Les Cahiers du GERAD ISSN: 0711–2440

Scheduling identical parallel machines with tooling constraints

A.C. Beezão, J.-F. Cordeau,
G. Laporte, H.H. Yanasse

G–2015–134

December 2015

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Scheduling identical parallel machines with tooling constraints

Andreza Cristina Beezão [a]
Jean-François Cordeau [b]
Gilbert Laporte [c]
Horacio Hideki Yanasse [d]

[a] *Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, Brazil*

[b] *GERAD & Canada Research Chair in Logistics and Transportation, HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

[c] *GERAD & Canada Research Chair in Distribution Management, HEC Montréal, Montréal (Québec) Canada, H3T 2A7*

[d] *Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo, São José dos Campos, Brazil*

andreza.beezao@gmail.com
jean-francois.cordeau@hec.ca
gilbert.laporte@cirrelt.ca
horacio.yanasse@unifesp.br

December 2015

Les Cahiers du GERAD
G–2015–134

Abstract: We model and solve the problem of sequencing a set of jobs with specified processing times and tool requirements on a set of identical parallel machines. Decisions concern the assignment of jobs to machines, their sequencing, and the allocation of tools on each machine. The objective function minimizes the makespan. We propose an adaptive large neighborhood search metaheuristic in which the destroy and repair operators exploit the structures of two well-known and related combinatorial optimization problems, namely the parallel machine scheduling problem and the job sequencing and tool switching problem on a single machine. Computational experiments conducted on two data sets of 1440 instances show that our algorithm produces excellent results and outperforms existing heuristics.

Key Words: Flexible manufacturing systems, parallel machines, tooling constraints, combinatorial optimization, adaptive large neighborhood search.

Acknowledgments: This work was partially supported by the FAPESP (São Paulo Research Foundation), by the Canadian Natural Science and Engineering Research Council under grants 2014-04959 and 2015-06189 and by the CNPq-Brazil (National Council for Scientific and Technological Development). This support is gratefully acknowledged. We also thank CIRRELT and Calcul Québec for providing the computing facilities used for our experiments.

1 Introduction

Competitiveness and market uncertainty require that manufacturing companies become increasingly flexible. Flexible manufacturing systems (FMS) arise as an alternative to incorporate versatility, efficiency and high levels of flexibility in mass production. They enable the processing of a variety of products when their components are properly scheduled. An FMS is a highly automated production system consisting of a computer-controlled set of multipurpose workstations, storage buffers and automated guided vehicles (AGV) (Novas and Henning, 2014). Machines require tools, which are grouped by types and stored in tool magazines with a limited number of slots, to carry out operations on parts.

In order to obtain optimal or near-optimal solutions for the scheduling of realistic FMS, the following elements must be handled simultaneously: machine loading, part routing, tool planning and allocation, AGV loading and routing, and task timing decisions. Machine loading consists of the assignment of processing tasks to machines, considering their limited associated buffers. Part routing generates the sequence of machines that each part must visit in order to be manufactured. Tool planning determines the number of tools of each type to support the production requirements, while the tool allocation problem tackles the assignment of tools to different tool magazines. AGV loading and routing establishes the route to be followed by the device, as well as the amount of tools and parts to be transported. Finally, task timing decisions specify the start and completion times of each machining activity.

The retrieval of tools from the storage, their transportation, loading and calibration represent about 25% to 30% of the total fixed and variable costs in an FMS (Cumings, 1986; Tomek, 1986; Ayres, 1988). Moreover, since tool switches between magazines and the tool storage are inevitable in most practical FMS, tooling constraints become a key issue in increasing the FMS productivity and the usage of its resources.

In this paper, we focus on the machine loading and on the tool allocation components of an FMS. More precisely, the *Identical Parallel Machines Problem with Tooling Constraints* (IPMTC) concerns the processing of a set of jobs with given processing times and tool requirements on a set of identical parallel machines in order to minimize the makespan, that is, the completion time to process the set of jobs using all the available machines. Our study makes two main contributions: (i) it develops a new solution method for the IPMTC that outperforms existing algorithms and (ii) it provides two new sets of instances for the IPMTC containing up to 200 jobs, 10 machines and 40 tools. The solution strategy can be divided into two main steps. First, we combine a heuristic for the *Job Sequencing and Tool Switching Problem* (SSP) on a single machine with a MIP model to determine an initial solution for the IPMTC. Then, we effectively search the solution space by implementing an adaptive large neighborhood search (ALNS) metaheuristic. We have designed several new operators taking advantage of the structure of the problem and we have also adapted existing operators for the *Vehicle Routing Problem* and the *Lot Sizing Problem*.

Our algorithm is compared with that of Fathi and Barnette (2002). To the best of our knowledge, this is the most recent heuristic in which tools appear to be the critical component in the context of the previously described IPMTC, which reinforces the academic importance of our developments.

The remainder of the paper is organized as follows. The problem is formally described in Section 2, which also reviews the literature on the IPMTC and related problems combining machine loading and tool allocation in an FMS. Section 3 presents the details of the ALNS metaheuristic, while computational results are reported in Section 4. Finally, we provide conclusions and future research directions in Section 5.

2 Problem description and literature review

In the IPMTC the aim is to process a set of jobs $J = \{1, \ldots, n\}$ in an FMS with m identical parallel machines, with the objective of minimizing the time at which all jobs are finished. Each machine is able to process all jobs but each job must be assigned to only one machine. There are no precedence relationships among the jobs and interruptions are not permitted. The term *identical* means that the processing times p_j $(j \in J)$ are the same on each machine, as well as the set of available tools $T = \{1, \ldots, l\}$ and the limited capacity C of the tool magazines. Machine buffers have a large capacity and their limitations are not considered. The

time $\bar{t}$ required to make a tool switch is constant for all tools on all machines, and the total switching time on a given machine is obtained by multiplying $\bar{t}$ by the respective number of tool switches. Therefore, the makespan Δ is given by the sum of the job processing times and the corresponding switching time on the more time-consuming machine $m^* \in M = \{1, \ldots, m\}$. Decisions concern the assignment of jobs to machines, their sequencing, and the allocation of tools on each machine. The first component is commonly called the *Parallel Machine Scheduling Problem* (PM) (also known as *Multiprocessor Scheduling Problem* or *Identical Parallel Machines Scheduling Problem*), and the second and third ones represent the classical SSP.

Various aspects related to FMS scheduling problems have been studied in the last decades. From 1970 to 1990, many authors analyzed general features and particularities of an FMS, such as its design, planning, scheduling and control. The interested reader is referred to Coffman and Denning (1973), Stecke (1983), Stecke (1985), Gray et al. (1988), Borenstein and Becker (1994), Crama and Oerlemans (1996) and Crama (1997). Other studies were more specific and identified correct tool management as a critical issue for the effective implementation of an FMS. Melnyk et al. (1989) used a simulation model to show how the level of tooling availability has a significant impact on the shop floor performance, and Veeramani et al. (1992) reviewed studies in academia and industry concerning different facets of systems with tooling constraints. As a main conclusion, the authors highlight the need for solution methods that combine mathematical programming and simulation techniques to build realistic models for such systems. Gray et al. (1993) suggested an integrated conceptual framework for resource planning in order to avoid underutilization of the system. On a related note, Mohamed and Bernardo (1997) claimed that an inadequate number of tools contributes to a decrease in FMS productivity.

