Efficient Scheduling of a Stochastic No-Wait Job Shop with Controllable Processing Times

Efficient Scheduling of a Stochastic No-Wait Job Shop with Controllable Processing Times

Alexander Aschauer[a,*], Florian Roetzer[a], Andreas Steinboeck[a], Andreas Kugi[a,b]

[a] *TU Wien, Automation and Control Institute ACIN, Vienna, Austria*
[b] *Center for Vision, Automation & Control, Austrian Institute of Technology, Vienna, Austria*

Abstract

This work derives a novel effective and efficient algorithm for a stochastic no-wait job-shop scheduling problem with controllable processing times. Some of the processing times are stochastic and the proposed solution effectively minimizes the makespan and increases the robustness of the makespan against deviating processing times. Therefore, a no-wait job shop with controllable deterministic processing times is solved by a decomposition into timetabling and sequencing. During timetabling, extra safety margins are added to the scheduled processing times without delaying jobs. In the sequence optimization subproblem, an extra penalty term is added to the cost function which punishes uncertain tasks at positions that have an impact on the makespan. Simulation results based on real plant data and tailor-made benchmark problems show that these measures can reduce the standard deviation of the makespan dramatically. This significantly improves the prediction accuracy of the scheduling method.

Keywords: no-wait job shop, controllable processing times, stochastic scheduling, recursive timetabling, tabu search, dynamic tabu list

1. Introduction

This paper originates from a scheduling problem in the metals industry. A hot rolling mill with several batch type reheating furnaces is considered. For every slab (product), a production plan specifies the necessary production steps (heating/reheating to a certain temperature, manipulating, rolling, straightening, cutting, ...). The nominal processing times are available. Heating times can be extended up to given upper bounds without loss of product quality. The

*Corresponding author
Email addresses: aschauer@acin.tuwien.ac.at (Alexander Aschauer), roetzer@acin.tuwien.ac.at (Florian Roetzer), steinboeck@acin.tuwien.ac.at (Andreas Steinboeck), kugi@acin.tuwien.ac.at (Andreas Kugi)

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

nominal heating times serve as lower bounds. In the considered process, after the initial heating step the subsequent production steps must be performed immediately to avoid unwanted cooling. A major source of uncertainty in the processing times is the semiautomatic manipulation of the slabs on the roller table. For the uncertain processing times, in particular for the manipulation of the products, statistical information is available. The goals of the scheduling are to determine a production schedule and an optimal product sequence which maximize the throughput (minimize the makespan) and which are robust against uncertainties of the processing times.

The described problem can be classified as a stochastic no-wait job-shop scheduling problem with controllable processing times. Sometimes controllable processing times means that some processing times can be reduced by allocating extra resources to the corresponding machines. In the current paper, however, heating times can be controlled within given bounds without any additional costs. Even though, the main problem of variable processing times remains the same.

1.1. Literature Review

Scheduling in hot rolling mills is a challenging and up-to-date problem. Özgür et al. (2020) provide a thorough survey on scheduling methods for hot rolling mills, which gives an overview of the optimization methods applied, the constraints incorporated, and the levels of abstraction. Nevertheless, research questions remain open and the problems identified in this paper have not been addressed in this combination.

The job-shop scheduling problem (JSP) is one of the classical scheduling problems and well studied in the literature. It is NP-hard (Lenstra et al., 1977) and powerful heuristics have been presented for its solution, e. g., the shifting bottleneck heuristic by Adams et al. (1988). The no-wait JSP is also NP-hard (Lenstra & Rinnooy Kan, 1979) so that heuristic solution methods are needed to solve larger problems. Extended surveys about no-wait scheduling problems were published by Hall & Sriskandarajah (1996) and Allahverdi (2016). In many works, the no-wait JSP is handled by a division into a timetabling and a sequencing subproblem (Macchiaroli et al., 1999; Schuster & Framinan, 2003; Bożejko & Makuchowski, 2009; Aschauer et al., 2017).

According to the survey of Shabtay & Steiner (2007), only very few works consider controllable processing times in JSP. Grabowski & Janiak (1987) proved that the JSP with controllable processing times is NP-hard and they proposed a branch-and-bound algorithm. Mokhtari et al. (2011) investigated a no-wait JSP with controllable processing times. They entitled this problem a no-wait job-shop crashing problem and solved it by dividing it into a crashing, a timetabling, and a sequencing subproblem. In this context, crashing means to determine the durations of the controllable processing times. In (Mokhtari et al., 2011), only lower or upper bound values are allocated to the controllable processing times. Aschauer et al. (2017) solved the no-wait JSP with controllable processing times by a decomposition into a timetabling and a sequencing subproblem. They handled the controllable processing times by non-delay recursive timetabling.

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Pinedo (2016) addressed stochastic scheduling in detail and stated that stochastic JSP with more than two machines did not receive much attention in the literature. However, Jamili (2016) published an article about robust job-shop scheduling. He calculated required buffer times to reach certain robustness levels. Horng et al. (2012) presented an evolutionary algorithm for the stochastic JSP. They used a rough stochastic scheduling model with short simulation length to calculate the fitness value of a population. Framinan & Perez-Gonzalez (2015) published a study on stochastic flow-shop scheduling problems. They found out that, for most settings, the solutions derived for the corresponding deterministic problems work extremely well also for the stochastic setting.

To the authors' knowledge, systematic consideration of stochastic processing times in no-wait JSP with controllable processing times has not been addressed in the literature so far. The question if the controllable processing times can be used to increase the robustness of the schedule should be answered in this paper.

1.2. Contribution and Content

This work proposes a scheduling solution for a stochastic no-wait JSP with controllable processing times. This problem has not been addressed in the literature in this combination. The stochastic processing times are handled by deterministic scheduling based on their nominal values and subsequent evaluation of the schedule in terms of the robustness of the makespan against deviating processing times. The deterministic scheduling is done by a decomposition into a timetabling and a sequencing subproblem. The recursive timetabling algorithm from (Aschauer et al., 2017) represents an approach to handle the highly interconnected problem from the perspective of individual tasks, which allows an easier implementation. This algorithm is extended to add more tolerance (i. e. safety margins) to the scheduled processing times without delaying jobs. In order to evaluate the robustness of the schedule, an algorithm to calculate the effective safety margins of the tasks with respect to increases of the makespan is developed. The sequencing is done by a tabu search heuristic. Samarghandi et al. (2013) showed that tabu search works well with most timetabling algorithms. Moreover, tabu search is perfectly suitable for parallel computation. The information about the safety margins and the makespan is used in the tabu search optimization heuristic to find a robust product sequence with minimal makespan. The method is tested in simulations with production data from a real plant and based on tailor-made benchmark problems.

In Section 2, the developed scheduling algorithm is presented. A problem formulation is given and the extended recursive timetabling algorithm is described in detail. Based on the timetabling result, an algorithm to calculate the safety margins is developed. The sequence optimization is done by a tabu search heuristic which needs the timetabling result and the safety margins to calculate the cost function. Section 3 contains numerical results. First, the developed scheduling algorithm is applied to existing benchmarks of no-wait JSP to show its performance compared to the state of the art. Then, scheduling results for production data from the hot rolling mill are presented. Uncertain

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

processing times are approximated by a probability distribution and results for stochastic processing times are presented. Furthermore, tailored benchmarks of stochastic no-wait JSP with controllable processing times are created by a modification of existing benchmarks. Based on these modified benchmark scenarios, the effectiveness of the stochastic scheduling approach is demonstrated. Finally, conclusions are drawn in Section 4.

2. Scheduling

This section presents an efficient scheduling strategy for the stochastic no-wait JSP with controllable processing times. The no-wait constraints connect all production steps (tasks) of a single product (job) insofar as each task has to start immediately after its predecessor. In no-wait environments with fixed processing times, the starting time of the job, which is the starting time of its first task, determines the temporal arrangement of all tasks of this job. With controllable processing times, however, the starting times of the individual tasks need to be determined. The strategy of solving a no-wait JSP with fixed processing times by dividing it into a timetabling and a sequencing subproblem decreases complexity. This strategy can also be applied here, however, the complexity of the timetabling increases. Despite the stochastic nature of the processing times, deterministic timetabling is applied based on nominal values and allowed upper bounds of the processing times. To account for the stochastic nature of the processing times, the deterministic timetabling result is evaluated in terms of safety margins with respect to deviations of the processing times. This information about safety margins and the standard deviations of the processing times are incorporated into an extra penalty term of the sequence optimization problem. All tasks that feature processing times with high standard deviations and that are scheduled in time slots which directly affect the makespan are penalized. In the literature, the sequence of these time slots is sometimes referred to as critical path. The sequence optimization is done by a tabu search heuristic (Aschauer et al., 2017, 2018). Optimization goals are to minimize the total production time (makespan) and its uncertainty in the form of stochastic variations. In the following subsections, a problem formulation, algorithms for the timetabling and the calculation of the safety margins, and the proposed sequence optimization are presented. The results of these algorithms, i. e., a sequence of jobs, a sequence of tasks or rather their respective starting times are integrated in a so-called schedule.

