

BERGISCHE
UNIVERSITÄT
WUPPERTAL

Bergische Universität Wuppertal

Fakultät für Wirtschaftswissenschaften
Schumpeter School of Business and Economics

Integrated scheduling of jobs and maintenance activities under stochastic deterioration

Inaugural Dissertation
for obtaining the academic degree of

Doctor rerum oeconomicarum (Dr. rer. oec.)

Submitted by

Antonia Arter (geb. Thiemeyer)

Wuppertal, July 2026

First Reviewer:
Prof. Dr. Dirk Briskorn
Lehrstuhl für Produktion und Logistik
Schumpeter School of Business and Economics, Bergische Universität Wuppertal

Second Reviewer:
Prof. Dr. Stefan Bock
Lehrstuhl für Wirtschaftsinformatik und Operations Research
Schumpeter School of Business and Economics, Bergische Universität Wuppertal

Acknowledgements

Five years ago, I set out with a rough idea and a great deal of naive optimism, convinced I would be done within three years. What followed was longer, harder, and far more rewarding than I could ever have imagined. Now that the thesis is finally complete, I want to thank the people without whom I would never have reached the end.

I would like to thank my supervisor, Prof. Dr. Dirk Briskorn, for his expertise and for the valuable feedback he provided throughout this research. His input consistently sharpened my arguments and helped me see problems from new angles.

I would also like to thank Prof. Dr. Stefan Bock for serving as the second examiner of this dissertation and for the time and effort he devoted to reviewing my work.

I owe special thanks to my co-authors, whose collaboration shaped much of this research. I am particularly grateful to Prof. Dr. Jochen Gönsch, who was involved from the very beginning and whose support was invaluable in helping me find my footing at the start. I would also like to thank Prof. Dr. Florian Jaehn and Dr. Andreas Hipp for our joint work on a survey article – long in the making as an idea, but finally set in motion over a cold drink in Bad Windsheim.

I would also like to thank my colleagues Rebecca, Cedric, and Nik, who always had an open ear for a research discussion but were just as ready for the moments beyond work. My thanks go equally to Sabine, and Lennart, who made these years so enjoyable and, each from their own area of expertise, were a great help in many questions of administration, coding, and teaching.

I am especially grateful to Rea, who shared not only an office with me but also countless discussions on my research, all the emotional highs and lows of these years and who has become one of my closest friends.

Finally, I would like to thank my family and friends for always being there for me. My greatest thanks, however, belong to my beloved husband, Christopher, who kept me from ever giving up and who never stopped believing in me. And, Christopher – I can't wait for what lies ahead, with the whole world waiting for the two of us.

Wuppertal, July 2026 *Antonia Arter*

Contents

Manuscript

Abbreviations I

1. Introduction 1
   1.1. Motivation . . . 1
   1.2. Research question and scope . . . 3
   1.3. Outline . . . 4

2. Research contributions 7
   2.1. Article I . . . 7
   2.2. Article II . . . 8
   2.3. Article III . . . 10
   2.4. Article IV . . . 12

3. Conclusion 15

References 17

Appendix 19

A. Article I – State-dependent machine availability in scheduling 21
   1. Introduction . . . 23
   2. Maintenance in machine scheduling . . . 26
      2.1. Machine state . . . 27
      2.2. Deterioration . . . 28
      2.3. Maintenance . . . 29
      2.4. Performance . . . 30
   3. Processing times independent of the machine state . . . 31
      3.1. Fixed machine deterioration . . . 32
      3.2. Deterioration of machine state linear in processing time . . . 34
      3.3. Job individual machine deterioration . . . 36

4. Processing times depending on the machine state . . . 39
4.1. Fixed machine deterioration . . . 39
4.1.1. Position-dependent, job-independent deterioration of machine state . . . 41
4.1.2. Position-dependent, job-dependent deterioration of machine state . . . 43
4.2. Deterioration of machine state linear in processing time . . . 45
5. Summary of the results and future research directions . . . 49
6. Conclusion . . . 53

B. Article II – Scheduling maintenance activities subject to stochastic job-dependent machine deterioration 61
1. Introduction . . . 63
2. Maintenance planning for a given job sequence . . . 69
2.1. Problem definition . . . 69
2.2. Computational complexity . . . 73
2.3. On the relation between transparent and non-transparent machines . 76
3. Solution approach for MPP-N . . . 78
3.1. Deterministic dynamic program . . . 78
3.2. Calculating transition costs . . . 80
4. Solution approach for MPP-T . . . 84
5. Computational study . . . 89
5.1. Instances . . . 90
5.2. Numerical results . . . 93
5.2.1. Run times . . . 93
5.2.2. Ratio range . . . 94
5.2.3. Distribution-independent parameters . . . 96
5.2.4. Distribution-dependent parameters . . . 98
6. Conclusion and outlook . . . 100

C. Article III – Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration 105
1. Introduction . . . 107
2. Literature review . . . 110
3. Optimization problem and model formulation . . . 116
3.1. Problem definition . . . 116
3.2. DP model . . . 119
4. Solution approaches . . . 122
4.1. The approximate dynamic programming approach . . . 122
4.2. Discretization approach . . . 125

4.3. Threshold policies . . . 126
4.3.1. Static threshold policies . . . 126
4.3.2. Dynamic threshold policies . . . 127
5. Computational study . . . 127
5.1. Instances . . . 128
5.1.1. Continuous probability distributions for deterioration . . . 128
5.1.2. Instance generation . . . 130
5.2. Numerical results . . . 131
5.2.1. Configuration of threshold policies . . . 132
5.2.2. Robustness of the ADP approach . . . 133
5.2.3. Running times . . . 135
5.2.4. Policy performance . . . 135
5.2.5. Summary of results and practical implications . . . 140
6. Conclusion and outlook . . . 140

D. Article IV – Integrated scheduling of jobs and maintenance activities on a single machine exposed to stochastic job-dependent machine deterioration 147
1. Introduction . . . 150
2. Literature review . . . 152
3. Problem definition . . . 156
4. Solution approaches for JSMPP . . . 158
4.1. PMA policies for given job sequences . . . 158
4.2. BaB approach for solving JSMPP . . . 159
4.2.1. Search tree and node selection strategies . . . 159
4.2.2. Lower bound . . . 161
4.3. Tabu search . . . 165
4.4. Heuristic priority rules . . . 166
4.5. Summary of solution approaches for JSMPP . . . 167
5. Computational study . . . 167
5.1. Instance generation . . . 168
5.2. Parametrization of solution approaches for JSMPP . . . 169
5.3. Numerical results . . . 170
5.3.1. Run times . . . 170
5.3.2. Solution quality . . . 174
5.3.3. Results for the heuristic approaches on larger instances . . . 177
6. Conclusion . . . 179

Manuscript

This manuscript, prepared in accordance with § 12 (4) of the Doctoral Regulations, establishes the scientific context of the four articles that constitute this cumulative dissertation. The full texts of the articles are included in the appendix.