The problem of optimizing the insertions and removals of tools in flexible manufacturing machines was tackled predominantly by heuristic methods. Both PM and SSP are known to be NP-hard (Garey and Johnson, 1979; Crama et al., 1994) and can be considered as special cases of the IPMTC. We thus conclude that the IPMTC is NP-hard and, therefore, intrinsically difficult to solve. To cope with this, researchers have solved the IPMTC through decomposition approaches, where an SSP is solved on each single machine based on a corresponding assignment of jobs previously determined by a PM algorithm.

The remainder of this section considers three main aspects: first, the parallel machine problem, then the main contributions to the SSP and, finally, the IPMTC under different contexts and objective functions.

A common feature of several heuristics in the field of sequencing and scheduling is the notion of list processing, which first defines a rule for assigning priorities to the still unscheduled jobs and, in subsequent stages, schedules the first job from the ordered list to the least loaded machine. Numerous priority dispatching rules have been proposed. More than thirty years ago, Panwalkar and Islander (1977) pointed out the relationship between the way the lists are constructed and the accuracy of the resulting algorithms. The *Knuth-Kleitman* and the *Longest Processing Time* (LPT) heuristics are widely applied for the conventional parallel machine scheduling (Parker, 1995; Lee, 1991; Lee et al., 2000; Hwang et al., 2005). Both methods are based on sorting a part or the full set of jobs in non-increasing order of their processing times and, then, applying the list scheduling algorithm mentioned above. Randomly generated lists are adopted as subroutines of more complete procedures. Fathi and Barnette (2002), for example, minimize the makespan by using the multiple-start greedy heuristic (Crama et al., 1994) on fifty random lists of jobs. In turn, the *Multifit* (Coffman et al., 1978) is based on the *First Fit Decreasing* technique from *Bin Packing* problems and is comparable to LPT in terms of computational efficiency. Most of the research on parallel machines with a makespan minimization objective has focused on the worst-case analysis of algorithms, which relies on the improvement or combination of well established methods as a strategy to reach tighter bounds. Lee and Massey (1988) combine the Multifit with the LPT, while Koulamas and Kyparisis (2009) proposed a modified LPT heuristic for the two uniform machine makespan minimization problem.

The SSP in the context of a single flexible manufacturing machine was introduced and formulated by Tang and Denardo (1988), with a uniform switching time and indefinite tool life. When the sequence of jobs is not prespecified, the problem aims at scheduling the jobs while loading the tools to minimize the total number of tool switches. Even though the SSP is admittedly important in the manufacturing arena, Shirazi and Frizelle (2001) analyzed seven medium and large manufacturing companies in the UK to conclude that none have applied a structured method to optimize the management of tools, entrusting that task to experienced

employees. Tang and Denardo (1988) consider the job scheduling and the tool loading as two independent problems and propose the *Keep Tool Needed Soonest* (KTNS) policy, which is a polynomial-time algorithm that determines the optimal tool loading plan for a given sequence of jobs. Solving the SSP via mathematical modeling is often impractical. The integer linear programming formulations presented by Tang and Denardo (1988) and by Laporte et al. (2004) are able to solve only small and medium-sized instances. To the best of our knowledge, three exact methods were developed for the SSP in the last decades. They are due to Laporte et al. (2004), Yanasse et al. (2009) and Ghiani et al. (2010) and consist of enumerative procedures such as branch-and-bound and branch-and-cut. Problems with up to 45 jobs were solved to optimality by Ghiani et al. (2010), based on modeling the problem as a non-linear *Traveling Salesman Problem* (TSP). Kabir (2011) proposed a dynamic programming algorithm and two sets of heuristics to solve the SSP in the context of limited tool life and multiple processing plans, where there may exist different configurations of tools able to process an arbitrary job. The KTNS does not guarantee optimal results under these circumstances. Moreover, such assumptions affect the minimization of processing times in the objective function, since a set of tools may be faster in processing a job than using other sets. As a consequence, the authors tackle the SSP by minimizing the makespan, instead of only the number of tool switches. More recently, Catanzaro et al. (2015) proposed a new integer linear programming formulation for the SSP and were able to provide tighter lower bounds to the problem. Hybrid algorithms and heuristics for the SSP include tabu search, job grouping and constructive strategies, as can be seen in Crama et al. (1994), Privault and Finke (1995), Gómez and Lorena (1998), Al-Fawzan and Al-Sultan (2002), Zhou et al. (2005), Salonen et al. (2006), Amaya et al. (2008), Amaya et al. (2012) and Chaves et al. (2012).

The IPMTC was first investigated by Berrada and Stecke (1986) and Koulamas (1991). Later, Hertz and Widmer (1996) proposed an algorithm based on tabu search, a popular approach to combinatorial optimization problems at that time. Since then, and to the best of our knowledge, Fathi and Barnette (2002) are the only authors to have studied the IPMTC under the above-mentioned assumptions. In short, they proposed three heuristic methods to tackle the assignment of jobs to machines. The first one is a local improvement algorithm where several neighborhoods and search strategies are applied. The second procedure is based on the LPT routine and the last one is an adaptation of a constructive heuristic implemented for the *k-Traveling Salesman Problem*. As a second step, the KTNS policy is applied to optimize the tool loading problem, that is, the assignment of tools to machines respecting the capacity of the magazines. The LPT procedure is consistently the worst in terms of makespan since it does not consider restrictions related to tool requirements and switches.

Broader studies consider the FMS under different circumstances. Atan and Pandit (1996) and Kumar and Sridharan (2009) looked at the FMS under a tool-sharing environment aiming at reducing the tool inventory and improving the utilization of tools. Non-identical machines and tools were considered by Keung et al. (2001) and Buyurgana et al. (2004). Finally, Chan and Swarnkar (2006), Das et al. (2009), Zeballos (2010) and Novas and Henning (2014) consider distinct levels of management of FMS, where objective functions concern machining and setup costs, part routing and material handling movements.