2.1. Problem Formulation

The problem consists of a production lot of N_J jobs described by the set $\{J_j | j = 1, \ldots, N_J\}$. Each job J_j consists of $N_{T,j}$ tasks summarized in the set $\{T_{j,n} | n = 1, \ldots, N_{T,j}\}$. The plant features in total N_M machines described by the set $\{M_m | m = 1, \ldots, N_M\}$. Each machine M_m can process only one product at a time, and each task $T_{j,n}$ is associated with a machine based on the machine number $m_{j,n} \in \{1, \ldots, N_M\}$. The processing time $d_{j,n}$ of a task $T_{j,n}$

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

is characterized by a lower bound $\underline{d}_{j,n}$, an upper bound $\overline{d}_{j,n}$, and a standard deviation $\sigma_{j,n}$[1]. Preemption of tasks is not allowed and tasks of the same job are connected by no-wait constraints. Scheduling goals are the minimization of the makespan C_m and its uncertainty characterized by its standard deviation σ_m. These goals are incorporated into the following optimization problem.

$$\min_{t_{j,n}} \quad \mathbf{w}^T \begin{bmatrix} C_m \\ \sigma_m \end{bmatrix} \tag{1a}$$

s.t.

$$t_{j,n} \geq 0 \qquad j = 1, \dots, N_J, n = 1, \dots, N_{T,j} \tag{1b}$$

$$t_{j,n+1} - t_{j,n} \in [\underline{d}_{j,n}, \overline{d}_{j,n}] \qquad j = 1, \dots, N_J, n = 1, \dots, N_{T,j} - 1 \tag{1c}$$

$$t^{end}_{j,n} \leq t_{k,l} \quad \vee \quad t^{end}_{k,l} \leq t_{j,n} \qquad \{(j,n,k,l) | m_{j,n} = m_{k,l} \wedge j \neq k\} \tag{1d}$$

Equation (1a) describes the optimization goal, which is to minimize the weighted sum of the makespan C_m and its standard deviation σ_m. The user-defined weighting vector $\mathbf{w}^T = [w_1, w_2]$ has non-negative entries and the ratio w_1/w_2 prioritizes the optimization goals. The optimization variables are the starting times $t_{j,n}$ of the tasks. The constraint (1b) guarantees non-negative starting times $t_{j,n}$. The constraint (1c) forces the task durations to lie within their lower and upper bounds. The constraint (1d) ensures that each machine does not handle more than one task at a time. The end times of the tasks follow as

$$t^{end}_{j,n} = \begin{cases} t_{j,n+1} & n = 1, \dots, N_{T,j} - 1 \\ t_{j,n} + \underline{d}_{j,n} & n = N_{T,j}. \end{cases} \tag{2}$$

The end time of a job is $t^{end}_{j,N_{T,j}}$. The makespan can be calculated as the maximum end time of all jobs, i. e.,

$$C_m = \max_j \quad t^{end}_{j,N_{T,j}}. \tag{3}$$

2.2. Timetabling

The purpose of the timetabling is to find feasible starting times for all tasks contained in a schedule. Typically, the sequence of jobs is given and the makespan should be minimized. Schuster (2006) showed that optimal timetabling is NP-hard even for the no-wait JSP with fixed processing times. A computationally cheaper alternative is non-delay timetabling. Non-delay timetabling generates a schedule job-wise in the order of the given job sequence. The term non-delay means that the job to be scheduled finishes as early as possible (without delay) subject to the previously scheduled jobs and the given constraints.

[1] In the scheduling problem of the hot rolling mill considered in this work, only the heating times have specified lower and upper bounds. The lower and upper bounds of the remaining processing times are equally set to the nominal values of the respective processing times, i. e., $\underline{d}_{j,n} = \overline{d}_{j,n} = d_{j,n}$.

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

In other words, the end time of the job to be scheduled is minimized, given that the schedule of the jobs which occur earlier in the sequence cannot be changed anymore. Practical examples show that non-delay timetabling yields excellent results (Schuster & Framinan, 2003).

2.2.1. Find Starting Times for One Job

The core algorithm of non-delay timetabling needs to find starting times for all tasks of a job. Here, a modified version of the algorithm presented by Aschauer et al. (2017) is used. The modification ensures that safety margins are added to the processing times. These safety margins are added without delaying the end time of the job and should help to compensate for elongated processing times. The algorithm works recursively and every recursion addresses one task $T_{j,n}$. The recursive algorithm is selected because the no-wait constraints link all tasks of a job and the information of previous and subsequent tasks is needed to make a scheduling decision. The recursive algorithm allows an implementation from the perspective of individual tasks, which is more easily programmable than an implementation by loops. The core algorithm is explained in all details while the underlying functions are only explained textually.

Pseudo code of the recursive core algorithm, which is the function called FINDSCHEDULE, is presented in Algorithm 1. The inputs of the function are the job number j, the task number n, an earliest possible starting time t_{est}, and a latest possible starting time t_{lst}. The values t_{est} and t_{lst} define the time slot when the preceding task $T_{j,n-1}$ could be finished, i. e., $t^{end}_{j,n-1} \in [t_{est}, t_{lst}]$. The outputs of the function are the scheduled starting time $t_{j,n}$ of the task $T_{j,n}$, an error flag err, and a potential starting time t_{pst}. The starting time $t_{j,n}$ is set in case of no error, i. e., $err = 0$, and the potential starting time t_{pst} is set in case of an error, i. e., $err = 1$.

The function FINDFIRSTSLOTAFTER finds the first free time slot $[t_1, t_2]$ at the machine $M_{m_{j,n}}$ that starts after t_{est}, i. e., $t_1 \geq t_{est}$, and which is sufficiently long, i. e., $t_2 - t_1 \geq \underline{d}_{j,n}$. For every machine M_m, the scheduled tasks of the previous jobs are stored. Along the timeline, there occur slots when the machines are utilized and slots when the machines are free. Figure 1 illustrates an example of the machine utilization. Here, t_1 is the beginning of the first free time slot after t_{est} and t_2 is the end time of this slot. In case t_{est} lies after the beginning of this slot and the slot is still long enough ($t_2 - t_{est} \geq \underline{d}_{j,n}$), t_1 is equal to t_{est}.

Due to the no-wait constraint, the time slot $[t_1, t_2]$ is only feasible if $t_1 \leq t_{lst}$, otherwise an error occurs and the error $err = 1$ and the potential starting time $t_{pst} = t_1$ are returned. This is the only possibility of generating a new error. The error occurs because the starting time t_1 of the free time slot starts after the latest possible starting time t_{lst}.

If $[t_1, t_2]$ is a feasible time slot, the recursion is either terminated at the last task of the job ($n = N_{T,j}$) or continued by a recursive function call of FINDSCHEDULE for the next task $T_{j,n+1}$. The current task $T_{j,n}$ with the feasible time slot $[t_1, t_2]$ can finish the earliest at $t_1 + \underline{d}_{j,n}$ and the latest at the minimum

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Algorithm 1: The recursive algorithm FINDSCHEDULE.

Input: job number j, task number n, earliest possible starting time t_{est}, latest possible starting time t_{lst}
Output: starting time of the task $t_{j,n}$, error err, potential starting time t_{pst}

```
% Finding the first free time slot beginning after t_est
[t_1, t_2] = FINDFIRSTSLOTAFTER(t_est, d_{j,n}, m_{j,n});
while 1 do
    % Checking the time slot [t_1, t_2] for feasibility
    if t_1 ≤ t_lst then
        % Checking the termination criterion of the recursion
        if n = N_{T,j} then
            t_{j,n} = t_1; err = 0; return;
        else
            % Recursive function call
            [t_{j,n+1}, err_{+1}, t_{pst+1}] =
              FINDSCHEDULE(j, n + 1, t_1 + d_{j,n}, MIN(t_2, t_lst + d̄_{j,n}));
            % Checking the recursive function call for an error
            if err_{+1} = 0 then
                t_{j,n} = SEL2NDLST(t_1, t_{j,n+1} − d̄_{j,n}, t_lst, t_{j,n+1} − d_{j,n} − tol);
                err = 0; return;
            else
                % Checking if the error is because of t_2
                if t_2 < t_{pst+1} & t_lst + d̄_{j,n} ≥ t_{pst+1} then
                    % Finding the first free time slot lasting until t_{pst+1}
                    [t_1, t_2] = FINDFIRSTSLOTUNTIL(t_{pst+1}, d_{j,n}, m_{j,n});
                else
                    % Returning the error err_{+1} to the previous task
                    err = 1; t_pst = t_{pst+1} − d̄_{j,n}; return;
                end
            end
        end
    else
        err = 1; t_pst = t_1; return;
    end
end
```

of t_2 and $t_{lst} + \overline{d}_{j,n}$. These times are forwarded to the next task as the earliest possible starting time and the latest possible starting time, respectively. If the function call for the next task returns without error ($err_{+1} = 0$), a scheduling decision concerning the current task $T_{j,n}$ can be made. This decision is made by the function SEL2NDLST, which was not contained in this form in the algorithm presented in (Aschauer et al., 2017).

The function SEL2NDLST selects the starting time $t_{j,n}$ of task $T_{j,n}$ as the

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

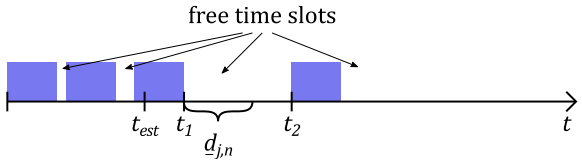

Figure 1: An example for the machine utilization and the selection of the first free time slot $[t_1, t_2]$ after t_{est} which is sufficiently long, i. e., $t_2 - t_1 \geq \underline{d}_{j,n}$.