Abbreviations

ADP	Approximate dynamic program
BaB	Branch and Bound
DP	Dynamic program
EMA	Emergency maintenance activity
ETC	Expected total completion time
ETWC	Expected total weighted completion time
JSMPP	Job sequencing and maintenance planning problem
MA	Maintenance activity
ML	Maintenance level
MPP-C	Maintenance planning problem with continuous deterioration
MPP-N	Maintenance planning problem on a non-transparent machine
MPP-T	Maintenance planning problem on a transparent machine
PMA	Planned maintenance activity

1. Introduction

1.1. Motivation

Conventional wisdom about maintenance is contradictory. There are commonly known proverbs recommending acting before damage occurs, and others recommending exactly the opposite by leaving a working system running as long as there is no true need to intervene. Which one is recommended may differ for a given situation, depending on whom one asks. Acting early through preventive maintenance may avoid costly disruptions but requires planned downtime and additional maintenance effort. Waiting longer may increase machine utilization in the short term but comes at the risk of unexpected failures and processing interruptions. Efficient production processes, therefore, rely on an appropriate mixture of both strategies. This dissertation investigates a job and maintenance scheduling problem from the perspective of machine scheduling theory.

Machine scheduling theory addresses problems related to allocating jobs to machines and evaluating the resulting schedules in terms of a suitable objective function. As standard references on machine scheduling one can name the works of Brucker (2004) and Pinedo (2012). In large parts of the literature in this field the availability of a machine is taken for granted. In practice this assumption is often unrealistic. Machines are exposed to deterioration arising from processing and, therefore, require maintenance, repairs, cooling periods, or other activities that temporarily render the machines unavailable. Consequently, scheduling problems with limited machine availability have attracted increasing attention in the last decades as the surveys of Sanlaville and Schmidt (1998), Ma et al. (2010) and Geurtsen et al. (2023) show.

The existing literature addresses limited machine availability due to maintenance from several perspectives. A substantial body of research considers exogenously imposed unavailability periods, such as fixed maintenance windows (Mosheiov et al. (2018)), periodic maintenance (Yu et al. (2014)), or externally specified machine downtimes (Zhong et al. (2014)). In contrast, others integrate maintenance scheduling explicitly into the overall scheduling scope (Sun et al. (2023)). Thereby, the *maintenance activities* (MAs) themselves are characterized by a range of features that have been studied in various combinations. In the following, a few of these features are presented to illustrate the variety of maintenance scheduling problems considered in the literature. One feature is the number of MAs that may or must be scheduled over the planning horizon. Many problem settings

impose restrictions such as allowing at most one MA to be scheduled (Mor and Mosheiov (2015)) or bounding the number of scheduled MAs (Rustogi and Strusevich (2014)). Further, the duration of a MA may be fixed (Wang et al. (2018)) or depend on the machine's workload so far (Zhang et al. (2018)). The effect of the MA on the machine state rages from a full restoration (Liu et al. (2017)) to a partial improvement by an arbitrary increment (Rustogi and Strusevich (2015)). It gets obvious that already these few features span a wide range of problem settings and give an impression on the structure and difficulty of the resulting optimization problems. Further, there is a large stream of literature investigating scheduling problems within the closely related setting of rate-modifying activities. Such activities are not necessarily needed to keep the system running but improve the machine's performance, see for example Lee and Leon (2001), Zhu et al. (2017) and the survey of Geurtsen et al. (2023).

When scheduling problems involving maintenance are investigated, the machine state, i.e. its condition, is often treated as a relevant component of the model. The evolution of this state is frequently assumed to depend on time rather than on specific properties of the jobs processed. This is the case, for example, in Mosheiov and Sidney (2010), where a setting with a worsening of the machine state linear in time is considered. Under this assumption, which underlies large streams of the literature, the job schedule and the deterioration process are only weakly connected.

In many real-world production systems, however, the machine state is directly affected by the jobs being processed. Different jobs may place different loads on the machine, resulting in varying impacts on its state. The work of Bock et al. (2012) takes exactly this observation into account and introduces an optimization problem of scheduling jobs on a single machine with deterministic job-dependent deterioration. While their setting captures the dependence on individual jobs, it assumes deterministic deterioration, which is often not the case in practice, as the actual deterioration caused by a job is often uncertain. The following example highlights such a setting and motivates the consideration of stochastic job-dependent deterioration.

Consider a wood processing machine. The machine is equipped with a cutting tool, such as a blade. Depending on the number, type and complexity of cuts required, different jobs cause the blade to deteriorate to different extents. Furthermore, since natural materials vary in hardness and grain structure, the exact amount of deterioration caused by a job can not be predicted perfectly and is therefore subject to uncertainty. Between two consecutive jobs, the blade may be sharpened as a preventive MA to restore its state. Omitting such maintenance may increase the production efficiency in the short term but also increases the risk that the blade becomes unexpectedly unavailable within processing one of the following jobs. If so, the current job has to be interrupted unexpectedly such that the blade can be either sharpened or replaced causing a larger delay than a comparable preventive MA. The central question is therefore not only how jobs should be scheduled on the machine

but also when MAs should be performed with regard to the uncertain evolution of the machine state.

Taking the foregoing discussions and the existing literature into account, it becomes apparent that comparably little is known about how to schedule jobs and maintenance when deterioration is stochastic and job-dependent. Addressing this gap constitutes the central motivation of this dissertation.

The relevance of this optimization problem is not merely theoretical, nor is it confined to the illustrative example given above. In discussions with a pharmaceutical manufacturer in Germany, a comparable situation is observed, in which production campaigns are fixed months, sometimes even years, in advance, with uncertainty in the deterioration of the machine. In such environments, adjustments to the job sequence are often not feasible, whereas maintenance decisions remain operationally flexible. Delays caused by MAs, and especially by machine breakdowns, directly affect the completion times of all subsequent jobs, so that they should already be taken into account when delivery dates are communicated to customers, in order to ensure reliable delivery commitments. Beyond meeting delivery commitments, such considerations also matter with respect to the capital tied up in production, which makes it worthwhile to complete jobs as early as possible. Consequently, even in production systems with a fixed job sequence, the question of how MAs should be planned is highly relevant.