3 ALNS metaheuristic

The ALNS metaheuristic framework was introduced by Ropke and Pisinger (2006) and has yielded excellent results on a wide range of problems. In its classical implementation, each ALNS iteration consists of the application of one destroy and one repair operator. These are chosen independently of each other according to a roulette wheel selection mechanism based on a weight that depends on the success of each operator in previous iterations (Pisinger and Ropke, 2007). The search is divided into a number of segments of α iterations. At the end of a segment, the weights are updated according to the scores accumulated during the last α iterations and proportionally to a reaction factor r that controls how quickly the weight adjustment reacts to changes in the success of the operators. After the application of a destroy and a repair operator, their scores are increased by a value α_1, α_2 or α_3 if the search has reached a new best solution, a solution better than the current one or a non-improving, but still acceptable, solution, respectively. Let $\Delta(s)$ denote the makespan in solution s. A solution s' is always acceptable if $\Delta(s') < \Delta(s)$, and acceptable with probability $e^{-(\Delta(s')-\Delta(s))/\theta}$, where θ is the temperature, otherwise. The initial temperature θ_0 is given by $(0.05\Delta_0)/\ln 2$,

so that a solution that is 5% worse than the initial makespan Δ_0 is accepted with probability 0.5. Then, given h operators with weight ω_i, where $i \in H = \{1, \ldots, h\}$, operator $j \in H$ is selected with probability $\omega_j / \sum_{i \in H} \omega_i$.

We have implemented nine destroy and five repair operators to take advantage of the characteristics of both the PM and SSP. The master framework used in the ALNS is based on a simulated annealing mechanism and the stopping criterion is determined by a maximum number of iterations and a minimum temperature θ equal to 0.01, which decreases from θ_0 at each iteration according a cooling factor $c \in (0, 1)$. The main steps of the ALNS are presented in Algorithm 1. Further details related to ALNS assumptions and specifications can be found in Ropke and Pisinger (2006) and Pisinger and Ropke (2007).

Algorithm 1 Basic steps of the ALNS framework.

```
Let s be an initial solution;
Initialize scores α1, α2, α3;
s* ← s;
repeat
    Choose destroy and repair operators, N⁻ and N⁺, according to the roulette wheel mechanism;
    Generate a new solution s′ from s by applying N⁻ and N⁺;
    if Δ(s′) < Δ(s) then
        s ← s′;
    else if Δ(s′) > Δ(s) and s′ can be accepted then
        s ← s′;
    end if
    if Δ(s′) < Δ(s*) then
        s* ← s′;
    end if
until the stopping condition is met;
return s* (best solution found).
```

3.1 Initial solution

Our method to construct an initial solution combines the near-optimal heuristic for the SSP of Chaves et al. (2012) with a MIP model that minimizes the makespan by breaking the SSP solution into m segments, one for each machine.

The SSP heuristic consists of a constructive phase followed by an improvement phase. The first phase is based on a graph where the vertices correspond to tools and there is an edge $j = (t_1, t_2)$ connecting vertices t_1 and t_2 if and only if tools t_1 and t_2 are required for the same job j. The algorithm prioritizes the removal of edges from low degree vertices, in order to produce isolated vertices. In other words, this strategy generates a processing order of jobs, $s = \{s_1, \ldots, s_n\}$, while removing from the magazine the tools that are no longer required and that will not incur more tool switches. The second phase consists in an improvement algorithm inspired by the *Iterated Local Search* (ILS) metaheuristic, commonly applied with the intent of generating new solutions from the perturbation of current ones. The procedure starts by using the KTNS policy to establish the number of tool switches related to the solution s. A local optimal solution $\hat{s}$ is then determined by the exchange of each pair of jobs belonging to s. In the sequence, the method modifies the execution order of $\beta\%$, $\beta \in [20, 40]$, arbitrary pairs of jobs from $\hat{s}$, resulting in a new solution s' for the local search. The jobs are exchanged in pairs, one more time, to generate the local optimal solution $\hat{s}'$. The algorithm continues from the best solution between $\hat{s}'$ and $\hat{s}$, and stops after 300 iterations.

As the resulting SSP solution is already well balanced in terms of the number of tool switches, the makespan Δ to be optimized by the MIP model considers only processing times. This choice yields a high quality starting point for the ALNS within a short computing time. Without loss of generality, assume that $s = \{s_1, \ldots, s_n\}$ is the processing order of jobs generated by the SSP heuristic and let x_{ij} be a binary variable equal to one if there is a segment starting with the job in position i and finishing with the job in position j, where $i, j \in \{1, \ldots, n\}$. The model is as follows:

$$\text{minimize } \Delta \tag{1}$$

subject to

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = m \tag{2}$$

$$\sum_{j=1}^{n} x_{1j} = 1 \tag{3}$$

$$\sum_{i=1}^{n} x_{in} = 1 \tag{4}$$

$$\sum_{j=1}^{n} x_{ij} \leq 1 \qquad i = \{2, \ldots, n\} \tag{5}$$

$$\sum_{i=1}^{n} x_{ij} \leq 1 \qquad j = \{1, \ldots, n-1\} \tag{6}$$

$$\sum_{k=j+1}^{n} x_{(j+1)k} \geq x_{ij} \qquad i, j = \{1, \ldots, n-1\} \tag{7}$$

$$\left(\sum_{k=i}^{j} p_{s_k}\right) x_{ij} \leq \Delta \qquad i, j = \{1, \ldots, n\} \tag{8}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = \{1, \ldots, n\} \tag{9}$$

The objective function (1) minimizes the makespan. Constraint (2) guarantees that the number of segments is equal to the number of machines, while constraints (3) – (6) define the possible extremities of a segment. Continuity is ensured by constraints (7): if there is a segment from i to j, there must exist $k \in \{j+1, \ldots, n\}$ such that $x_{(j+1)k} = 1$. The makespan is finally defined by constraints (8) and constraints (9) impose restrictions on the decision variable. After obtaining an assignment of jobs to the set of parallel machines, the last step consists of the management of tools, which is determined by the application of the KTNS policy on each individual machine. We denote by ALNS-B the ALNS algorithm initialized with this procedure. We have also experimented with the ALNS by using randomly generated initial solutions, which we denote by ALNS-R.

3.2 Destroy and repair operators

In the context of the IPMTC, destroy (repair) operators consist of different ways of removing (inserting) a specified number q of jobs from (into) the schedule for the set of parallel machines. This section describes nine destroy and five repair operators designed to achieve both intensification and diversification, as suggested by Ropke and Pisinger (2006).

3.2.1 Random removal

The simple removal procedure removes q jobs selected at random from the current solution.

3.2.2 Shaw removal

This operator was first proposed by Shaw (1997). It removes jobs that are considered similar and whose reinsertion could lead to better solutions. The measure of similarity considered here is the absolute difference between the contribution of each job to the objective function. The first job j' is selected at random from s. Then, the algorithm computes the relatedness measure $R(j', j)$ between j' and all the remaining jobs $j \in \bar{J} = J \backslash \{j'\}$ to choose the next job to be removed. These removals are also randomized, but controlled by a parameter $\delta \geq 1$ that avoids determinism in destroy operators, as explained in the steps 8 to 11 of the Algorithm 2.