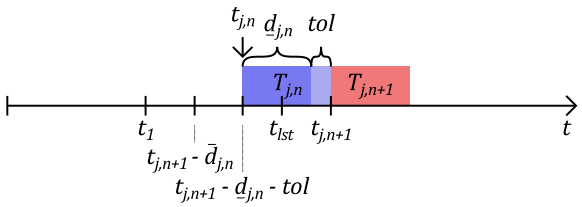

Figure 2: A typical example for the scheduling of task $T_{j,n}$. The function SEL2NDLST selects the starting time $t_{j,n}$ as the second latest out of the four potential times t_1, $t_{j,n+1} - \bar{d}_{j,n}$, $t_{j,n+1} - \underline{d}_{j,n} - tol$, and t_{lst}. In this example, the selected time is $t_{j,n} = t_{j,n+1} - \underline{d}_{j,n} - tol$.

second latest out of the four potential times t_1, $t_{j,n+1} - \bar{d}_{j,n}$, $t_{j,n+1} - \underline{d}_{j,n} - tol$, and t_{lst}. The rationale behind this selection strategy is described in detail in the following. The tuning parameter *tol* represents a safety margin insofar as a task should start *tol* time units earlier than necessary. This does not cause a delay of the end time of the job. A minimal end time of the job is already guaranteed by the selection of the first free time slots after the earliest possible starting times of the tasks during the buildup of the recursion. Whenever $t_{j,n+1} > t_1 + \underline{d}_{j,n}$, a possible safety margin can be added to the task which should preferably be selected as *tol*. A valid schedule must fulfill $t_{j,n} \leq t_{lst}$. Moreover, $t_{j,n} \geq t_1$ and $t_{j,n} \geq t_{j,n+1} - \bar{d}_{j,n}$ are necessary conditions. Due to the previous inequality conditions also $t_1 \leq t_{lst}$ and $t_{j,n+1} - \bar{d}_{j,n} \leq t_{lst}$ hold. By complete enumeration of all permissible permutations of the four potential starting times, it can be proved that the second latest time is the right choice, which always fulfills the stated conditions and meets the goal best. The enumeration of the permissible permutations is shown in the Appendix. Figure 2 shows a typical example for the scheduling decision made by the function SEL2NDLST.

When the call of FINDSCHEDULE for the next task $T_{j,n+1}$ returns an error ($err_{+1} = 1$), it also returns a suggestion t_{pst+1} for a new (later) starting time of the next task. If $t_2 < t_{pst+1}$ and $t_{lst} + \bar{d}_{j,n} \geq t_{pst+1}$ there is a chance to eliminate the error err_{+1} here. In fact, a new (later) free time slot $[t_1, t_2]$ is sought and the while-loop is repeated. The function FINDFIRSTSLOTUNTIL finds the first free time slot at the respective machine $M_{m_{j,n}}$ with $t_2 - t_1 \geq \underline{d}_{j,n}$ that take until t_{pst+1}, i. e., $t_2 \geq t_{pst+1}$. If there is no chance to eliminate the error err_{+1}

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

here, the flag $err = 1$ is returned to the previous function call together with the potential starting time $t_{pst} = t_{pst+1} - \overline{d}_{j,n}$.

The computational complexity of Algorithm 1 can significantly vary and depends on the number of previously scheduled jobs and on the input data. Without loss of generality, let us assume for this analysis that every job J_j has the same number of tasks $N_{T,j} = N_T$. In a best-case estimation, every free time slot immediately leads to a valid schedule and the complexity of Algorithm 1 is $\mathcal{O}(N_T)$. In a worst-case estimation, every free time slot, except for the last one, leads to an error and the errors are returned over many recursions. An upper bound for the number of erroneous free time slots is the total number of tasks of all jobs $N_J N_T$. An upper bound for the number of recursions of one erroneous free time slot is the number of tasks N_T of the current job. Therefore, a worst-case estimation of the complexity of Algorithm 1 is $\mathcal{O}(N_J N_T^2)$. However, this upper bound is very unlikely to be reached. The number of possible erroneous free time slots is much lower for the first jobs to be scheduled and a transfer of an error over many recursions rarely happens. In simulations, the computational complexity of Algorithm 1 was always close to the lower bound $\mathcal{O}(N_T)$. Note that these considerations do not account for the complexity of the underlying functions.

2.2.2. Timetabling of a Sequence

The job sequence is stored in a so-called permutation vector

$$\mathbf{p} = [p_1, \dots, p_i, \dots, p_{N_J}]^T. \tag{4}$$

Each entry p_i represents a job number, i. e., $p_i \in \{1, \dots, N_J\}$ with $p_i \neq p_k$ for $i \neq k$.

Timetabling according to Section 2.2.1 is executed sequentially in the order $j = p_1, \dots, p_{N_J}$. The function FINDSCHEDULE is called for the first task $T_{j,1}$ of each job J_j with default values $t_{est} = 0$ and $t_{lst} = \infty$ as earliest and latest possible starting time, respectively, i. e.,

$$[t_{j,1}, err] = \text{FINDSCHEDULE}(j, 1, 0, \infty). \tag{5}$$

The choice $t_{lst} = \infty$ and the fact that all machines are empty at least in the distant future always guarantee a successful termination ($err = 0$) of the recursion. The starting times $t_{j,n}$, $n = 2, \dots, N_{T,j}$ of the remaining tasks of the job J_j are of course also stored during the recursion.

After the termination of FINDSCHEDULE for a job J_j, the required time slots of the tasks $[t_{j,n}, t_{j,n}^{end}]$, with $t_{j,n}^{end}$ according to (2), are reserved at the respective machines $M_{m_{j,n}}$. The timetabling continues with a call of FINDSCHEDULE for the next job according to the sequence $\mathbf{p}$.

Because of the arrangement $t_{est} = 0$ as the earliest possible starting time of the first tasks $T_{j,1}$, the actual production sequence of the jobs can differ from the required sequence $\mathbf{p}$. Strict adherence to the sequence is ensured by setting the earliest possible starting time t_{est} of the task $T_{j,1}$, $j = p_i$ to the already scheduled starting time $t_{p_{i-1},1}$ of the previous job. However, Mokhtari et al. (2011) found

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

that such an extra constraint decreases the performance of the timetabling. Since the sequence of the jobs should be optimized, the permutation vector $\mathbf{p}$ can be seen as a planning sequence and the actual production sequence could be derived from the obtained timetabling result if needed.

2.3. Sequencing

The sequencing searches for the optimal permutation vector $\mathbf{p}^*$ which minimizes a cost function. Schuster (2006) showed that the sequencing problem for the no-wait JSP is NP-hard, which motivates the use of heuristic solution methods also here. The study of Samarghandi et al. (2013) demonstrates that the tabu search heuristic performs well with many timetabling algorithms. Therefore, it is also applied here. The scheduling goals are the minimization of the makespan and its disturbances, cf. (1a). Disturbances of the makespan increase its standard deviation σ_m. This occurs especially if tasks which have only little safety margins are delayed.

2.3.1. Calculation of the Effective Safety Margins

Timetabling yields starting times for all tasks of all jobs. Based on these starting times, effective safety margins for all tasks of all jobs can be calculated. The effective safety margin is the extra time a task can take longer than its minimum duration $\underline{d}_{j,n}$ until it elongates the makespan C_m.

Algorithm 2 shows pseudo code to calculate all effective safety margins $[tol_{j,n}]$, $j = 1, ..., N_J$, $n = 1, ..., N_{T,j}$. The inputs of the algorithm are an array with all starting times $[t_{j,n}]$ and the makespan C_m. The effective safety margins are initialized, i. e., $tol_{j,n} = -1$, to recognize unassigned values. A function sortDescending sorts the starting times $[t_{j,n}]$ in descending order and stores them in the list *task_list*. In a subsequent for-loop, effective safety margins are calculated according to the order of *task_list*. First, the indexes j and n of the task to be investigated are determined.

Figure 3 illustrates the calculation of the effective safety margin $tol_{j,n}$ of the task $T_{j,n}$. Tasks in the same row belong to the same job and tasks of the same color are executed at the same machine. The next task of the same job is $T_{j,n+1}$ and the next task at the same machine is $T_{k,l}$.

In Algorithm 2, first, the effective safety margin depending on the next task of the same job and, second, the effective safety margin depending on the next task at the same machine is calculated. The final effective safety margin $tol_{j,n}$ of the task $T_{j,n}$ is the minimum of both values.

The effective safety margin depending on the next task of the same job is calculated by the effective safety margin $tol_{j,n+1}$ of the next task $T_{j,n+1}$ plus the task's own safety margin $t_{j,n+1} - (t_{j,n} + \underline{d}_{j,n})$. Due to $t_{j,n+1} > t_{j,n}$, $t_{j,n+1}$ occurs earlier in *task_list* and the effective safety margin $tol_{j,n+1}$ of task $T_{j,n+1}$ has already been calculated. If $T_{j,n}$ is the last task of a job ($n = N_{T,j}$), the safety margin of this task is $C_m - (t_{j,n} + \underline{d}_{j,n})$.

For the effective safety margin depending on the next task at the same machine, first, the next task $T_{k,l}$ starting after task $T_{j,n}$ at machine $M_{m_{j,n}} =$

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Algorithm 2: The algorithm for the calculation of the effective safety margins.