1.2. Research question and scope

As already anticipated in the previous section, this dissertation investigates a single machine setting where a set of jobs has to be processed. In processing the jobs, they deteriorate the machine to a stochastic job-dependent extent, so that the machine state worsens. The machine state can be restored by MAs. Two types of MAs are distinguished, namely *planned maintenance activities* (PMAs), which can be scheduled deliberately whenever the machine is not currently processing a job, and *emergency maintenance activities* (EMAs), which are triggered automatically by a machine breakdown. Within this machine environment the two-fold need for optimization of job sequencing and PMA scheduling arises, while EMAs occur only as an involuntary consequence of a machine breakdown.

The underlying optimization problem is rather easy to state but, as this dissertation will show, difficult to solve. A relevant component within this setting is the availability of information. By assuming stochastic job-dependent deterioration, the deterioration caused by a job becomes fully known once the processing of the job has been completed. Since this information only becomes available afterwards, decisions on PMAs have to be taken before the deterioration of the next job is known. The need for a policy deciding

on PMAs in light of the current machine state and the deterioration properties of the following jobs arises.

To the best of our knowledge, such a setting has received only limited attention in the machine scheduling literature so far. This gap was identified in the course of a broader review of the literature on state-dependent machine unavailability in scheduling. Surveying this wider field, which itself forms part of this dissertation, revealed that the combination of stochastic job-dependent deterioration including the scheduling of PMAs has remained largely unexplored. This gap marks the starting point for the methodological contributions of this dissertation. Specifically, this dissertation addresses the research question of how the integrated scheduling of jobs and PMAs on a single machine can be approached when the machine state deteriorates to a stochastic job-dependent extent whenever a job is processed. The objective is to minimize the expected total (weighted) completion time of jobs. Due to the complexity of the resulting optimization problem, this overarching question is approached incrementally by investigating different optimization problems that progressively relax the underlying model assumptions. For each of them, the computational complexity is settled and appropriate solution approaches are developed. As outlined above, this line of research is embedded in the broader field of scheduling under state-dependent machine availability, within which the developed optimization problems are positioned.

1.3. Outline

Reflecting a conventional format for cumulative dissertations, this dissertation comprises the findings of four interconnected articles. Specifically, these are the articles

I Arter, A., Briskorn, D., Hipp, A., Jaehn, F. (2026b). State-dependent machine availability in scheduling. *International Transactions in Operational Research*, *33*(6), 3769–3799. https://doi.org/10.1111/itor.70183,

II Briskorn, D., Gönsch, J., Thiemeyer, A. (2024). Scheduling maintenance activities subject to stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *319*(1), 62-78. https://doi.org/10.1016/j.ejor.2024.06.030,

III Arter, A., Briskorn, D., Gönsch, J. (2026a). Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *335*(1), 67-82. https://doi.org/10.1016/j.ejor.2026.04.013,

IV Arter, A., Briskorn, D. (2026). Integrated scheduling of jobs and maintenance activities on a single machine exposed to stochastic job-dependent machine deterioration. Working Paper.

The results of the individual articles are briefly presented and placed within the overall context of the dissertation in Section 2. Rather than providing an extensive summary of each article's contents, the aim is to focus on the developed methodology and the key findings. The full articles are reprinted in Appendices A–D. The presentation of contributions starts with Article I, which provides a literature review on the broader field of state-dependent machine availability in scheduling. After this, the methodological contributions of Articles II–IV are discussed. Finally, Section 3 concludes the manuscript and gives an outlook on future research.

The following remarks provide guidance on the structure and referencing conventions used throughout this manuscript. Since Article I provides a comprehensive review of the relevant literature, no separate literature review is included in this manuscript. Throughout this manuscript, the four articles are referenced as Article I to Article IV in accordance with the numbering above.

2. Research contributions

This section provides an overview of the four articles this dissertation comprises and establishes their scientific context. Article I surveys the existing literature on scheduling problems involving state-dependent machine availability. Articles II-IV deal with scheduling problems on a single machine subject to stochastic job-dependent deterioration. The shared setting allows their models and solution methods to be discussed in a unified manner, while each article addresses a distinct extension of the underlying problem setting.

2.1. Article I

Before presenting the methodological contributions of this dissertation in Sections 2.2 - 2.4, the focus first turns to the broader research field. Article I surveys the literature in the field of state-dependent machine availability in scheduling. Here, machine unavailability periods are driven by the machine state rather than by elapsed time, which sets the field apart from the large body of time-based maintenance scheduling covered by earlier surveys. At the same time, it provides a conceptual framework into which the investigated optimization problems of this dissertation fit.

One of the key contributions of Article I is a classification scheme that extends the established three-field notation of Graham et al. (1979) to capture the maintenance features of a scheduling model. These features comprise the range of the machine state, the type of deterioration a job causes (constant, linear in its processing time, or individual), the properties of the PMAs (whether their duration depends on the machine state and whether they restore the machine fully or only partially), and whether a job's processing time is affected by the machine state. Using this scheme, the survey classifies more than ninety problems from the existing literature.

A few observations stand out. Across the reviewed problems, imposing any form of state-dependent availability tends to change the complexity status relative to the corresponding problem with unrestricted availability. Further, settings including individual job-dependent deterioration turn out to be comparatively underrepresented. Moreover, the combination of preventive and reactive maintenance, PMAs and EMAs, under stochastic deterioration is identified as a largely unexplored research area.

Within the developed scheme, this open area is the gap this dissertation sets out to address. More specifically, a set of optimization problems that can be located within this gap is considered. Among the features the scheme can capture, the problems considered within this dissertation are single machine problems with a bounded machine state. The state deteriorates as jobs are processed and the deteriorations are job-dependent. PMAs restore the machine to its perfect state with a duration that depends on the current machine state. On top of the job-dependency, the deterioration is additionally assumed to be stochastic. As only a limited share of the literature covers stochastic deterioration, this feature is not directly reflected in the four-field notation introduced in Article I.

All optimization problems investigated in the scope of this dissertation fall into the aforementioned category, which is highlighted in the survey as both practically relevant and comparatively sparsely studied. To summarize, Article I both frames the dissertation and locates its specific contribution within the existing literature.

2.2. Article II

Article II marks the starting point of the methodological contributions by investigating maintenance scheduling on a single machine subject to stochastic job-dependent deterioration. Assuming a fixed job sequence and stochastic deterioration following discrete probability distributions with integer realizations allows the maintenance decision to be analyzed independently of job sequencing. This separation provides insights into the fundamental structure of the problem. The objective is to minimize the *expected total completion time* (ETC) of jobs.