Algorithm 2 Randomness during the removal phase

input $\delta \geq 1$;
Let s be the current solution;
Let D be a set of removed jobs;
while $|D| < q$ **do**
  Let j' be a job selected at random from s;
  Let $\bar{J}$ be the set of remaining jobs;
  Sort $\bar{J}$ such that $j_1 < j_2$ if $R(j', j_1) < R(j', j_2)$;
  Choose a random number y from $[0, 1]$;
  $j'' \leftarrow \lfloor y^{\delta} |\bar{J}| \rfloor$;
  $D \leftarrow D \cup \{j''\}$;
  Remove job j'' from the solution s;
end while

3.2.3 Worst removal

This operator removes jobs with a high cost in the current solution s in the hope of reinserting them in better positions. Let $c(s, j') = f(s) - f_{-j'}(s)$ be the cost of a job j', where f_{-j} represents the makespan of solution s when job j is removed. The operator performs a removal in the list of jobs sorted, in a non-increasing order, according their costs. The randomness in the selection of jobs is ensured by a parameter $\delta \geq 1$ (Algorithm 2).

3.2.4 Largest deviation removal

The best solution of the assignment of jobs to machines is not necessarily a good solution for the complete IPMTC, since it does not consider tooling constraints. To cope with this, the operator removes jobs that generate large deviations in the makespan when considering the sum of processing times and tool switches, in comparison with only processing times. We estimate the number of switches between two jobs j_1 and j_2 by using the simple lower bound $LB = \max\{0, |T_{j_1} \cup T_{j_2}| - C\}$, where T_j represents the set of tools required for the processing of job j.

3.2.5 Random machine removal

In order to diversify the search, this operator selects at random a machine and removes jobs assigned to it. The method proceeds to a new randomly chosen machine if the number of jobs removed is smaller than q.

3.2.6 Pseudo-random machine removal

The first machine and job are randomly selected, while the next removals prioritize jobs assigned to this machine and most related with those already removed. The concept of relatedness is the same used in the Shaw removal operator.

3.2.7 Critical machine removal

This operator starts by choosing the most time-consuming machine and follows the same removal scheme as for the previous operator. The objective is to impose the removal of variables exploiting the structure of the current solution, that is, which increase considerably the objective function value.

3.2.8 Neighborhood graph removal

The aim of this operator is the removal of jobs currently assigned to an improper machine. We define a complete directed and weighted graph where the weight of each edge (j_1, j_2) represents the best objective function value reached so far in a solution where j_1 and j_2 are processed consecutively and by the same machine. The score of a node (job) is given by summing the weight of edges incident to it and jobs with high scores are likely to be selected for removal. A parameter $\delta \geq 1$ controls the randomness in the removals performed (Algorithm 2).

3.2.9 Request graph removal

Similarly to the last method, the cost of an edge (j_1, j_2) belonging to the request graph is the number of times jobs j_1 and j_2 are processed by the same machine in the 100 best solutions found so far in the search. Jobs with small scores are likely to be removed first, in order to prioritize removals of unpromising edges. Removals are randomized and controlled by a parameter $\delta \geq 1$ (Algorithm 2).

3.2.10 Greedy$_p$ insertion

For this operator, q jobs are inserted according to the removal order into positions that generate the minimum increase in total processing times.

3.2.11 Greedy$_\Delta$ insertion

This operator is slightly different from the previous one. It repeatedly inserts jobs respecting the removal order and in the position that least increases the makespan Δ in the current solution.

3.2.12 Look-ahead greedy$_p$ insertion

This operator inserts the job having the minimum global cost position considering processing times. More specifically, assume that $\epsilon_p^+(j)$ is the change in processing times due to the insertion of job $j \in Q$ in the position that generates the minimum increase in processing times. The next insertion concerns job $\tilde{j}$ such that $\epsilon_p^+(\tilde{j}) = \min_{j \in Q} \epsilon_p^+(j)$, where Q is the set of removed jobs.

3.2.13 Look-ahead greedy$_\Delta$ insertion

This operator generalizes the deep greedy$_p$ insertion by inserting the job having the minimum global cost position considering the makespan criterion.

3.2.14 2-Regret insertion

Greedy movements postpone the allocation of critical jobs to the final iterations, where there is not much freedom of action (Pisinger and Ropke, 2007). To deal with this, the regret movement chooses jobs for which the difference between their insertion in the first and second cheapest position is the largest.

4 Computational experiments

This section presents extensive computational experiments conducted to evaluate the performance of the ALNS. We first report the generation of two sets of instances, followed by the tuning phase and the results obtained by the proposed method.

4.1 Instance description

Since the set of 37 instances used by Fathi and Barnette (2002) was not available, we have tested our method on two new data sets. Initially, we used a set of 1440 instances indicated as IPMTC-I and adapted from Yanasse et al. (2009), which were themselves generated for the SSP following the scheme presented in Crama et al. (1994). The adaptation started with the generation of random processing times p_j $(j \in J)$ using a uniform distribution over the interval [1, 59]. The number of machines ranges from 2 to 4 and the number of jobs is either 8, 15 or 25 for $m = 2$, 15 or 25 for $m = 3$, and 25 for $m = 4$. Each machine-job setting results in two machine-job-tool combinations, depending on the number of required tools, which is equal to 15 or 20. We define three values of switching times given by $s_0 = \text{rand}(p_{min}, p_{max})$, $s_1 = \lceil 1.15 p_{max} \rceil$ and $s_2 = \lfloor 0.75 p_{min} \rfloor$, where p_{min} and p_{max} represent the minimum and maximum values of processing times established above and $\text{rand}(a, b)$ returns a uniformly distributed number on the interval $[a, b]$. We then take four cases for each of

these combinations, totalizing a sample of 144 settings. Finally, we generate ten instances for each setting, yielding a total of 1440 instances.

In a second step and with the intent of exploring the limits of our ALNS metaheuristic, we increased the problem size by also proposing a set of 1440 large instances with 50, 100 and 200 jobs, indicated as IPMTC-II. The number of machines is either 3, 4 or 5 for $j = 50$, 4, 5, 6 or 7 for $j = 100$, and 6, 7, 8, 9 or 10 for $j = 200$. The number of required tools is now given by 30 or 40 and its proportion with the magazine capacity C is guaranteed by the scheme described in Crama et al. (1994). The method for generating both processing times and switching times remains the same. We then obtain a total of 1440 problems by generating ten instances for each two cases of such machine-job-tool combinations.

4.2 Tuning instances and parameters

To tune the ALNS parameters, we have chosen a subset of 30 representative instances from the data set IPMTC-I. In order to set all parameters, each one took several values determined in advance, while the others were kept fixed. The ALNS metaheuristic was run five times for each parameter configuration and the setting related to the best average results was chosen. The final values of all parameters are displayed in Table 1.

Table 1: Values for the ALNS parameters.

Parameter	Meaning	Value
it	Iterations	10,000, 25,000 or 100,000
α	Consecutive iterations in a segment	1% it
γ	Percentage of jobs removed in each ALNS iteration	10% or 25%
δ	Avoids determinism in destroy operators	1.8
c	Cooling factor (SA)	0.995
r	Reaction factor	0.01
$(\alpha_1, \alpha_2, \alpha_3)$	Weight adjustment mechanism	$(50, 30, 20)$

4.3 Experimental results

The algorithm was implemented in C++ and experiments were conducted on a 2.67 GHz Intel Xeon X55 machine with 12 Gb of memory running under Linux. We have used IBM CPLEX 12.6, with default settings, to solve the MIP model (1)–(8). We imposed a time limit of 4200 seconds for the solution of each instance.