Input: starting times of all tasks $[t_{j,n}]$, makespan C_m
Output: effective safety margins of all tasks $[tol_{j,n}]$
% Initializing the effective safety margins
$[tol_{j,n}] = [-1]$;
% Sorting the tasks in descending order of their starting times
$task\_list$ = sortDescending($[t_{j,n}]$);
for $i = 1$: length($task\_list$) **do**
  $[j, n]$ = getIndex($task\_list[i]$);
  % Effective safety margins due to the next task of the same job
  if $n < N_{T,j}$ **then**
    $tol_{j,n} = tol_{j,n+1} + t_{j,n+1} - (t_{j,n} + \underline{d}_{j,n})$;
  else
    $tol_{j,n} = C_m - (t_{j,n} + \underline{d}_{j,n})$;
  end
  % Effective safety margin due to the next task at the same machine
  $[k, l, end]$ = getNext($t_{j,n}, m_{j,n}$);
  while !end **do**
    % Minimum of previous value and new effective safety margin
    $tol_{j,n} = \min(tol_{j,n}, tol_{k,l} + t_{k,l} - (t_{j,n} + \underline{d}_{j,n}))$;
    if $l > 1$ **then**
      if $tol_{k,l-1} \geq 0$ **then**
        % Minimum of previous value and new effective safety margin
        $tol_{j,n} = \min(tol_{j,n}, tol_{k,l-1} + (t_{k,l-1} + \underline{d}_{k,l-1}) - (t_{j,n} + \underline{d}_{j,n}))$;
        break;
      else
        $[k, l, end]$ = getNext($t_{k,l-1}, m_{k,l-1}$);
      end
    else
      break;
    end
  end
end

$M_{m_{k,l}}$ has to be identified. This is done by the function getNext. If $T_{j,n}$ is the last task at the respective machine $M_{m_{j,n}}$, getNext sets the flag $end = 1$. The while-loop is entered if end is not set. The effective safety margin depending on the next task at the same machine is calculated as the sum of the effective safety margin $tol_{k,l}$ of the next task $T_{k,l}$ and the safety margin $t_{k,l} - (t_{j,n} + \underline{d}_{j,n})$ between the starting time $t_{k,l}$ of task $T_{k,l}$ and the minimal end time $t_{j,n} + \underline{d}_{j,n}$ of task $T_{j,n}$. The minimum of this value and the previously stored value $tol_{j,n}$ is taken. Due to $t_{k,l} > t_{j,n}$, the effective safety margin $tol_{k,l}$ has already been

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

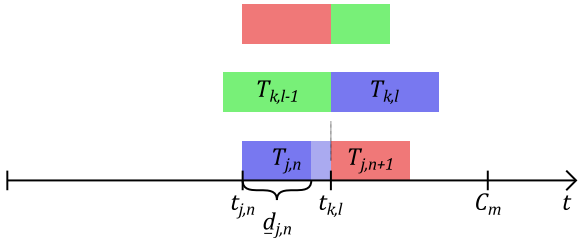

Figure 3: An example for calculating the effective safety margin of the task $T_{j,n}$ depending on the next task of the same job and on the next task at the same machine. Tasks in the same row belong to the same job and tasks with the same color are executed at the same machine.

calculated.

Because of the no-wait constraint, a delayed start of task $T_{k,l}$ would also delay the previous task $T_{k,l-1}$ of job J_k. Therefore, the first if-clause in the while-loop checks if task $T_{k,l}$ has a previous task ($l > 1$) and the second if-clause checks if the effective safety margin $tol_{k,l-1}$ of the previous task $T_{k,l-1}$ has already been calculated ($tol_{k,l-1} \geq 0$). If $tol_{k,l-1}$ has been calculated, it can be used in the calculation of the effective safety margin $(tol_{k,l-1}+(t_{k,l-1}+\underline{d}_{k,l-1})-(t_{j,n}+\underline{d}_{j,n}))$ of the task $T_{j,n}$. The minimum of this value and the previously stored value $tol_{j,n}$ is taken and the while-loop terminates. Else, the next task starting after task $T_{k,l-1}$ at the machine $M_{m_{k,l-1}}$ is identified by the function GETNEXT and the while-loop continues.

The computational complexity of Algorithm 2 is estimated by lower and upper bounds. The while-loop in Algorithm 2 is only rerun if the effective safety margin $tol_{k,l-1}$ has not been assigned before. This can only happen if $t_{k,l-1} \leq t_{j,n}$, which is at most possible once per machine. Thus, the maximum possible number of iterations of the while-loop is the number of machines N_M. Without loss of generality, let us again assume for this analysis that every job J_j has the same number of tasks $N_{T,j} = N_T$ and therefore, the total number of tasks is $N_J N_T$. This leads to a best-case complexity of Algorithm 2 of $\mathcal{O}(N_J N_T)$ if the while-loop is never rerun and to a worst-case complexity of $\mathcal{O}(N_M N_J N_T)$. Simulations showed that the while-loop is hardly ever rerun and practically, the computational complexity of Algorithm 2 is always close to its lower bound $\mathcal{O}(N_J N_T)$. Note that these considerations do not account for the complexity of the underlying functions.

2.3.2. Tabu Search

Tabu search is a heuristic local search algorithm which iteratively tries to improve a cost function. It was invented by Glover & Laguna (1997). Its basic idea is to generate a neighborhood $\mathcal{N}(\mathbf{p})$ of the current permutation vector $\mathbf{p}$ and continue the search at the best performing neighbor. To avoid unwanted loops and oscillations (revisiting of previous solutions) during the search, a so called tabu list is maintained, which prohibits certain neighbors for a limited

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

number of iterations. Tabu search is perfectly suitable for parallel computation. The calculations of the cost functions of the single neighbors are independent of each other and can thus be easily parallelized. In the current work, the cost functions of the neighbors are evaluated in a parallel for-loop based on the openMP-package in the programming language C.

For the problem considered in this paper, the cost function $J(\mathbf{p})$ is designed as the weighted sum of the makespan and potential disturbances of the makespan, i. e.,

$$J(\mathbf{p}) = C_m + w_{stoch} \sum_{j=1}^{N_J} \sum_{n=1}^{N_{T,j}} \max\{0, 3\sigma_{j,n} - tol_{j,n}\}. \tag{6}$$

Compared to the cost function (1a), disturbances of the makespan are not considered by its standard deviation σ_m but by the second term in (6). The calculation of the standard deviation σ_m would be computationally too expensive. The makespan C_m is calculated based on (3) and the effective safety margins $tol_{j,n}$ of all tasks $T_{j,n}, j = 1, \dots, N_J, n = 1, \dots, N_{T,j}$ are derived by Algorithm 2. Both, equation (3) and Algorithm 2, need the timetabling result of the sequence $\mathbf{p}$ as input. A penalty value for a potential disturbance is added if the effective safety margin $tol_{j,n}$ of a task is less than three times the standard deviation $\sigma_{j,n}$ of the task. The standard deviations $\sigma_{j,n}$ of all tasks $T_{j,n}, j = 1, \dots, N_J, n = 1, \dots, N_{T,j}$ are given in advance. The weighting factor of the potential disturbances is $w_{stoch} \geq 0$.

The sequence $\mathbf{p}_0$ used to initialize the tabu search is built by a construction heuristic described in (Aschauer et al., 2017). This construction heuristic builds up the sequence job by job. It evaluates all remaining unscheduled jobs and adds the best fitting job to the sequence.

Beginning from $\mathbf{p}_0$, the iteration rule of the tabu search is

$$\mathbf{p}_{k+1} = \underset{\mathbf{p} \in \mathcal{N}(\mathbf{p}_k)}{\arg\min}\, J(\mathbf{p}). \tag{7}$$

Here, the neighborhood $\mathcal{N}(\mathbf{p}_k)$ of the permutation vector $\mathbf{p}_k$ is the set of all permutations that can be generated by an exchange or shift operation, except for the permutations which are 'tabu'. More details about exchange and shift operations can be found in (Aschauer et al., 2018). Three tabu mechanisms are implemented to force the algorithm to explore new regions of the solution space and to avoid loops and oscillations, i. e.,

(1) Permanent storage of the evaluated permutation vectors $\mathbf{p}_0, \dots, \mathbf{p}_k$.
(2) Temporal storage of previously evaluated neighbors $\mathcal{N}(\mathbf{p}_0), \dots, \mathcal{N}(\mathbf{p}_k)$ in a first-in-first-out (FIFO) list with fixed list size.
(3) Temporal storage of the positions of the exchange or shift operations that generated the best permutation vectors $\mathbf{p}_1, \dots, \mathbf{p}_k$ in a FIFO list with dynamic list size.

In this work, the best sequencing results were achieved by applying all three tabu mechanisms. Tabu list (1) prevents the algorithm from revisiting exactly the

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

same permutation vector. The tabu lists (2) and (3) enforce a broader search. The strategy to update the dynamic list size of the tabu list (3) is as follows. A counter is incremented if the cost function remains the same ($J(\mathbf{p}_{k+1}) = J(\mathbf{p}_k)$) or decremented if the cost function changes ($J(\mathbf{p}_{k+1}) \neq J(\mathbf{p}_k)$). If the counter exceeds a certain threshold, the list size is set to a very high value (e. g., $\lfloor 0.9 N_J \rfloor$ with the flooring operator $\lfloor \cdot \rfloor$). If the counter falls below this threshold again, the list size is set back to its normal value. The very large list size implies that after some iterations almost all positions of the exchange or shift operations are 'tabu' and the algorithm is forced to explore new regions. The tabu list for the positions is cleared if no neighbors can be found anymore, i. e., if $\mathcal{N}(\mathbf{p}_k) = \{\}$.

Figure 4 shows the evaluation of the cost function $J(\mathbf{p}_k)$ during the search with different numbers of tabu lists. The dotted line represents the cost function of the best permutation found and the final best value is written on the right. Figure 4 contains convergence plots for the data set 'la26' of the results in Section 3.1, which are representative for all investigated problems. The spikes in the convergence plots originate from the dynamic list size of the tabu list (3). A constant or periodically changing cost function $J(\mathbf{p}_k)$ indicates a loop and the algorithm does not provide further improvements. No single tabu list can prevent the algorithm from oscillating. The tabu lists (1) and (3) break these oscillations but a periodic pattern is observed at the beginning. The best result is achieved by applying all three tabu lists.