Since the machine environment considered in Article II also forms the basis of Articles III and IV, it is briefly introduced in the following. This provides a common foundation that will be referred to throughout the remainder of this manuscript. The machine state is measured by a numeric *maintenance level* (ML) ranging between zero and a maximum ML $l^{\max}$. A machine with ML $l^{\max}$ is considered to be in perfect shape. At the beginning of the planning horizon the machine is handed over with an initial ML l_0. Two types of MAs, as introduced in Section 1.1, improve the ML of the machine. PMAs can be scheduled at any time the machine is not currently processing a job, that is, at the beginning of the planning horizon, between any pair of consecutive jobs or after completing the last job. A PMA restores the ML to $l^{\max}$ with a duration linear in the ML improvement. EMAs are triggered automatically whenever the ML would fall below zero during job processing. They restore the ML to $l^{\max}$ and require a fixed duration exceeding PMA durations. The interrupted job can be resumed after the EMA is completed. Neither type of MA can be interrupted, and both delay all subsequent jobs.

Article II distinguishes between two problem variants based on the available information about the machine's ML. In the non-transparent machine setting, decisions on PMAs must be made at the beginning of the planning horizon, before the evolution of the ML is realized. In the transparent machine setting, PMA decisions are made after the completion of each job based on the currently observed ML. Consequently, the two variants differ not only in the available information but also in the flexibility of scheduling PMAs and thus in the way PMA decisions are represented. A solution for the *maintenance planning problem on a non-transparent machine* (MPP-N) is, then, a binary vector indicating for each job whether or not a PMA is scheduled immediately before starting the processing of the job. Since the ML cannot be observed, the resulting PMAs are fixed in advance and identical for every possible evolution of the ML. On the transparent machine, in contrast, each decision depends on the ML reached after the preceding job. Therefore, a solution for the *maintenance planning problem on a transparent machine* (MPP-T) specifies a PMA decision for every combination of job and ML. Hence, it represents a detailed PMA policy rather than a fixed PMA plan. Every solution for MPP-N is also a solution for MPP-T, namely one that ignores the additional information of the current ML. For both variants, the objective to be minimized remains the ETC.

The article proves that both problem variants are NP-hard, even if each job admits no more than two possible deterioration realizations. Complementing this, two structural results relate the two investigated problem variants to each other. The optimal objective value of MPP-T never exceeds that of MPP-N, and the infimum of the ratio between both objective values equals zero, i.e. the advantage gained from observing the machine's ML during the processing and deciding on PMAs based on the observed ML can be arbitrarily large.

Both problem variants are solved to optimality using dynamic programming. For MPP-N a solution can be interpreted as a partition of the job sequence into batches, that is, maximal subsequences of jobs without an intermediate PMA. This yields a deterministic *dynamic program* (DP) whose transition costs correspond to the expected contribution to the objective function of a single batch. Transition costs are determined by a nested stochastic DP tracking the ML development within a batch. An optimal solution is directly obtained from the fully evaluated DP. For MPP-T, the observable ML serves as a state variable, so that a single stochastic DP suffices. Its stages alternate between decision nodes, where PMA decisions are made, and stochastic nodes, where deterioration realizes. The resulting optimal decisions define the optimal PMA policy. Both algorithms run in pseudo-polynomial time, thereby settling the complexity status of both problem variants.

Within a computational study the proposed solution approaches for MPP-T and MPP-N are tested on a instance set covering several types of deterioration distributions. All instance classes are solved to optimality within short run times, whereby solving MPP-T requires considerably less run time than solving MPP-N. Especially, the DP formulation

clearly outperforms an additionally proposed linear programming formulation. Beyond run times, the study quantifies the value of machine transparency. For a substantial share of the instance classes both machine types yield identical objective values, so that observing the ML offers no benefit at all. However, for instance classes exposed to unlikely but severe deteriorations, transparency pays off especially when EMAs have significantly longer durations than PMAs.

Beyond its direct contribution, Article II establishes the methodological and conceptual foundation for the subsequent articles. In particular, it develops efficient solution approaches for deriving optimal PMA policies under discretely distributed deterioration for fixed job sequences, which can serve as subroutines once the job sequence is relaxed. One more conclusion can be drawn from the results of Article II that shaped the further development of this dissertation. The results demonstrate that assuming a transparent machine can substantially decrease the optimal ETC. In particular, PMA decisions based on the current ML become increasingly beneficial as deterioration causes more severe delays in the production process. For this reason, the remainder of this dissertation focuses exclusively on transparent machines.

2.3. Article III

Article III extends the transparent machine setting introduced in Article II by generalizing the deterioration model from discrete to continuous probability distributions. This refinement better reflects real-world deterioration processes. The fixed job sequence is retained at this point but will be relaxed in Article IV of this dissertation.

The resulting optimization problem is therefore the problem of scheduling PMAs on a single machine with a fixed job sequence, involving stochastic job-dependent deterioration that follows continuous probability distributions. All remaining assumptions concerning the machine environment stated in Article II remain valid. As before, the objective is to minimize the ETC of jobs.

If the stochastic assumption on deteriorations is adjusted, the exact DP proposed in Article II can no longer be applied directly. Since the ML now takes values in the real-valued interval $[0, l^{\max}]$, the state space becomes continuous, and a solution to the *maintenance planning problem with continuous deterioration* (MPP-C) is a PMA policy assigning a decision to every combination of job and ML in $[0, l^{\max}]$. The DP formulation itself remains unchanged, but the continuous state space no longer allows for the DP evaluation proposed in Article II. Since MPP-C generalizes MPP-T, its NP-hardness carries over immediately.

To solve MPP-C, three classes of solution approaches are developed. An *approximate*

dynamic program (ADP) replaces the DP value function by a stage-wise approximation based on a linear combination of basis functions. Thereby, three configurations differing in the set of basis functions are distinguished.
Secondly, a discretization approach is developed that approximates the continuous deterioration distributions by discrete ones and rounds the ML to a finite grid, so that the exact DP of Article II can be applied. Its solution is then translated into a policy for the continuous problem by interpolation between grid points. With sufficiently fine discretizations, the resulting solutions approximate the optimum solution arbitrarily close.
Finally, two heuristic threshold policies are proposed, which trigger a PMA whenever the ML falls below a potentially job-dependent threshold. The static threshold policy fixes this threshold relative to $l^{\max}$. The dynamic threshold policy sets the threshold based on the probability that the next job triggers an EMA and by doing so can take the decision maker's risk appetite into account.

The computational study shows that the discretization approach provides the strongest results among all considered solution approaches. The results further suggest that, once the discretization is sufficiently fine, additional refinements no longer improve the solution quality.
The best performing ADP configuration closely approximates the results of the discretization approach while requiring substantially less memory. The dynamic threshold policy provides surprisingly good results at negligible computational cost. In contrast, the static threshold policy is not competitive which highlights the importance of incorporating the deterioration distribution of the upcoming job into PMA decisions.