To obtain a comparison basis, we implemented the two best composite local improvement procedures of Fathi and Barnette (2002), namely CLIP1 and CLIP2. Further, we analyzed the application of the heuristic developed by Chaves et al. (2012) only combined with the model (1)–(8). This method is named Ch-B and is rather useful in assessing the quality of the search performed by the proposed destroy and repair algorithms.

Moreover, since the number of jobs removed from the current solution has a significant impact on the ALNS accuracy, we propose six scenarios to experiment our method, obtained by varying the number of iterations among 10, 25 and 100 thousands and the number of removals q equal to 10% or 25% of the total number of jobs:

- scenario 1 (Sc1): 10,000 iterations and 10% of removals;
- scenario 2 (Sc2): 10,000 iterations and 25% of removals;
- scenario 3 (Sc3): 25,000 iterations and 10% of removals;
- scenario 4 (Sc4): 25,000 iterations and 25% of removals;
- scenario 5 (Sc5): 100,000 iterations and 10% of removals;
- scenario 6 (Sc6): 100,000 iterations and 25% of removals.

4.3.1 Results on the set IPMTC-I

To ensure fair comparisons, all previously mentioned methods were run ten times for each instance of the set IPMTC-I, under the same machine configuration. Tables 2-6 summarize our computational results. Tables 2 and 3 present the gaps relative to the best known solution value obtained by any of the four methods CLIP1, CLIP2, ALNS-B and ALNS-R for a given instance, considering our worst and best scenarios, Sc1 and Sc4, respectively. The three first columns are the number of machines ($|M|$), jobs ($|J|$) and required tools ($|T|$). The average percentage deviation is reported in columns Gap and the percentage deviation with respect to the best solution found is displayed in columns Gap*. Our algorithm is able to produce better results for all classes of instances. The best results are obtained when the percentage of removals is 25% and the method has more freedom of action. Using different switching times does not seem to affect solution quality.

Table 2: Gap relative to the best solution found considering methods CLIP1, CLIP2, ALNS-B and ALNS-R – Scenario 1.

$\|M\|$	$\|J\|$	$\|T\|$	CLIP1		CLIP2		ALNS-B		ALNS-R	
			Gap	Gap*	Gap	Gap*	Gap	Gap*	Gap	Gap*
2	8	15	12.23	2.20	10.55	1.90	2.94	0.65	4.51	0.41
2	8	20	13.50	2.08	12.10	1.77	2.66	0.47	6.45	0.13
2	15	15	18.75	8.94	16.77	7.74	0.41	0.01	0.43	0.02
2	15	20	16.64	7.21	14.29	5.45	0.34	0.01	0.38	0.00
2	25	15	17.83	8.97	16.28	6.48	0.16	0.01	0.19	0.02
2	25	20	19.51	10.46	17.76	8.60	0.26	0.02	0.28	0.03
3	15	15	18.62	6.56	14.04	4.30	0.69	0.05	0.70	0.04
3	15	20	14.57	5.61	11.77	3.60	0.71	0.04	0.68	0.02
3	25	15	20.65	9.69	19.61	7.35	1.71	0.08	1.83	0.21
3	25	20	20.97	10.58	18.36	7.05	1.31	0.11	1.43	0.09
4	25	15	15.32	4.57	13.58	3.10	0.60	0.06	0.65	0.10
4	25	20	25.44	11.20	23.18	8.80	2.55	0.18	2.81	0.29
Average			**17.84**	**7.34**	**15.69**	**5.51**	**1.19**	**0.14**	**1.70**	**0.11**

Table 3: Gap relative to the best solution found considering methods CLIP1, CLIP2, ALNS-B and ALNS-R – Scenario 4.

$\|M\|$	$\|J\|$	$\|T\|$	CLIP1		CLIP2		ALNS-B		ALNS-R	
			Gap	Gap*	Gap	Gap*	Gap	Gap*	Gap	Gap*
2	8	15	12.25	2.22	10.58	1.92	0.00	0.00	0.00	0.00
2	8	20	13.52	2.10	12.12	1.78	0.00	0.00	0.00	0.00
2	15	15	18.76	8.95	16.78	7.74	0.02	0.00	0.02	0.00
2	15	20	16.64	7.22	14.29	5.45	0.05	0.00	0.04	0.00
2	25	15	17.86	9.00	16.31	6.51	0.07	0.01	0.08	0.01
2	25	20	19.52	10.48	17.77	8.61	0.11	0.02	0.11	0.02
3	15	15	18.63	6.57	14.05	4.31	0.10	0.00	0.09	0.00
3	15	20	14.57	5.61	11.78	3.60	0.06	0.00	0.06	0.00
3	25	15	20.75	9.78	19.71	7.44	0.40	0.04	0.41	0.04
3	25	20	21.12	10.72	18.51	7.19	0.30	0.02	0.32	0.05
4	25	15	15.47	4.71	13.72	3.24	0.34	0.05	0.34	0.06
4	25	20	25.76	11.48	23.50	9.08	0.70	0.06	0.66	0.05
Average			**17.90**	**7.40**	**15.76**	**5.57**	**0.18**	**0.02**	**0.18**	**0.02**

In terms of runtime, the ALNS is slower than CLIP1 and CLIP2, as expected since its components are more time-consuming than a local improvement procedure. Despite this, the largest computing time required by the ALNS-B is very reasonable considering the six scenarios, at less than ten minutes, on average.

As previously stated, the stopping condition of the ALNS is the number of iterations performed and the temperature in the simulated annealing. Tests showed that 25,000 iterations usually provide a good trade-off between computing time and solution quality. As indicated in Table 1, the number of consecutive iterations

in a segment of the metaheuristic is proportional to the total number of iterations performed. This is also true of the number of times that scores, weights and probabilities are adjusted to select destroy and repair operators. Thus, in general, a large number of iterations does not necessarily lead to the best results.

Our method also proved to be robust with respect to different initial solutions. In fact, the similar gaps obtained by the variants ALNS-B and ALNS-R show the ability of our algorithm to produce good results regardless of the starting point. Denote by t(s) and by t*(s) the total runtime and the time needed to find the best solution, respectively. Table 4 shows that the heuristic producing the initial solution Ch-B has the worst performance among all algorithms considered in the experiments. These results highlight the need for a post-optimization phase and confirm the good performance of the destroy and repair methods.

Table 4: Comparative performance of all methods and scenarios analyzed.