3. Results

Numerical results are presented, first, for standard no-wait JSP with fixed processing times. For this class of problems, benchmarks and optimal solutions are available in the literature. Solving these benchmark problems by the developed scheduling algorithm will demonstrate its outstanding performance. Second, processing times of the considered plant are analyzed in detail, a probability distributions of uncertain processing times is derived, and scheduling results with plant data are presented. Third, new benchmark problems similar to the considered plant environment are created and the stochastic scheduling results are compared against results from purely deterministic scheduling. All computations are performed on a PC with 3700Mhz Intel i7 quad-core processor, 16GB RAM, and Windows 10 operating system. The data handling is done in MATLAB while the scheduling algorithms are implemented in the programming language C and called from the MATLAB environment via *mex* functions.

3.1. Deterministic Scheduling Results of Benchmark Problems

To demonstrate the performance of the developed scheduling algorithm, benchmark problems of no-wait JSP with fixed processing times, which have already been investigated in the literature, are scheduled.

A benchmark problem of size $N_M \times N_J$ is specified by a data set containing the fixed processing times $d_{j,n}$ and the machine numbers $m_{j,n}$ of all tasks $T_{j,n}$, $j = 1, \ldots, N_J$, $n = 1, \ldots, N_M$. For every job J_j, the number of tasks $N_{T,j}$

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

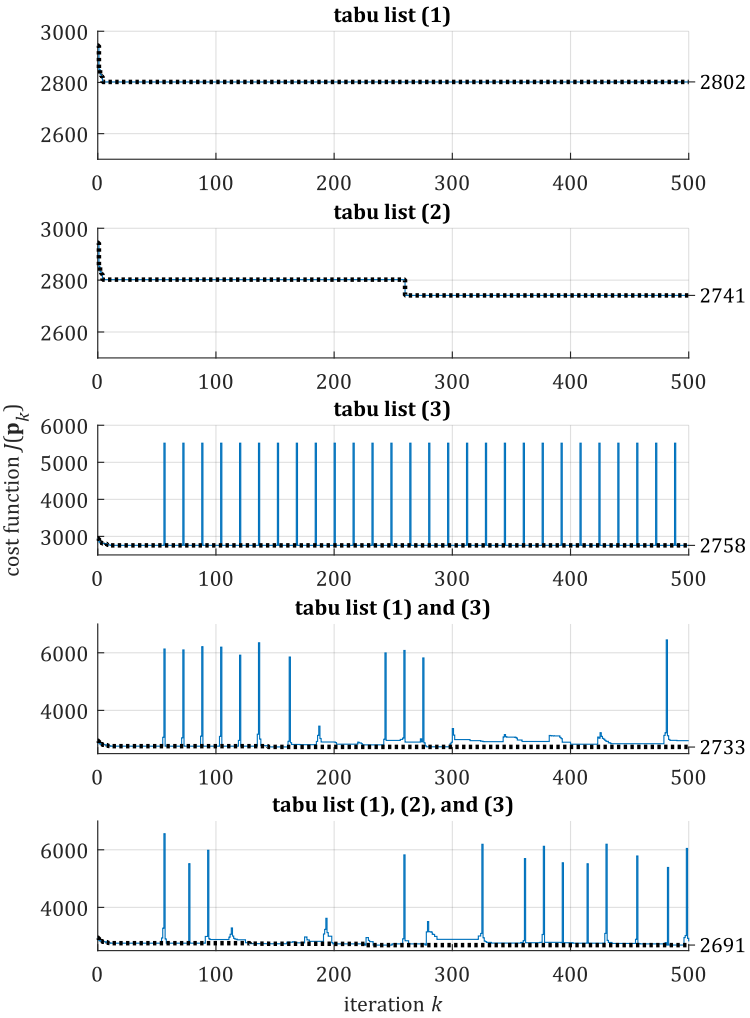

Figure 4: Convergence plots of data set 'la26' with different numbers of tabu lists.

equals the number of machines N_M, i. e., each job is processed once at every machine. The data sets are indicated by name abbreviations of the inventing

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Table 1: Scheduling results for small no-wait JSP benchmark problems.

data set	size	C_m^{opt}	C_m^{Schu}	$\Delta_{\%}^{Schu}$	C_m^{Mok}	$\Delta_{\%}^{Mok}$	C_m	$\Delta_{\%}$
la01	5×10	971	1043	7.42	975	0.41	975	0.41
la02	5×10	937	990	5.66	937	0.00	963	2.77
la03	5×10	820	832	1.46	820	0.00	820	0.00
la04	5×10	887	889	0.23	888	0.11	887	0.00
la05	5×10	777	817	5.15	784	0.90	781	0.51
ft10	10×10	1607	1620	0.81	0	0.00	1607	0.00
orb01	10×10	1615	1663	2.97	1615	0.00	1615	0.00
orb02	10×10	1485	1555	4.71	1518	2.22	1485	0.00
orb03	10×10	1599	1603	0.25	1617	1.13	1599	0.00
orb04	10×10	1653	1653	0.00	1653	0.00	1684	1.88
orb05	10×10	1365	1415	3.66	1385	1.47	1370	0.37
orb06	10×10	1555	1555	0.00	1557	0.13	1555	0.00
orb07	10×10	689	706	2.47	711	3.19	711	3.19
orb08	10×10	1319	1319	0.00	1319	0.00	1319	0.00
orb09	10×10	1445	1535	6.23	1445	0.00	1445	0.00
orb10	10×10	1557	1618	3.92	1581	1.54	1580	1.48
la16	10×10	1575	1637	3.94	1575	0.00	1575	0.00
la17	10×10	1371	1430	4.30	1371	0.00	1384	0.95
la18	10×10	1417	1555	9.74	1507	6.35	1417	0.00
la19	10×10	1482	1610	8.64	1491	0.61	1491	0.61
la20	10×10	1526	1705	11.73	1541	0.98	1526	0.00
average				**4.12**		**0.95**		**0.61**

authors and can be downloaded from the OR-library ([dataset] Beasley, 1990).

The presented algorithm can easily handle this type of benchmark problems by setting the lower and upper bounds of the processing times to the given fixed processing times, i. e., $\overline{d}_{j,n} = \underline{d}_{j,n} = d_{j,n}$. For deterministic scheduling, $\sigma_{j,n} = 0$, which implies $J(\mathbf{p}) = C_m$ in (6).

Table 1 shows scheduling results for several small no-wait JSP benchmark problems. For these small problems, optimal makespan values C_m^{opt}, which can for example be derived by the branch and bound algorithm by Mascis & Pacciarelli (2002), are available.

C_m^{Schu} are the makespan values achieved by Schuster (2006) with a tabu search algorithm after 500 iterations. C_m^{Mok} are the makespan values obtained by Mokhtari et al. (2011) with a genetic algorithm. The makespan C_m is achieved by the presented algorithm after 500 tabu search iterations. The stopping criterion of 500 tabu search iterations is used to allow for a fair comparison with existing results from the literature. Moreover, a fixed number of iterations has the benefit of almost identical computational times for identical problem sizes. In some cases, the makespan C_m could be further improved by more iterations. The results of all three solution methods are compared against the

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

optimal makespan C_m^{opt}. As a measure of suboptimality, the relative deviations in percent are calculated in the form

$$\Delta_{\%} = \frac{C_m - C_m^{opt}}{C_m^{opt}} 100. \tag{8}$$

This yields an average suboptimality of only 0.61 % for the presented algorithm. For comparison, Schuster (2006) reached an average suboptimality of 4.12 % and Mokhtari et al. (2011) reached an average suboptimality of 0.95 %. The computational times for all small no-wait JSP benchmark problems are well below 1 s and therefore not explicitly given.

For the larger no-wait JSP benchmark problems presented in Table 2, strictly optimal solutions are not available. Therefore, the makespan results C_m^{Schu} published by Schuster (2006) are used as benchmark values for these large problems. The values C_m^{Schu} were obtained by a tabu search algorithm after 500 iterations within the computational time T_c^{Schust}. Bożejko & Makuchowski (2009) improved these results on average by 1.54 % using a hybrid tabu search algorithm that requires the same computational time. The algorithm developed in the current paper improves the results of Schuster (2006) on average by 3.02 % and requires the remarkably smaller computational time T_c. This improvement has two main reasons: a) accelerated hardware[2], b) parallel evaluation of the cost functions of the neighborhood permutations during the tabu search on all available CPU cores.

These examples demonstrate the effectiveness and the efficiency of the proposed algorithm.

3.2. Plant Environment

For the scheduling problem of the industrial hot rolling mill considered in this paper (cf. Section 1), some of the processing times are subject to stochastic disturbances. The most uncertain processing step is the semiautomatic manipulation of the products on the roller table. The disturbances of the other processing times are very small. Therefore, only the processing times of the semiautomatic manipulation are considered as stochastic.

3.2.1. Probability Distribution

For the semiautomatic manipulation, a sufficiently large sample of 1845 recorded real processing times is available for statistical evaluation. Figure 5 shows the frequency distribution $h(d)$ of the durations d of semiautomatic manipulation and a fitted probability density function in the form of a β-distribution. The β-distribution is a good choice for an asymmetric probability density on a limited interval. The probability density function of the

[2]The computation is done on a PC with 3700Mhz Intel i7 quad-core processor, 16GB RAM, and Windows 10 operating system. Schuster (2006) used a PC with 1400Mhz Athlon processor running the operating system Windows 2000.