Article III, thus, extends the setting of Article II from discrete to continuous stochastic deterioration. This is an important step within this dissertation since continuous deterioration illustrates a more natural modeling assumption especially with regard to technical machine components. By introducing a stochastic DP formulation and suitable approximation methods, Article III shows how well performing PMA policies can be obtained for fixed job sequences. In particular, the discretization approach needs to be highlighted since it provides the methodological link back to the discrete framework proposed in Article II. Once the continuous distributions are discretized sufficiently finely the approximated solution approaches the continuous optimum solution. In addition, Article III shows that the dynamic threshold policy achieves near-optimal results at negligible cost, which makes it the natural recommendation for practice.
These insights prepare the basis for Article IV, where the job sequence itself becomes part of the decision problem and candidate job sequences shall be evaluated using the corresponding optimal PMA policy.

2.4. Article IV

Article IV extends the final step of this dissertation with regard to the investigated optimization problems by relaxing the fixed job sequence and including weights for the jobs to be scheduled. An optimum solution for the resulting optimization problem, referred to as the integrated *job sequencing and maintenance planning problem* (JSMPP), is now a pair consisting of a job sequence and a PMA policy such that the *expected total weighted completion time* (ETWC) is minimized. Among the already investigated optimization problems, JSMPP is therefore the most generalized optimization problem considered in the course of this dissertation. With regard to the computational complexity it can be fallen back on the results of the foregoing articles, rendering JSMPP to be NP-hard as well.

Within Article IV, different solution approaches are developed, all sharing the procedure of determining job sequences and subsequently deriving PMA policies using the solution approaches developed in the foregoing articles. For the continuous deterioration setting, the discretization approach from Article III is applied with a sufficiently fine discretization, yielding PMA policies that are regarded as optimal.

Since the job sequence is no longer fixed, a method for exploring the space of sequences is required. Therefore, Article IV develops an exact *Branch and Bound* (BaB) framework based on a lower bound suitable for JSMPP. Sequences are constructed from the last position backwards. The lower bound construction is based on a method that separates the already fixed jobs from the unscheduled ones and decouples their contributions such that the contribution of the scheduled jobs can be computed using the solution approaches of the foregoing articles optimally. Different node selection strategies are introduced resulting in a total of three BaB approaches, of which two are exact solution approaches. The set of investigated solution approaches is complemented by further heuristics, namely a tabu search and a priority rule based on the jobs' processing times and expected deterioration. Across all these solution approaches, the optimum PMA policy for an exact evaluation of a sequence may also be replaced by the dynamic threshold policy defined in Article III offering heuristic alternatives.

The computational study confirms that the exact BaB approaches solve instances of manageable size to optimality but run times scale exponentially in the number of jobs. In contrast, the heuristics extend to realistic instance sizes with the tabu search performing best among all heuristics. The heuristic BaB framework remains applicable for instances of larger size and provides solutions with low ETWCs. Two key insights emerge from the computational study. First, obtaining high-quality solutions requires both an accurate evaluation of candidate job sequences and a thorough exploration of the space of job sequences. Second, deterioration must be explicitly considered in the sequencing decision, since neglecting it has a substantial impact whenever maintenance delays dominate job

processing times.

The main contribution of Article IV is to extend the research from isolated maintenance planning to the integrated scheduling problem combining job and PMA scheduling under stochastic job-dependent deterioration. The efficient solution approaches developed for this problem build directly on the methods established in Articles II and III for determining optimal PMA policies under either discrete or continuous deterioration for fixed job sequences. Together, the three Article II-IV form the methodological core of this dissertation, progressively extending the scope from maintenance planning to the integrated scheduling of jobs and PMAs with each contributing a specific optimization problem.

3. Conclusion

This dissertation studies the integrated scheduling of jobs and PMAs under stochastic job-dependent deterioration on a single machine. The survey Article I embeds the entire research line of this dissertation into the broader field of state-dependent machine availability in scheduling and, through its classification scheme, locates the specific contribution of this dissertation. From this field, one particular machine setting is selected and investigated in greater depth throughout the remaining articles. Starting from a tightly constrained problem setting, the investigated optimization problem is relaxed step by step. Each of the Articles II-IV generalizes its predecessor. Table 3.1 presents the machine settings, the resulting optimization problems, the complexity results and the developed solution approaches.

	Article II	**Article III**	**Article IV**
Machine setting	single machine	single machine	single machine
Machine transparency	transparent, non-transparent	transparent	transparent
Job sequence	fixed	fixed	decision variable
Deterioration	job-dependent, discrete	job-dependent, continuous	job-dependent, discrete/continuous
Objective function	ETC	ETC	ETWC
Optimization Problem	MPP-N, MPP-T	MPP-C	JSMPP
Complexity	NP-hard	NP-hard	NP-hard
Solution approaches	DP	ADPs, discretization, threshold policies	BaB approaches, tabu search, priority rules

Table 3.1. Overview of the optimization problems studied in this dissertation, their complexity results, and the developed solution approaches.

Article II isolates the maintenance decision on a fixed job sequence with discretely distributed deterioration, settles the complexity of both the non-transparent and the transparent variant, and provides exact DPs fast enough to serve as subroutines elsewhere. Article III retaines the fixed job sequence but replaces discrete by continuous deterioration, which forfeits the combinatorial structure the DP developed in Article II relies on. Several solution approaches are developed among which a simple dynamic threshold policy proves to be a strong and easy to implement solution and a discretization approximates

the optimum arbitrarily close. Article IV finally resolves the fixed sequence, coupling the PMA planning with a weighted job sequencing decision, and contributes an exact BaB framework whose lower bound is built precisely around the evaluation of fixed job sequences prepared in the first two articles.

Several insights recur across the articles rather than belonging to any single one. Observing the machine state is valuable, but its worth is highly situational. In particular, it pays off mainly when rare but severe deteriorations are possible and EMAs come to a high cost. The observable ML is a sufficient state descriptor throughout, which is what makes the PMA scheduling problem for fixed job sequences tractable and, in turn, reusable as a module as soon as the job sequence has to be determined. Across settings, the accuracy with which a candidate solution is evaluated proves as decisive for solution quality. Taken together, this dissertation shows that the PMA scheduling, though even hard in isolation, admits fast and modular solution methods that carry over to the integrated job and PMA scheduling problem.