Method	**Gap(%)**	**Gap*(%)**	**t(s)**	**t*(s)**	**Imp**
ALNS-B - Sc1	9.82	8.55	33	11	5.58
ALNS-B - Sc2	8.68	8.40	84	28	5.72
ALNS-B - Sc3	9.39	8.49	69	22	5.90
ALNS-B - Sc4	8.53	**8.36**	196	58	5.87
ALNS-B - Sc5	9.10	8.44	167	45	6.07
ALNS-B - Sc6	**8.51**	**8.36**	534	123	5.88
ALNS-R - Sc1	10.48	8.52	115	50	11.39
ALNS-R - Sc2	8.69	8.40	346	135	9.85
ALNS-R - Sc3	9.95	8.43	285	102	11.71
ALNS-R - Sc4	8.53	8.37	867	283	10.08
ALNS-R - Sc5	9.70	8.40	738	220	11.93
ALNS-R - Sc6	**8.51**	**8.36**	1198	332	10.07
Ch-B	30.28	23.72	35	-	-
CLIP1	27.86	16.22	0	0	4.06
CLIP2	25.47	14.30	0	0	4.47

Note: Boldface numbers indicate the minimum % of deviation among the methods.

While it is clear that ALNS-B and ALNS-R are comparable in terms of solution quality, it is interesting to note that ALNS-B spends less than 31% of the runtime required by ALNS-R. Finally, the number of improvements applied to achieve the best solution using ALNS-R is slightly larger, as indicated in column Imp, which is justified by the poorer quality of its starting points. Figures 1 and 2 confirm the ALNS behavior. They display how the New, Current and Best solutions change over 25,000 iterations. Statistics on the left correspond to the ALNS-B variant, while the ALNS-R results are summarized on the right side.

Tables 5 and 6 show the performance of the ALNS destroy and repair operators, respectively. The second and third sets of columns provide the average percentage of calls during which a new best solution was

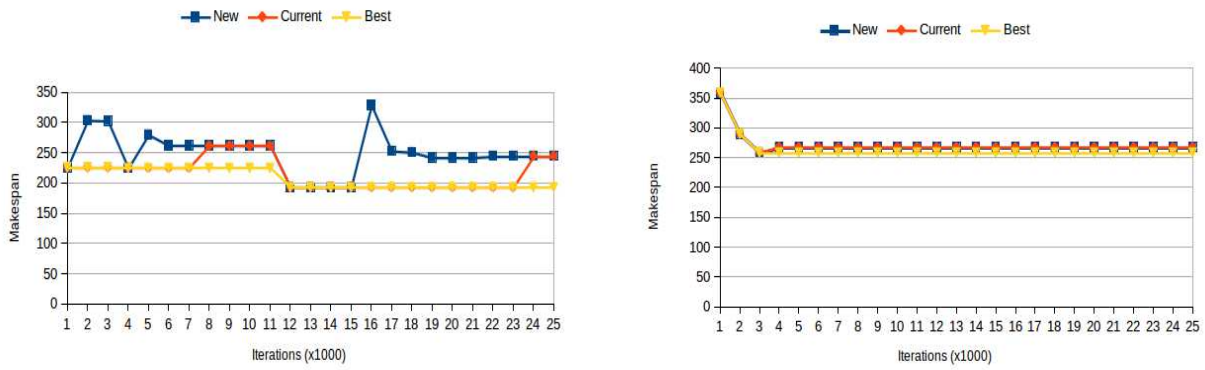

Figure 1: ALNS-B and ALNS-R behaviors for a 4-machines and 25-jobs problem (instance 1262 from IPMTC-I).

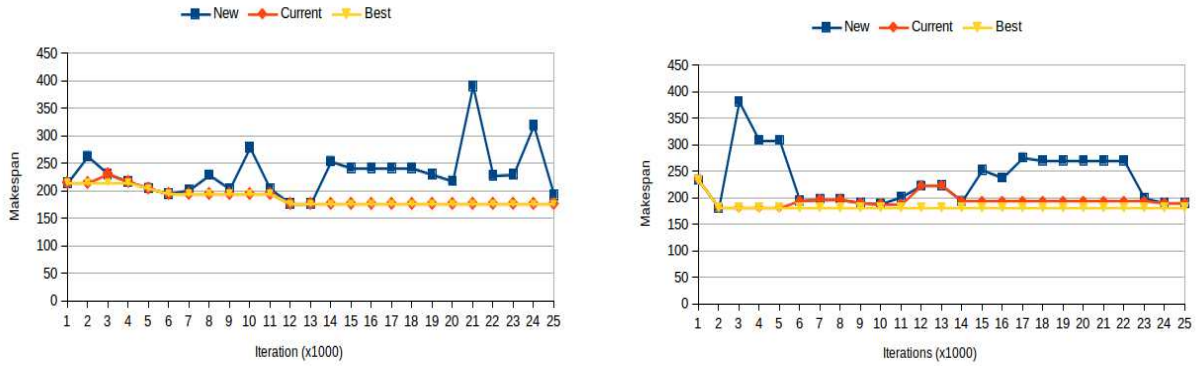

Figure 2: ALNS-B and ALNS-R behaviors for a 4-machines and 25-jobs problem (instance 1301 from IPMTC-I).

found by using each operator type. Among the destroy methods, Critical machine removal (D7) led to an improvement the most often, immediately followed by Largest deviation removal (D4). Both operators are based on simple greedy and intensifying movements, but are equally important to diversify the search. From the repair operators, 2-Regret insertion (R5) is the most successful. It is interesting to note the superiority of the operator Greedy$_\Delta$ insertion (R2) in comparison with Greedy$_p$ insertion (R1). Although both consist of greedy strategies, the second one performs a deeper analysis considering not only processing times, but also the contribution of the number of tool switches to the objective function.

Table 5: Statistics for the destroy operators.

Operator	Average % of improvements (By scenario)						Average % of improvements	Solution degradation
	Sc1	Sc2	Sc3	Sc4	Sc5	Sc6		
D1	7.78	4.43	6.95	2.69	4.63	2.29	4.80	0.10
D2	8.60	4.37	6.78	2.64	4.73	1.55	4.78	0.10
D3	8.90	4.30	6.95	2.65	6.14	1.84	5.13	0.10
D4	**14.45**	**8.31**	10.06	**4.64**	7.33	2.32	7.85	0.11
D5	12.60	7.30	7.57	4.12	5.02	2.14	6.46	0.10
D6	12.50	7.12	8.22	4.37	4.78	2.60	6.60	0.11
D7	13.85	6.74	**11.32**	3.92	**8.74**	**2.62**	**7.87**	0.11
D8	9.59	4.26	8.23	2.71	5.85	1.55	5.37	0.10
D9	9.01	4.41	6.98	2.69	4.96	1.54	4.93	0.10

Note: Boldface numbers indicate the maximum average % of calls and the maximum degradation among the operators.

Table 6: Statistics for the repair operators.

Operator	Average % of improvements (By scenario)						Average % of improvements	Solution degradation
	Sc1	Sc2	Sc3	Sc4	Sc5	Sc6		
R1	6.32	3.36	6.45	1.74	4.86	0.95	3.95	0.11
R2	15.16	5.05	12.67	3.16	10.68	1.87	8.10	0.10
R3	6.53	2.59	5.66	1.51	3.97	0.78	3.51	0.10
R4	14.85	4.46	12.72	2.88	10.44	1.89	7.87	0.11
R5	**16.72**	**5.12**	**13.63**	**3.24**	**11.13**	**2.65**	**8.75**	0.11

Note: Boldface numbers indicate the maximum average % of calls and the maximum degradation among the operators.