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Table 2: Scheduling results for large no-wait JSP benchmark problems.

data set	size	C_m^{Schu}	C_m^{Boz}	$\Delta_{\%}^{Boz}$	C_m	$\Delta_{\%}$	T_c^{Schu}[s]	T_c[s]	T_c^{Schu}/T_c
la26	10×20	2664	2738	2.78	2691	1.01	14	2.3	6.1
la27	10×20	2968	2794	-5.86	2881	-2.93	27	2.3	11.9
la28	10×20	2886	2741	-5.02	2759	-4.40	24	2.3	10.6
la29	10×20	2671	2596	-2.81	2589	-3.07	12	2.3	5.3
la30	10×20	2939	2791	-5.04	2712	-7.72	12	2.3	5.3
la31	10×30	3822	3869	1.23	3736	-2.25	151	11.5	13.1
la32	10×30	4186	4045	-3.37	4070	-2.77	176	12.2	14.4
la33	10×30	3869	3751	-3.05	3725	-3.72	120	12.6	9.5
la34	10×30	3957	3936	-0.53	3772	-4.68	102	12.8	8.0
la35	10×30	3908	3918	0.26	3938	0.77	120	12.1	9.9
la36	15×15	2993	2893	-3.34	2837	-5.21	9	1.8	5.1
la37	15×15	3171	3107	-2.02	3054	-3.69	7	1.8	4.0
la38	15×15	2734	2706	-1.02	2737	0.11	6	1.7	3.5
la39	15×15	2804	2725	-2.82	2860	2.00	9	1.7	5.3
la40	15×15	2977	2804	-5.81	2594	-12.87	12	1.7	7.2
abz7	15×20	1820	1775	-2.47	1713	-5.88	20	5.7	3.5
abz8	15×20	1815	1727	-4.85	1714	-5.56	51	5.5	9.4
abz9	15×20	1781	1705	-4.27	1710	-3.99	52	5.5	9.5
swv01	10×20	2396	2424	1.17	2330	-2.75	11	2.2	4.9
swv02	10×20	2492	2484	-0.32	2417	-3.01	16	2.0	8.0
swv03	10×20	2489	2404	-3.42	2435	-2.17	17	2.2	7.6
swv04	10×20	2520	2545	0.99	2580	2.38	23	2.1	10.8
swv05	10×20	2482	2489	0.28	2494	0.48	22	2.2	10.1
swv06	15×20	3502	3463	-1.11	3352	-4.28	29	5.8	5.0
swv07	15×20	3343	3299	-1.32	3309	-1.02	32	6.0	5.4
swv08	15×20	3611	3567	-1.22	3529	-2.27	29	5.9	4.9
swv09	15×20	3436	3439	0.09	3349	-2.53	39	5.8	6.7
swv10	15×20	3569	3561	-0.22	3488	-2.27	23	6.0	3.8
swv11	10×50	5586	5634	0.86	5502	-1.50	1736	133.7	13.0
swv12	10×50	5358	5465	2.00	5559	3.75	2212	134.7	16.4
swv13	10×50	5546	5807	4.71	5587	0.74	2360	142.2	16.6
swv14	10×50	5483	5458	-0.46	5362	-2.21	1602	131.3	12.2
swv15	10×50	5516	5619	1.87	5470	-0.83	2077	136.4	15.2
swv16	10×50	6337	6233	-1.64	5793	-8.58	1347	127.3	10.6
swv17	10×50	5884	5900	0.27	5760	-2.11	1760	119.8	14.7
swv18	10×50	6247	5931	-5.06	5754	-7.89	1430	128.8	11.1
swv19	10×50	6308	6283	-0.40	5836	-7.48	1481	135.4	10.9
swv20	10×50	6019	5945	-1.23	5829	-3.16	1843	123.2	15.0
yn1	20×20	2654	2630	-0.90	2598	-2.11	68	10.3	6.6
yn2	20×20	2705	2647	-2.14	2502	-7.50	41	11.0	3.7
yn3	20×20	2644	2465	-6.77	2575	-2.61	134	10.7	12.6
yn4	20×20	2705	2630	-2.77	2627	-2.88	53	10.4	5.1
average				**-1.54**		**-3.02**			**8.9**

β-distribution is

$$f_\beta(x) = \begin{cases} \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} & 0 < x < 1 \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

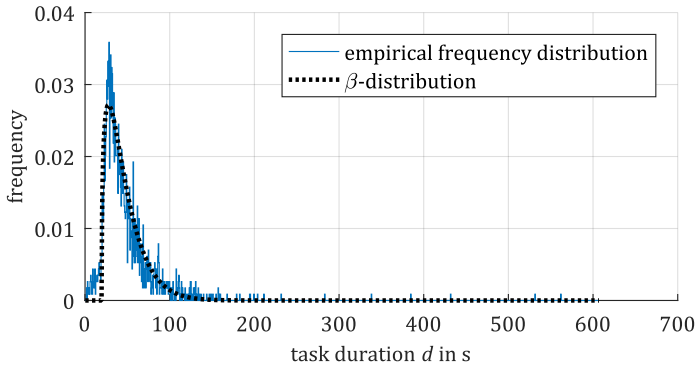

Figure 5: Frequency distribution of the duration of semiautomatic manipulation.

which is controlled by the parameters α and β (Papoulis & Pillai, 2002). $B(\alpha, \beta)$ is the beta-function, i. e.,

$$B(\alpha, \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\mathrm{d}x, \tag{10}$$

which is needed for normalization. A transformation of (9) to the interval $[d_s, d_s + d_r]$ of possible processing times yields the probability density function

$$f(d) = \frac{1}{d_r} f_\beta \left(\frac{d - d_s}{d_r} \right), \tag{11}$$

which is fitted to the empirical frequency distribution $h(d)$ shown in Figure 5. An optimization problem

$$\underset{\alpha,\beta,d_s,d_r}{\arg\min} \sum_{samples} |h(d) - f(d)|^2 \tag{12a}$$

$$\text{s.t.} \quad d_r \leq d_{max} - d_{min} \tag{12b}$$

is formulated and solved by the MATLAB optimizer *fmincon* to estimate the parameters α, β, d_s, and d_r. The optimizer *fmincon* is executed with the default optimization algorithm *interior-point*. Constraint (12b) limits the probability density function (11) to the range of the samples. Here, d_{max} and d_{min} are the samples with maximal and minimal duration, respectively.

The expected value of the β-distribution from (9) is $E(x) = \alpha/(\alpha+\beta)$ and its variance is $var(x) = \alpha\beta/((\alpha+\beta+1)(\alpha+\beta)^2)$. For the timetabling, the lower and upper bounds $\underline{d}_{j,n}$ and $\overline{d}_{j,n}$ of the processing times of the semiautomatic manipulation are chosen equal to the expected value of d according to (11), i. e.,

$$\underline{d}_{j,n} = \overline{d}_{j,n} = E(d) = d_s + d_r \frac{\alpha}{\alpha+\beta}. \tag{13}$$

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

The standard deviation $\sigma_{j,n}$ of semiautomatic manipulation tasks is the standard deviation of d according to (11), i. e.,

$$\sigma_{j,n} = \sqrt{var(d)} = \frac{d_r}{\alpha+\beta}\sqrt{\frac{\alpha\beta}{\alpha+\beta+1}}. \tag{14}$$

Random numbers $\tilde{d}_{j,n}$ to simulate the stochastic processing times of the semiautomatic manipulation tasks can therefore be calculated as

$$\begin{aligned}\tilde{d}_{j,n} &= d_s + d_r \mathrm{B}_{rnd}(\alpha,\beta) \\ &= \underline{d}_{j,n} + \sigma_{j,n}(\alpha+\beta)\sqrt{\frac{\alpha+\beta+1}{\alpha\beta}}\left(\mathrm{B}_{rnd}(\alpha,\beta) - \frac{\alpha}{\alpha+\beta}\right).\end{aligned} \tag{15}$$

$\mathrm{B}_{rnd}(\alpha,\beta)$ is a random number distributed in the interval $(0,1)$ according to the β-distribution from (9).

3.2.2. Scheduling Results

For a production lot containing 27 jobs, scheduling results of the original sequence, a deterministic optimization, and a stochastic optimization are compared. Each job has between 4 and 28 tasks. The original sequence is the production sequence of the jobs realized at the plant, which was manually chosen by an experienced operator. Unfortunately, the realized starting times of the tasks are not available. Therefore, the starting times of the tasks for the original sequence are calculated by Algorithm 1 with $tol = 0$. The scheduling result of the original sequence is marked with the superscript *orig*. Deterministic optimization means that the scheduling procedure of Section 2 is executed without measures against disturbances of the processing times ($tol = 0$ in Algorithm 1 and $w_{stoch} = 0$ in the cost function (6)). This scheduling result is marked with the superscript *det*. In comparison, the stochastic optimization also considers deviating processing times ($tol = 30\,\mathrm{s}$, various values of w_{stoch}). These results are further called stochastic and marked with the superscript *stoch*. The scheduling results are evaluated in a Monte Carlo simulation (1000 runs) with randomly generated processing times $\tilde{d}_{j,n}$ of semiautomatic manipulation tasks according to equation (15). Table 3 shows the mean values $\overline{C_m}$ of the obtained makespan and the standard deviations σ_m. The mean makespan $\overline{C_m}$ is improved by 4.77 % by the deterministic optimization compared to the original sequence and the standard deviation σ_m deteriorates by 7.6 %. The standard deviation σ_m of the makespan can be reduced by up to 57.6 % by the stochastic scheduling approach compared to the original sequence depending on the weighting factor w_{stoch}. The results from Table 3 show that a weighting factor $w_{stoch} = 2.0$ yields a good compromise between improved makespan and reduced uncertainty.