Several research directions remain open. First, the presented methods based on the DP formulation can be easily adopted to different problem variants. E.g., it is conceivable that the duration of PMAs is stochastic, that delays arise due to setup times for PMAs, or that jobs must either be restarted as soon as an EMA occurs or may be interrupted for PMAs. By adding these features, the valuation basis of the procedure changes, but the proposed framework can be adopted. As long as an extension can be incorporated into the DP formulations derived in Articles II and III, the resulting solution methods remain applicable.
Second, it might be interesting to consider a problem variant where deteriorations do not follow static distributions. This scenario is reasonable, for instance, if the distribution depends on the experiences in the production process so far, especially when the deterioration distributions are estimated a priori and the information collected during the processing of jobs helps to re-estimate them.
Third, so far the continuous structure of the problem is not fully exploited. As the dynamic threshold policy performs very well, future work could try to analytically determine optimal parameter values for similar policies, which opens a broad field in which the continuous structure of the ADP developed in Article III may prove beneficial.
Fourth, objective functions other than the ETC or ETWC could be studied, both within the settings considered in this dissertation and in the optimization problem extensions outlined above.

Finally, beyond the direct scope of this dissertation, the survey of Article I points to the still largely unexplored combination of preventive and reactive maintenance under stochastic deterioration, and it remains close at hand to transfer the results of this dissertation to real-world applications and to investigate their applicability using real data sets.

References

Arter, A., Briskorn, D. (2026). Integrated scheduling of jobs and maintenance activities on a single machine exposed to stochastic job-dependent machine deterioration. Working Paper.

Arter, A., Briskorn, D., Gönsch, J. (2026a). Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *335*(1), 67–82.

Arter, A., Briskorn, D., Hipp, A., Jaehn, F. (2026b). State-dependent machine availability in scheduling. *International Transactions in Operational Research*, *33*(6), 3769–3799.

Bock, S., Briskorn, D., Horbach, A. (2012). Scheduling flexible maintenance activities subject to job-dependent machine deterioration. *Journal of Scheduling*, *15*(5), 565–578.

Briskorn, D., Gönsch, J., Thiemeyer, A. (2024). Scheduling maintenance activities subject to stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *319*(1), 62–78.

Brucker, P. (2004). Scheduling Algorithms. Springer. 4th ed.

Geurtsen, M., Didden, J., B.H.C., Adan, J., Atan, Z., Adan, I. (2023). Production, maintenance and resource scheduling: A review. *European Journal of Operational Research*, *305*(2), 501–529.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Annals of Discrete Mathematics. Elsevier. volume 5, pp. 287–326.

Lee, C.Y., Leon, V.J. (2001). Machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, *128*(1), 119–128.

Liu, C., Fan, Y., Zhao, C., Wang, J. (2017). Multiple common due-dates assignment and optimal maintenance activity scheduling with linear deteriorating jobs. *Journal of Industrial and Management Optimization*, *13*(2), 713–720.

Ma, Y., Chu, C., Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & industrial engineering*, *58*(2), 199–211.

Mor, B., Mosheiov, G. (2015). Scheduling a deteriorating maintenance activity and due-window assignment. *Computers & Operations Research*, *57*, 33–40.

Mosheiov, G., Sarig, A., Strusevich, V.A., Mosheiff, J. (2018). Two-machine flow shop and open shop scheduling problems with a single maintenance window. *European Journal of Operational Research*, *271*(2), 388–400.

Mosheiov, G., Sidney, J.B. (2010). Scheduling a deteriorating maintenance activity on a single machine. *Journal of the Operational Research Society*, *61*(5), 882–887.

Pinedo, M.L. (2012). Scheduling. volume 29. Springer.

Rustogi, K., Strusevich, V.A. (2014). Combining time and position dependent effects on a single machine subject to rate-modifying activities. *Omega*, *42*(1), 166–178.

Rustogi, K., Strusevich, V.A. (2015). Single machine scheduling with time-dependent linear deterioration and rate-modifying maintenance. *Journal of the Operational Research Society*, *66*(3), 500–515.

Sanlaville, E., Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, *35*, 795–811.

Sun, X., Liu, T., Geng, X.N., Hu, Y., Xu, J.X. (2023). Optimization of scheduling problems with deterioration effects and an optional maintenance activity. *Journal of Scheduling*, *26*(3), 251–266.

Wang, T., Baldacci, R., Lim, A., Hu, Q. (2018). A branch-and-price algorithm for scheduling of deteriorating jobs and flexible periodic maintenance on a single machine. *European Journal of Operational Research*, *271*(3), 826–838.

Yu, X., Zhang, Y., Steiner, G. (2014). Single-machine scheduling with periodic maintenance to minimize makespan revisited. *Journal of Scheduling*, *17*(3), 263–270.

Zhang, X., Wu, W.H., Lin, W.C., Wu, C.C. (2018). Machine scheduling problems under deteriorating effects and deteriorating rate-modifying activities. *Journal of the operational research society*, *69*(3), 439–448.

Zhong, X., Ou, J., Wang, G. (2014). Order acceptance and scheduling with machine availability constraints. *European journal of operational research*, *232*(3), 435–441.

Zhu, Z., Zheng, F., Chu, C. (2017). Multitasking scheduling problems with a rate-modifying activity. *International Journal of Production Research*, *55*(1), 296–312.

Appendix

This appendix contains the reprinted versions of the four articles underlying this cumulative dissertation.

A. Article I – State-dependent machine availability in scheduling

Title: State-dependent machine availability in scheduling
Authors: Arter, A., Briskorn, D., Hipp, A., and Jaehn, F.
Journal: International Transactions in Operational Research
Status: Published
Citation: Arter, A., Briskorn, D., Hipp, A., Jaehn, F. (2026b). State-dependent machine availability in scheduling. *International Transactions in Operational Research*, *33*(6), 3769–3799. https://doi.org/10.1111/itor.70183

The following pages present a reprinted version of the above article, reformatted to match the layout of this dissertation.

Copyright:
© 2026 International Federation of Operational Research Societies. Published by John Wiley & Sons Ltd. Content reused with permission from John Wiley & Sons (RightsLink License No. 6306450977089).

State-dependent machine availability in scheduling

Antonia Arter[1], Dirk Briskorn[1], Andreas Hipp[2], Florian Jaehn[2]

[1]: University of Wuppertal, Schumpeter School of Business and Economics

[2]: Helmut Schmidt University, Institute of Business Administration

This research has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 452470135.

Abstract

This paper addresses the field of scheduling problems with machine availability constraints. We give a systematic overview of models that cover any type of state-dependent machine availability restriction caused by jobs or machines. This type of availability constraint has received more attention in recent Operations Research studies because it can be found in various real-world settings. In these settings, the machine state is defined not only by the machine's continuous utilization time but also by other aspects related to machine availability. For instance, the job position in the production sequence on a given machine can have an effect on processing times, deterioration levels of machines, or the length of maintenance operations. Since, to the best of our knowledge, there is no existing systematic survey, we outline the characteristics and demarcate this type of machine availability from the mass of adjacent research. Furthermore, we present and categorize relevant literature dealing with each type of state-dependent machine availability and discuss the relationships between these categories.