Finally, the fourth column presents the average degradation of the solutions found with and without the application of each operator. Through the same subset of 30 instances used during the tuning phase, ALNS-B was run excluding one operator at a time while keeping the others fixed. The solution degradation is nearly constant, ranging from 10% to 11%.

4.3.2 Results on the set IPMTC-II

We tested the limits of our method by running experiments on the large-size instances with the algorithms ALNS-B, ALNS-R, CLIP1 and CLIP2. With respect to the ALNS-B, initial tests indicated tractability issues even when solving instances with 50 jobs. To understand this behavior, we have experimented it on a subset of 15 representative instances from the set IPMTC-II, with the number of jobs varying among 50, 100 and 200. The ALNS-B was run twice for each problem, yielding a total of 30 experiments. For instances with 100 and 200 jobs, the method reached about 3600 seconds before finishing the search for the initial solution. For instances with 50 jobs, the ALNS-B was completely executed, but 83.76% of the runtime, on average, was spent with the initial solution method, as displayed in column Run(%) of Table 7. The fourth and fifth columns, t(s) and t$_i$(s), provide the total runtime and the time required to determine the initial solution, respectively.

Table 7: Performance of the ALNS-B – Scenario 4.

$\|M\|$	$\|J\|$	$\|T\|$	**t**(s)	**t**$_i$(s)	**Run**(%)
3	50	30	1000	861	86.10
3	50	30	957	820	85.68
3	50	30	1159	975	84.12
3	50	30	1135	943	83.08
4	50	30	1294	1092	84.39
4	50	30	1276	1067	83.62
5	50	40	924	719	77.81
5	50	40	907	701	77.29
5	50	40	3202	2813	87.85
5	50	40	3161	2772	87.69
Average			**1501.50**	**1276.30**	**83.76**

In order to analyze the impact of problem size on our ALNS, Table 8 presents the gap relative to the best known solution value obtained by any of the three methods ALNS-R, CLIP1 and CLIP2 for a given instance, considering the scenario Sc4, which was the best scenario identified during the experiments on the data set IPMTC-I. The column headings are the same as explained in Section 4.3.1, except the additional column Exc(%), which reports the percentage of instances of each machine-job-tool combination reaching the limit of 4 hours (14400 seconds) before completing the total number of iterations. As can be seen, the algorithm ALNS-R generates gaps equal to 1.7% and 2.2%, on average, for instances with 50 and 100 jobs, respectively, in contrast with the values of 19.6% and 11.44% obtained by CLIP2. For the very large instances with 200 jobs, the gap related to the ALNS-R is 2.32%, against 14.27% for CLIP2. It is important to note that even though the ALNS-R has been stopped by the time limit in 86.52% of the problems, our algorithm still outperforms CLIP1 and CLIP2 for all classes of instances.

5 Conclusion

We have modeled and solved the identical parallel machines problem with tooling constraints (IPMTC), which incorporates tooling decisions in flexible manufacturing systems. We have proposed an adaptive large neighborhood search (ALNS) metaheuristic composed of existing operators and of new operators designed specifically for the IPMTC and its two underlying problems. The metaheuristic was analyzed with two initial solution methods and under six different scenarios in order to assess the efficiency of the operators. To fully evaluate the effectiveness and accuracy of our ALNS, two new sets of instances were generated, containing up to 200 jobs, 10 machines and 40 tools. Computational experiments show that our method outperforms existing algorithms for all instances of the proposed benchmarks.

Table 8: Gap relative to the best solution found considering methods CLIP1, CLIP2 and ALNS-R – Scenario 4.

$\|M\|$	$\|J\|$	$\|T\|$	CLIP1		CLIP2		ALNS-R		Exc(%)
			Gap	Gap*	Gap	Gap*	Gap	Gap*	
3	50	30	23.79	15.16	22.56	14.73	1.62	0.00	9.33
3	50	40	18.89	11.86	18.41	10.94	1.29	0.00	74.17
4	50	30	20.84	11.75	18.57	11.23	1.67	0.00	31.88
4	50	40	20.27	11.54	20.46	11.20	1.89	0.00	82.33
4	100	30	18.31	7.69	15.28	6.37	1.96	0.00	100.00
4	100	40	13.77	4.45	10.93	3.49	1.97	0.12	100.00
5	50	30	18.66	9.99	15.90	7.93	1.62	0.00	15.83
5	50	40	21.69	12.28	21.69	12.71	2.08	0.00	63.00
5	100	30	15.12	5.14	11.88	4.24	1.87	0.02	100.00
5	100	40	12.40	3.38	9.78	2.33	2.37	0.48	100.00
6	100	30	16.51	5.95	12.14	2.98	2.41	0.46	100.00
6	100	40	13.04	2.89	9.63	1.41	3.12	0.89	100.00
6	200	30	22.25	12.03	17.06	9.51	1.84	0.00	100.00
6	200	40	18.79	9.37	14.99	7.81	2.54	0.03	100.00
7	100	30	14.63	5.13	10.82	3.35	2.06	0.37	100.00
7	100	40	15.08	5.09	11.04	3.22	1.85	0.06	100.00
7	200	30	21.38	12.87	15.90	8.60	2.29	0.04	100.00
7	200	40	18.82	9.39	13.88	7.23	2.64	0.05	100.00
8	200	30	21.52	10.98	16.29	9.40	2.17	0.00	100.00
8	200	40	17.57	7.65	12.38	5.04	2.38	0.08	100.00
9	200	30	18.81	9.51	13.96	7.35	2.17	0.01	100.00
9	200	40	16.77	7.54	12.52	6.24	2.73	0.07	100.00
10	200	30	19.72	10.43	14.27	7.84	2.08	0.05	100.00
10	200	40	17.24	8.20	11.51	4.87	2.38	0.05	100.00
Average			**18.16**	**8.76**	**14.66**	**7.08**	**2.13**	**0.12**	**86.52**

References

Al-Fawzan, M.A., Al-Sultan, K. ., 2002. A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. Computers and Industrial Engineering 44, 35–47.

Amaya, J.E., Cotta, C., Fernández, A.J., 2008. A memetic algorithm for the tool switching problem. Lecture Notes in Computer Science 5296, 190–202.

Amaya, J.E., Cotta, C., Fernández-Leiva, A.J., 2012. Solving the tool switching problem with memetic algorithms. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 26, 221–235.

Atan, T.S., Pandit, R., 1996. Auxiliary tool allocation in flexible manufacturing systems. European Journal of Operational Research 89, 642–659.

Ayres, R.U., 1988. Future trends in factory automation. Manufacturing review 1, 93–103.

Berrada, M., Stecke, K.E., 1986. A branch and bound approach for machine load balancing in flexible manufacturing systems. Management Science 32, 1316–1335.