3.3. Stochastic Scheduling Results of Benchmark Problems

The stochastic no-wait JSP with controllable processing times has not been addressed in the literature so far. Therefore, benchmark problems are generated and deterministic and stochastic scheduling results are compared against each other.

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Table 3: Comparison of deterministic and stochastic scheduling results for a production lot of the considered hot rolling mill.

w_{stoch}	$\overline{C_m^{orig}}$	$\overline{C_m^{det}}$	$\overline{C_m^{stoch}}$	$\Delta_{\%}$	σ_m^{orig}	σ_m^{det}	σ_m^{stoch}	$\Delta_{\%}$
		46517.5		-4.77		65.0		7.6
0.5			46631.2	-4.54			48.7	-19.4
1.0	48848.2		46753.8	-4.29	60.4		44.2	-26.8
2.0			46635.8	-4.53			30.3	-49.8
4.0			46909.5	-3.97			25.6	-57.6

3.3.1. Generation of Benchmark Problems

Benchmark problems for the stochastic no-wait JSP with controllable processing times are generated based on the existing benchmark problems listed in Table 2. Each benchmark problem consists of N_J jobs and each job of N_M tasks. For each task $T_{j,n}$, $j = 1, \ldots, N_J$, $n = 1, \ldots, N_M$, a processing time $d_{j,n}$ and the respective machine number $m_{j,n}$ are given. Based on this given data and Table 4 the stochastic benchmark problems are generated.

The lower bounds $\underline{d}_{j,n}$ of the processing times are set to the given processing times of the benchmark problems, i. e.,

$$\underline{d}_{j,n} = d_{j,n}. \tag{16}$$

The upper bounds $\overline{d}_{j,n}$ are chosen to be approximately 10 % higher, i. e.,

$$\overline{d}_{j,n} = \lceil 1.1 d_{j,n} \rceil, \tag{17}$$

with the ceiling operator $\lceil \cdot \rceil$.

Moreover, for the generated benchmark problems, it is assumed that approximately half of the jobs are subject to stochastic disturbances. For these stochastic jobs J_j, $j = 1, \ldots, \lfloor \frac{N_J}{2} \rfloor$, it is assumed that approximately half of the tasks have stochastic processing times. Tasks $T_{j,n}$ of the stochastic jobs which are executed on machines with uneven machine numbers ($m_{j,n}$ uneven) are chosen to be stochastic.

To model the stochastic processing times $\tilde{d}_{j,n}$, the shifted and stretched β-distribution (11) with $\alpha = 2$ and $\beta = 5$ is used. To be able to calculate stochastic processing times $\tilde{d}_{j,n}$ according to (15), shifts $d_{s|j,n}$ and ranges $d_{r|j,n}$ are needed. Therefore, on the one hand, the expected values of the stochastic processing times are set to the given processing times of the benchmarks problems, i. e., $E(\tilde{d}_{j,n}) = d_{j,n}$. According to equation (13) the shifts $d_{s|j,n}$ of a stochastic processing time follows as

$$d_{s|j,n} = d_{j,n} - d_{r|j,n} \frac{2}{7}. \tag{18}$$

On the other hand, the ranges $d_{r|j,n}$ of the stochastic processing times are chosen to be machine dependent and defined by percentages of the given processing

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Table 4: Machine dependent percentages of the range of the stochastic processing times.

machine number m	1	3	5	7	9	11	13	15	17	19
percentage $p_{\%}(m)$	20	40	60	50	30	10	20	30	40	50

times of the benchmarks problems. As stated above, only tasks which are executed on machines with uneven machine numbers are subject to stochastic disturbances. The percentages $p_{\%}(m)$ for these machines are given in Table 4 and the range $d_{r|j,n}$ of the stochastic processing times can be calculated as

$$d_{r|j,n} = \frac{p_{\%}(m_{j,n})}{100} d_{j,n}. \tag{19}$$

Based on (14) and (19), the standard deviations $\sigma_{j,n}$ of the stochastic tasks follow in the form

$$\sigma_{j,n} = \frac{p_{\%}(m_{j,n})}{100} d_{j,n} \frac{\sqrt{5}}{14}. \tag{20}$$

For the other tasks, the standard deviations are $\sigma_{j,n} = 0$. According to (15), random numbers to simulate the stochastic processing times can be computed in the form

$$\tilde{d}_{j,n} = d_{j,n} + \sigma_{j,n} \frac{14}{\sqrt{5}} \left(B_{rnd}(2,5) - \frac{2}{7} \right). \tag{21}$$

3.3.2. Scheduling Results

The benchmark problems generated in Section 3.3.1 are scheduled both deterministically with $tol = 0$ and $w_{stoch} = 0$ as well as stochastically with $tol = 10$ and $w_{stoch} = 1$. These scheduling results are evaluated in a Monte Carlo simulation (1000 runs) with randomly generated processing times according to (21). The mean values $\overline{C_m}$ of the obtained makespans and their standard deviations σ_m are given in Table 5 for deterministic and stochastic scheduling. The mean makespan is almost identical on average but the standard deviation is reduced by 41.45 % on average by the stochastic scheduling approach.

4. Conclusions

In this paper, an effective and efficient algorithm is presented to handle stochastic no-wait JSP with controllable processing times. So far, this problem has not been addressed in the literature and also its deterministic counterpart received only very little attention. However, the problem is of high practical relevance. Many industrial processes are subject to no-wait constraints (e. g., handling of hot or perishable goods, lack of storage) and very often the processing times can be controlled within certain bounds. No-wait constraints dramatically restrict the scheduling. Flexibility in at least some of the processing times can thus be beneficially used for both a tighter and a more robust schedule.

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Table 5: Comparison of deterministic and stochastic scheduling results for benchmark problems of stochastic no-wait JSP with controllable processing times.

data set	size	$\overline{C_m^{det}}$	$\overline{C_m^{stoch}}$	$\Delta_{\%}$	σ_m^{det}	σ_m^{stoch}	$\Delta_{\%}$	T_c
la26	10×20	2681.7	2689.9	0.31	8.7	5.2	-39.83	5.0
la27	10×20	2690.8	2788.7	3.64	4.8	6.0	26.42	4.9
la28	10×20	2642.7	2691.2	1.84	8.3	8.5	2.31	4.8
la29	10×20	2483.3	2431.6	-2.08	10.9	8.2	-24.41	5.0
la30	10×20	2660.3	2744.6	3.17	9.7	2.4	-75.50	5.1
la31	10×30	3687.0	3783.2	2.61	3.1	9.6	213.15	29.8
la32	10×30	4068.4	4192.1	3.04	10.2	1.6	-84.76	31.4
la33	10×30	3703.4	3642.9	-1.63	12.3	4.9	-60.04	31.4
la34	10×30	3682.9	3791.8	2.96	9.4	6.2	-33.78	28.1
la35	10×30	3864.2	3713.6	-3.90	15.0	6.1	-59.49	28.6
la36	15×15	2647.6	2790.3	5.39	9.0	2.8	-68.65	2.4
la37	15×15	2933.7	2995.8	2.12	5.8	6.9	18.93	2.5
la38	15×15	2584.1	2546.0	-1.48	7.6	5.1	-32.46	2.2
la39	15×15	2649.8	2721.3	2.70	7.6	2.8	-63.20	2.6
la40	15×15	2680.6	2660.1	-0.77	8.1	0.6	-92.31	2.7
abz7	15×20	1610.2	1613.5	0.21	4.0	1.5	-61.14	10.3
abz8	15×20	1657.7	1582.6	-4.53	2.7	3.6	31.45	11.1
abz9	15×20	1541.3	1623.2	5.32	4.4	1.1	-75.48	10.5
swv01	10×20	2288.2	2314.5	1.15	6.2	3.5	-44.22	5.7
swv02	10×20	2436.7	2436.0	-0.03	9.6	0.3	-97.08	5.7
swv03	10×20	2356.2	2404.3	2.04	7.6	1.4	-81.72	5.6
swv04	10×20	2446.6	2429.7	-0.69	7.5	1.8	-75.57	5.6
swv05	10×20	2411.3	2309.0	-4.24	4.7	3.2	-32.35	6.1
swv06	15×20	3306.9	3314.0	0.22	7.7	0.0	-100.00	11.3
swv07	15×20	3086.3	3136.7	1.63	5.7	3.1	-46.82	11.5
swv08	15×20	3375.5	3444.5	2.05	12.7	3.0	-76.32	12.0
swv09	15×20	3220.4	3169.3	-1.59	3.7	4.3	14.49	15.2
swv10	15×20	3415.7	3393.3	-0.66	7.5	1.5	-79.82	15.1
swv11	10×50	5462.3	5476.7	0.26	7.3	5.9	-19.41	341.5
swv12	10×50	5476.6	5416.0	-1.11	5.5	0.6	-89.95	332.4
swv13	10×50	5592.0	5507.3	-1.52	8.7	1.4	-84.37	331.5
swv14	10×50	5320.2	5309.3	-0.20	6.6	2.8	-57.44	341.5
swv15	10×50	5250.0	5227.8	-0.42	2.2	2.1	-4.72	330.9
swv16	10×50	6028.8	5844.9	-3.05	13.1	9.2	-29.89	310.5
swv17	10×50	5883.1	5590.7	-4.97	16.0	13.7	-14.43	312.9
swv18	10×50	5690.2	5762.0	1.26	16.4	5.7	-65.40	330.6
swv19	10×50	5957.6	5959.4	0.03	13.2	6.2	-52.90	327.3
swv20	10×50	5671.9	5518.7	-2.70	13.9	9.0	-35.22	302.4
yn1	20×20	2397.9	2381.5	-0.69	6.7	3.7	-45.51	19.6
yn2	20×20	2447.8	2300.4	-6.02	5.5	3.0	-45.64	17.9
yn3	20×20	2313.1	2307.0	-0.26	6.5	2.8	-57.38	18.9
yn4	20×20	2405.2	2445.1	1.66	4.9	2.9	-40.38	18.8
average				**0.03**			**-41.45**	

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

The proposed solution solves the deterministic problem by a decomposition into a timetabling and a sequencing subproblem. An extension in the deterministic timetabling adds extra safety margins to the scheduled times without delaying the end times of the jobs. In the sequence optimization subproblem, an extra penalty term is added to the cost function to punish uncertain tasks at intolerant positions, i. e., positions that have certainly or most probably a direct influence on the makespan.