Keywords: machine deterioration · maintenance scheduling · runtime complexity in scheduling

1. Introduction

In machine scheduling, a large number of research papers assume continuous machine availability and constant machine performance. However, in real-world settings, almost every machine is subject to some type of wear and tear. Therefore, in the late 1980s, the Operations Research community began to deal with machine scheduling models that account for such effects (Sanlaville and Schmidt (1998)). Typical works in this research

field, such as the ones of Low et al. (2010), Costa et al. (2016), Błażewicz et al. (2003), or Perez-Gonzalez and Framinan (2018), consider models in which machines are characterized by given time points or intervals at which they wear out. Studies of models with arbitrarily selectable time windows to fix the machine's wear and tear are presented by Kubzin and Strusevich (2006) or Rustogi and Strusevich (2012a), for example. What these models have in common is that the availability or performance (or both) of machines is determined by the machines' states. For example, the machine state determines whether a job can be started or processed, and if so, under what constraints. In turn, the processing of jobs might affect the machine state. Designated operations, which we will refer to as maintenance activities (MAs), might improve the conditions for processing jobs in the future, for example, prolong the machine's availability or speed up the machine. We will say that the machine state is maintained. Processing the remaining jobs, which are not an MA, is the genuine purpose of the machines. Processing a job now may worsen the conditions for processing jobs in the future, such as reducing the remaining operating time or slowing down the machine. We will say that the machine state is deteriorated.

Not only MAs and jobs can alter the machine state, but time itself might change it, for example, due to aging of components or operating supplies. If the machine state depends on time, then the state at a certain time is typically a function of the current time in relation to a reference point in time. The reference point in time might be exogenous. For instance, the state might monotonically deteriorate with time or MAs might be predetermined and fixed in time, and the state monotonically deteriorates between two consecutive MAs. Contrarily, the time reference point might be determined as part of the schedule. Again, the state might monotonically deteriorate between two consecutive MAs, but the MAs are to be scheduled as part of the solution (as opposed to being exogenously given). If time is the only impact factor on the state, then it does not depend on operations of the machine between MAs. That is, deterioration is experienced to the same extent during the processing of a job as during a period of idle time (of the same length).

The majority of existing work, in particular in older papers, deals with machines' states depending exclusively on time. In fact, there are a number of surveys. Sanlaville and Schmidt (1998) and Schmidt (2000) provide the first surveys on machine availability constraints in scheduling. Both papers are restricted to models in which machine restrictions are exclusively time-based, that is, MAs are performed at given workload-independent time intervals. Ma et al. (2010) and Strusevich et al. (2017) focus on preventive maintenance scheduling and breakdowns, again, considering only time related restrictions. Concentrating on the complexity of scheduling models, Błażewicz et al. (2019) do not mainly focus on models with machine deterioration, but study, inter alia, models in which machine availability is restricted because of rolling wave planning or job prioritization. Further, resource- and time-dependent processing times are discussed. Another survey is provided by Gawiejnowicz (2020), which covers time-dependent scheduling problems, in-

cluding their history in research and practice. This work is extended by Gawiejnowicz (2026), delivering a methodological overview on problems with time-dependent processing times. He gives a basic overview of the field of scheduling and focuses on the applied solution and proof methods, that is, techniques are compared that are used in scheduling with time-dependent processing times, with techniques in classical scheduling. Ahmad and Kamaruddin (2012) focus their survey on real-world cases for scheduling models with time-dependent constraints. The recently published paper of Geurtsen et al. (2023) is the closest work regarding the consideration of state-dependent machine availability. They consider machine deterioration and maintenance planning, where the specific nature of jobs' processing times are position-based. But in fact, these properties are only mentioned as some of many, without explicitly addressing them. In general, the authors also assume that the machine state and MAs depend on time, not on workload or factors other than time. The focus of their survey is to merge work on resource, maintenance, and production planning with scheduling. Finally, Pei et al. (2023) provide another survey that covers a related problem field by focusing on learning and deterioration effects on processing times of jobs in machine scheduling. Although learning effects are not part of the scope of our paper, Pei et al. (2023) also study deterioration effects other than time, which are covered in the following.

In conclusion, all of these surveys have in common that they primarily outline models dealing with time-dependent machine availability constraints and do not cover the density of non-time based restricted models as is done in this work. Only Geurtsen et al. (2023) and Pei et al. (2023) mention effects on the machine state other than time, but do not treat them as factors that are potentially independent of time-dependent deterioration characteristics.

Our focus, however, is on models where the machines' states are deteriorated by processing jobs or are otherwise influenced by the schedule (possibly in addition to time). Note that in schedules without idle time, states that depend exclusively on time can be seen as states that are continuously deteriorated while processing jobs, so there is considerable overlap. Nevertheless, as long as idle time is feasible, the corresponding models formally differ. However, machines' states that are modified by jobs and MAs, and thus depend on the schedule itself, are mostly found in rather recent work. The purpose of the paper at hand is to categorize and summarize the existing work in this field and to propose future research avenues.

Since we can expect machine states in settings with multiple machines to be independent of each other, we restrict ourselves to a single machine when describing machine states in Section 2. However, our literature survey thereafter also covers models with multiple machines.

In addition to interesting theoretical results, it goes without saying that maintenance scheduling is a research field that has a significant impact on practice. Maintenance is of

fundamental importance at almost every stage of the production process. Thereby, this importance extends from the establishment of suitable maintenance service networks to the detailed planning of maintenance schedules for machinery or individual machine parts. The findings of research taking maintenance into account have already found application in numerous sectors of the economy. Applications include rail track maintenance (Miwa and Oyama (2018)), agriculture (Zhang et al. (2026)), crude oil scheduling (Evazabadian et al. (2019)), and electricity generators (Chattopadhyay (2001)).

The structure of this paper is as follows. In Section 2, we introduce relevant features of machine scheduling models where the schedule impacts the machines' states. We thereby introduce a new notation for the well-known classification scheme of Graham et al. (1979) to add the features. Using this scheme, in Sections 3 and 4, we summarize and categorize existing work by concentrating on properties and runtime complexity of the models. Before giving a conclusion of the scope presented in this paper in Section 6, we look at possible future research in Section 5 that might fill theoretical or real-world oriented gaps in the field of state-dependent machine availability scheduling.

2. Maintenance in machine scheduling

In the following, we detail the fundamental features of machine scheduling problems subject to maintenance. Here we distinguish between how the state of a machine is represented (Section 2.1), how this state is deteriorated and maintained (Section 2.2 and Section 2.3), and how it affects the machine's performance (Section 2.4).