Borenstein, D., Becker, J., 1994. Simflex: um avaliador de sistemas flexíveis de manufatura. Revista de Administração 29, 77–84.

Buyurgana, N., Saygin, C., Kilic, S.E., 2004. Tool allocation in flexible manufacturing systems with tool alternatives. Robotics and Computer-Integrated Manufacturing 20, 341–349.

Catanzaro, D., Gouveia, L.E.N., Labb, M., 2015. Improved integer linear programming formulations for the job sequencing and tool switching problem. European Journal of Operational Research 244, 766–777.

Chan, F.T.S., Swarnkar, R., 2006. Ant colony optimization approach to a fuzzy goal programming model for a machine tool selection and operation allocation problem in an fms. Robotics and Computer-Integrated Manufacturing 22, 353–362.

Chaves, A.A., Senne, E.L.F., Yanasse, H.H., 2012. Uma nova heurística para o problema de minimização de trocas de ferramentas. Gestão e Produção 19, 17–30.

Coffman, E.G., Denning, J.P., 1973. Operating systems theory. Englewood Cliffs: Prentice Hall.

Coffman, E.G., Garey, M.R., Johnson, D.S., 1978. An application of bin-packing to multiprocessor scheduling. SIAM Journal on Computing 7, 1–17.

Crama, Y., 1997. Combinatorial optimization models for production scheduling in automated manufacturing systems. European Journal of Operational Research 99, 136–153.

Crama, Y., Kolen, A.W.J., Oelermans, 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., Oerlemans, A., 1996. Production planning in automated manufacturing. Springer Verlag.

Cumings, S., 1986. Developing integrated tooling systems: a case study at garret turbine engine company. In: Proceedings of the Fall Industrial Engineering Conference (Institute of Industrial Engineers, Norcross, GA). Boston, USA, pp. 21–26.

Das, K., Baki, M., Li, X., 2009. Optimization of operation and changeover time for production planning and scheduling in a flexible manufacturing system. Computers and Industrial Engineering 56, 283–293.

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

Garey, M., Johnson, D.S., 1979. Computers and Intractability. Freeman, New York.

Ghiani, G., Grieco, A., Guerriero, E., 2010. Solving the job sequencing and tool switching problem as a nonlinear least cost hamiltonian cycle problem. Networks 55, 379–385.

Gómez, A.T., Lorena, L.A.N., 1998. Modelagem de sistemas de manufatura flexíveis considerando restrições temporais e a capacidade do magazine. Gestão e Produção 5, 69–80.

Gray, A., Seidmann, A., Stecke, K., 1988. A synthesis of tool-management issues and decision problems in automated manufacturing. Tech. Rep. 590-b, School of Business Administration, University of Michigan.

Gray, A.E., Seidmann, A., Stecke, K.E., 1993. A synthesis of decision models for tool management in automated manufacturing. Management Science 39, 549–567.

Hertz, A., Widmer, M., 1996. An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. Discrete Applied Mathematics 65, 319–345.

Hwang, H.-C., Lee, K., Chang, S.Y., 2005. The effect of machine availability on the worst-case performance of lpt. Discrete Applied Mathematics 148, 49–61.

Kabir, L., 2011. Minimizing makespan of a flexible machine under tooling constraints. Master's thesis, Ryerson University.

Keung, K.W., Ip, W.H., Lee, T.C., 2001. A genetic algorithm approach to the multiple machine tool selection problem. Journal of Intelligent Manufacturing 12, 331–342.

Koulamas, C.P., 1991. Total tool requirements in multi-level machining systems. International Journal of Production Research 29, 417–437.

Koulamas, C.P., Kyparíris, G.J., 2009. A modified lpt algorithm for the two uniform parallel machine makespan minimization problem. European Journal of Operational Research 96, 61–68.

Kumar, N.S., Sridharan, R., 2009. Simulation modeling and analysis of part and tool flow control decisions in a flexible manufacturing system. Robotics and Computer-Integrated Manufacturing 25, 829–838.

Laporte, G., Salazar, J.J., Semet, F., 2004. Exact alghrithms for the job sequencing and tool switching problem. IIE Transactions 36, 37–45.

Lee, C.-Y., 1991. Parallel machines scheduling with non-simultaneous machine available time. Discrete Applied Mathematics 30, 53–61.

Lee, C.-Y., He, Y., Tang, G., 2000. A note on "parallel machine scheduling with non-simultaneous machine available time". Discrete Applied Mathematics 100, 133–135.

Lee, C.-Y., Massey, J.D., 1988. Multiprocessor scheduling: Combining lpt and multifit. Discrete Applied Mathematics 20, 233–242.

Melnyk, S., Ghosh, S., Ragatz, G., 1989. Tooling constrains and shop floor scheduling: a simulation study. Journal of Operations Management 8, 69–89.

Mohamed, Z.M., Bernardo, J.J., 1997. Tool planning models for flexible manufacturing systems. European Journal of Operational Research 103, 497–514.

Novas, J.M., Henning, G.P., 2014. Integrated scheduling of resource-constrained flexible manufacturing systems using constraint programming. Expert Systems with Applications 41, 2286–2299.

Panwalkar, S.S., Islander, W., 1977. A survey of scheduling rules. Operations Research 25, 45–61.

Parker, R.G., 1995. Deterministic Scheduling Theory.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers and Operations Research 34, 2403–2435.

Privault, C., Finke, G., 1995. Modelling a tool switching problem on a single nc-machine. Journal of Intelligent Manufacturing 6, 87–94.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation science 40, 455–472.

Salonen, K., Raduly-Baka, C., Nevalainen, O.S., 2006. A note on the tool switching probelm of a flexible machine. Computers and Industrial Engineering 50, 458–465.

Shaw, P., 1997. A new local search algorithm providing high quality solutions to vehicle routing problems. Tech. rep., Department of Computer Science, University of Strathclyde, United Kingdom.

Shirazi, R., Frizelle, G.D.M., 2001. Minimizing the number of tool switches on a flexible machine: an empirical study. International Journal of Production Research 39, 3547–3560.

Stecke, K., 1983. Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems. Management Science 29, 273–288.

Stecke, K., 1985. Design, planning, scheduling and control problems of flexible manufacturing systems. Annals of Operations Research 3, 3–12.

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.

Tomek, P., 1986. Tooling strategies related to fms management. The FMS Magazine 5, 102–107.

Veeramani, D., Upton, D., Barash, M., 1992. Cutting-tool management in computer-integrated manufacturing. International Journal of Flexible Manufacturing Systems 3/4, 237–265.

Yanasse, H.H., Rodrigues, R.C.M., Senne, E.L.F., 2009. Um algoritmo enumerativo baseado em ordenamento parcial para resolução do problema de minimização de trocas de ferramentas. Gestão e Produção 16, 370–381.

Zeballos, L.J., 2010. A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems. Robotics and Computer-Integrated Manufacturing 26, 725–743.

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