Scheduling results were presented, first, for existing benchmarks of no-wait JSP with fixed processing times to demonstrate the outstanding performance of the presented scheduling solution compared to the state of the art. The results by Schuster (2006), Mokhtari et al. (2011), and Bożejko & Makuchowski (2009) are outperformed on average.

Second, scheduling results for a hot rolling mill demonstrate the practical relevance of the considered problem. The best configuration yielded an improvement of the mean makespan of 4.53 % and an improvement of the standard deviation of 49.8 % compared to the original sequence.

Finally, benchmark problems for stochastic no-wait JSP with controllable processing times were generated by modification of existing no-wait JSP benchmarks. For these benchmark problems, stochastic scheduling reduced the average standard deviation of the makespan by around 40 % while the mean makespan remained almost unchanged compared to the deterministic scheduling result.

Future research may address the consideration of additional safety margins for the occurrence of errors in the production process.

Acknowledgments

The overall support of the industrial research partner Plansee SE, especially by Michael Eidenberger-Schober, Joachim Resch, and Gernot Reichl, is gratefully acknowledged.

This research work has been performed in the EU project Power Semiconductor and Electronics Manufacturing 4.0 (SemI40), which is funded by the programme Electronic Component Systems for European Leadership (ECSEL) Joint Undertaking (grant agreement no. 692466) and the programme "IKT der Zukunft" (project no. 853343) of the Austrian Ministry for Transport, Innovation and Technology (bmvit) between May 2016 and April 2019. More information on IKT der Zukunft can be found at https://iktderzukunft.at/en/. Moreover, the project SemI40 is co-funded by grants from Germany, Italy, France, and Portugal.

References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, *34*, 391–401. doi:10.1287/mnsc.34.3.391.

Allahverdi, A. (2016). A survey of scheduling problems with no-wait in process. *European Journal of Operational Research*, *255*, 665–686. doi:10.1016/j.ejor.2016.05.036.

Aschauer, A., Roetzer, F., Steinboeck, A., & Kugi, A. (2017). An efficient algorithm for scheduling a flexible job shop with blocking and no-wait constraints. *IFAC-PapersOnLine*, *50*, 12490–12495. doi:10.1016/j.ifacol.2017.08.2056.

Aschauer, A., Roetzer, F., Steinboeck, A., & Kugi, A. (2018). Scheduling of a flexible job shop with multiple constraints. *IFAC-PapersOnLine*, *51*, 1293–1298. doi:10.1016/j.ifacol.2018.08.354.

[dataset] Beasley, J. E. (1990). jobshop1. OR-Library. URL: `http://people.brunel.ac.uk/ mastjjb/jeb/orlib/jobshopinfo.html`.

Bożejko, W., & Makuchowski, M. (2009). A fast hybrid tabu search algorithm for the no-wait job shop problem. *Computers & Industrial Engineering*, *56*, 1502–1509. doi:10.1016/j.cie.2008.09.023.

Framinan, J., & Perez-Gonzalez, P. (2015). On heuristic solutions for the stochastic flowshop scheduling problem. *European Journal of Operational Research*, *246*, 413–420. doi:10.1016/j.ejor.2015.05.006.

Glover, F., & Laguna, M. (1997). *Tabu Search*. Springer. doi:10.1007/978-1-4615-6089-0.

Grabowski, J., & Janiak, A. (1987). Job-shop scheduling with resource-time models of operations. *European Journal of Operational Research*, *28*, 58–73. doi:10.1016/0377-2217(87)90169-x.

Hall, N. G., & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, *44*, 510–525. doi:10.1287/opre.44.3.510.

Horng, S.-C., Lin, S.-S., & Yang, F.-Y. (2012). Evolutionary algorithm for stochastic job shop scheduling with random processing time. *Expert Systems with Applications*, *39*, 3603–3610. doi:10.1016/j.eswa.2011.09.050.

Jamili, A. (2016). Robust job shop scheduling problem: Mathematical models, exact and heuristic algorithms. *Expert Systems with Applications*, *55*, 341–350. doi:10.1016/j.eswa.2016.01.054.

Lenstra, J. K., & Rinnooy Kan, A. H. G. (1979). Computational complexity of discrete optimization problems. *Annals of Discrete Mathematics*, *4*, 121–140. doi:10.1016/s0167-5060(08)70821-5.

Lenstra, J. K., Rinnooy Kan, A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, *1*, 343–362. doi:10.1016/s0167-5060(08)70743-x.

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Macchiaroli, R., Mole, S., & Riemma, S. (1999). Modelling and optimization of industrial manufacturing processes subject to no-wait constraints. *International Journal of Production Research*, *37*, 2585–2607. doi:10.1080/002075499190671.

Mascis, A., & Pacciarelli, D. (2002). Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, *143*, 498–517. doi:10.1016/s0377-2217(01)00338-1.

Mokhtari, H., Nakhai, I., Abadi, K., & Zegordi, S. H. (2011). Production capacity planning and scheduling in a no-wait environment with controllable process times: An integrated modeling approach. *Expert Systems with Applications*, *38*, 12630–12642. doi:10.1016/j.eswa.2011.04.051.

Özgür, A., Uygun, Y., & Hütt, M.-T. (2020). A review of planning and scheduling methods for hot rolling mills in steel production. *Computers & Industrial Engineering*, . doi:10.1016/j.cie.2020.106606. In press.

Papoulis, A., & Pillai, S. U. (2002). *Probability, Random Variables and Stochastic Processes*. McGraw-Hill.

Pinedo, M. L. (2016). *Scheduling*. Springer. doi:10.1007/978-3-319-26580-3.

Samarghandi, H., ElMekkawy, T. Y., & Ibrahem, A.-M. M. (2013). Studying the effect of different combinations of timetabling with sequencing algorithms to solve the no-wait job shop scheduling problem. *International Journal of Production Research*, *51*, 4942–4965. doi:10.1080/00207543.2013.784410.

Schuster, C. J. (2006). No-wait job shop scheduling: tabu search and complexity of subproblems. *Mathematical Methods of Operations Research*, *63*, 473–491. doi:10.1007/s00186-005-0056-y.

Schuster, C. J., & Framinan, J. M. (2003). Approximative procedures for no-wait job shop scheduling. *Operations Research Letters*, *31*, 308–318. doi:10.1016/s0167-6377(03)00005-1.

Shabtay, D., & Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, *155*, 1643–1666. doi:10.1016/j.dam.2007.02.003.

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.

Appendix

Selection of the Starting Times

potential starting times

$$t_1,\ t_{j,n+1} - \overline{d}_{j,n},\ t_{j,n+1} - \underline{d}_{j,n} - tol,\ t_{lst}$$

pre-conditions

$$\begin{aligned} t_1 &\leq t_{lst} \\ t_{j,n+1} - \overline{d}_{j,n} &\leq t_{lst} \end{aligned}$$

necessary conditions

$$\begin{aligned} t_{j,n} &\geq t_1 \\ t_{j,n} &\geq t_{j,n+1} - \overline{d}_{j,n} \\ t_{j,n} &\leq t_{lst} \end{aligned}$$

goal

$$t_{j,n+1} - t_{j,n} = \underline{d}_{j,n} + tol$$

Table 6: Enumeration of the starting times which fulfill the pre-conditions. The second latest time always fulfills the necessary conditions and is closest to the stated goal.

t_{lst}	$\geq$	$t_{j,n+1} - \underline{d}_{j,n} - tol$	$\geq$	t_1	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$
t_{lst}	$\geq$	$t_{j,n+1} - \underline{d}_{j,n} - tol$	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$	$\geq$	t_1
t_{lst}	$\geq$	t_1	$\geq$	$t_{j,n+1} - \underline{d}_{j,n} - tol$	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$
t_{lst}	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$	$\geq$	$t_{j,n+1} - \underline{d}_{j,n} - tol$	$\geq$	t_1
t_{lst}	$\geq$	t_1	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$	$\geq$	$t_{j,n+1} - \underline{d}_{j,n} - tol$
t_{lst}	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$	$\geq$	t_1	$\geq$	$t_{j,n+1} - \underline{d}_{j,n} - tol$
$t_{j,n+1} - \underline{d}_{j,n} - tol$	$\geq$	t_{lst}	$\geq$	t_1	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$
$t_{j,n+1} - \underline{d}_{j,n} - tol$	$\geq$	t_{lst}	$\geq$	$t_{j,n+1} - \overline{d}_{j,n}$	$\geq$	t_1

$\underbrace{\qquad\qquad}_{= t_{j,n}}$

Post-print version of the article: A. Aschauer, F. Roetzer, A. Steinboeck, and A. Kugi, "Efficient scheduling of a stochastic no-wait job shop with controllable processing times," *Expert Systems with Applications*, vol. 162, p. 113 879, 2020. DOI: 10.1016/j.eswa.2020.113879
The content of this post-print version is identical to the published paper but without the publisher's final layout or copy editing.