva, G. S., & Carvalho, M. A. M. (2017). Improved heuristic algorithms for the Job Sequencing and Tool Switching Problem. *Computers & Operations Research, 88*, 208–219.

Pessoa, L. S., & Andrade, C. E. (2018). Heuristics for a flowshop scheduling problem with stepwise job objective function. *European Journal of Operational Research, 266*, 950–962.

Privault, C., & Finke, G. (2000). k-server problems with bulk requests: An application to tool switching in manufacturing. *Annals of Operations Research, 96*, 255–269.

Raduly-Baka, C., Knuutila, T., & Nevalainen, O. (2005).Minimising the number of tool switches with tools of different sizes. Turku Centre for Computer Science: Turku;

Raduly-Baka, C., & Nevalainen, O. S. (2015). The modular tool switching problem. *European Journal of Operational Research, 242*, 100–106.

Rajendran, C. (1994). A heuristic for scheduling in flowshop and flowline-based manufacturing cell with multi-criteria. *International Journal of Production Research, 32*, 2541–2558.

Sarmadi, H., & Gholami, S. (2011). Modeling of tool switching problem in a flexible manufacturing cell: With two or more machines. In G. Lee (Ed.), *International Conference on Mechanical and Electrical Technology, 3rd, (ICMET-London 2011)Three Park Avenue New York, NY 10016-5990: 1–3* (pp. 2345–2349). ASME. Volumes.

Shirazi, R., & Frizelle, G. (2001). Minimizing the number of tool switches on a flexible machine: An empirical study. *International Journal of Production Research, 39*, 3547–3560.

Talbi, E.-. G. (2009). Metaheuristics. *From design to implementation*. Hoboken, N.J.: John Wiley & Sons.

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, 36*, 767–777.

Tzur, M., & Altmann, A. (2004). Minimization of tool switches for a flexible manufacturing machine with slot assignment of different tool sizes. *IIE Transactions, 36*, 95–110.

Van Hop, N., & Nagarur, N. N. (2004). The scheduling problem of PCBs for multiple non-identical parallel machines. *European Journal of Operational Research, 158*, 577–594.

Van Hop, N. (2005). The tool-switching problem with magazine capacity and tool size constraints. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 35*, 617–628.