In order to handily represent the features of machine scheduling models with MAs, we first introduce a classification scheme. Similar schemes have been introduced by Graham et al. (1979) and Kendall (1953) for machine scheduling and queuing systems, respectively. In accordance with the structure of this section, we represent features by a four-field tuple $[\delta, \epsilon, \zeta, \eta]$ in our classification scheme. This notation allows full integration into the notation of Graham et al. (1979), so that $\alpha|\beta, [\delta, \epsilon, \zeta, \eta]|\gamma$ specifies a machine scheduling model with machine environment according to α, objective according to γ, maintenance features according to $[\delta, \epsilon, \zeta, \eta]$, and further side constraints according to β. While an overview of common values of α, β, and γ is given in Table 1, the following sections introduce the possible values of $[\delta, \epsilon, \zeta, \eta]$.

Field	Value	Meaning
α	1	single machine
	Fm	flow shop with m machines
	Pm	m identical parallel machines
	Rm	m unrelated parallel machines
β	$d_j = d$	equal due dates
	$p_j = p$	equal processing times
	$pmtn$	preemption
	$prec$	precedence constraints
	r_j	release dates
γ	$C_{\max}$	maximum completion time
	$L_{\max}$	maximum lateness
	$\sum C_j$	total completion time
	$\sum E_j$	total earliness
	$\sum T_j$	total tardiness
	$\sum U_j$	number of tardy jobs
	$\sum w_j C_j$	total weighted completion time
Indices		
	i	machine
	j	job
	n	number of jobs
	r	position of a job on assigned machine/in schedule

Table 1. Common indices and values of α, β, and γ in the Graham notation

2.1. Machine state

A machine state reflects all relevant information about the condition a machine is in. In machine scheduling models, the state might include whether the machine is currently operational and how long it can operate before maintenance becomes necessary. Obviously, in real-world settings a machine state can often be accurately described only by the states of the machine's components, each of which may be difficult to quantify. However, to the best of our knowledge, within the simplifying scope of machine scheduling models, the most general perception of a machine state in the literature is reflected by a single numeric value s. Higher s then reflects a better condition the machine is in. Notably, such a state value is often not explicitly introduced in the literature but most settings at hand can be captured by considering such a state value.

There might be an upper bound $\overline{s}$ or a lower bound $\underline{s}$ (or both) for a machine state, reflecting whether the machine is in perfect shape or on the verge of breaking down. If the actual state is equal to the upper bound, maintenance operations cannot improve the state any further. If the actual state is equal to the lower bound, the machine will break down if any further deterioration is caused before maintenance is conducted.

Models in the literature differ in their instantiations of the lower bound and the upper bound. We reflect this by having $\delta = [\underline{s}, \overline{s}]$ as the first field in our classification scheme.

Note that $\underline{s}$ and $\overline{s}$ might take specific numerical values or $-\infty$ and ∞, respectively, or might not be specified. For example, $\delta = [0, 5]$, $\delta = [0, \infty]$, and $\delta = [\underline{s}, \overline{s}]$ describe settings where the state is at least 0 and at most 5 in each instance, at least 0 (but not bounded from above) for each instance, and specified by the instance itself, respectively. Since a finite lower bound can be chosen arbitrarily (by adjusting the upper bound accordingly), we restrict ourselves to considering only two specific values for the lower bound in the following, namely 0 and $-\infty$.

2.2. Deterioration

Regarding models where the machine state depends on the schedule, we focus on models in the literature where jobs cause deterioration. Sticking to a numeric value representing the machine state as sketched in Section 2.1, this would be decreased by the processing of jobs. Thereby, often the machine state can be assumed to deteriorate uniformly while a job is being processed. The extent to which the state is deteriorated may or may not depend on the job. We find three common manifestations in the literature. First, the deterioration might be *fixed*, that is, processing a job deteriorates the state by a constant ω. Second, the deterioration might be *linear in the job's processing time*, that is, processing a job deteriorates the state by a constant ω^p per time unit of processing time, such that the deterioration caused by job j with processing time p_j is $p_j \cdot \omega^p$. Finally, the deterioration might be *individual* and not (necessarily) correlated with any other property of the job. In this case we denote the level of deterioration ω_j. The last case can clearly be seen as a generalization of the first two.

We reflect these three manifestations by having $\epsilon = \omega$, $\epsilon = \omega^p$, and $\epsilon = \omega_j$, respectively, as the second field in our classification scheme.

All of the proposed deterioration manifestations can be found in real-world applications dealing with deteriorating machines or systems. For example, a constant deterioration $\epsilon = \omega$ per job might go along with deterioration that occurs automatically with the starting of a job and depends only to a negligible extent on the job itself. Such settings can be exemplarily found considering the wear of machine parts in aircraft which are exposed to high wear during takeoff and landing, but the deterioration during the flight itself is negligible. Further, deterioration correlating with the processing time of jobs, $\epsilon = \omega^p$, can be found in machine settings with deterioration according to the lifetime and utilization of the machine like if the machine is exposed to continuous friction. The manifestation $\epsilon = \omega_j$ comprises all settings with job-dependent deterioration due to different job properties.

We do not consider any scheduling-independent machine characteristics that could lead to one of the ϵ-values, that is, the type of a machine itself has no direct influence on the

extent of deterioration. ϵ is always related to job processing. Nevertheless, this case can be displayed indirectly by varying the ϵ if a set of machines is given. For instance, for one machine it might hold $\epsilon = \omega$, and for another one $\epsilon = \omega_j$.

2.3. Maintenance

Since numbers represent states as outlined in Section 2.1, these numbers are increased by MAs. In the literature we find different assumptions about the duration of MAs and their impact on the state.

First, we can distinguish between cases where durations of MAs are independent of the (current) machine state and those where they are not. We say that the MAs' durations are independent if *the duration of an MA neither depends on the state the machine is in when the MA starts*, nor *on the state when the MA is completed.* Vice versa, the duration of an MA itself does not influence how the machine's state changes due to the MA, that is, the state of the machine after performing MAs is predefined.

The other way around, the machine state can influence the MA's duration. We say both the state and the MA are dependent. Then, the machine state when starting the MA, or when completing it (or possibly both), determines the duration of an MA. As a result, the duration of MAs is not fixed in advance. Alternatively, for a given machine state when starting the MA, the MA's duration determines the state the machine is left in.

Second, MAs might be restricted to *bring the machine into perfect shape.* Note that in this case, the duration of an MA is either constant or determined by the state the machine is in when the MA starts. Alternatively, in a more flexible setting, the *target state* to be achieved by conducting an MA may deviate from the maximum state.

We reflect the four corresponding manifestations by having $\zeta = (dep, full)$ with $dep \in \{yes, no\}$ describing whether or not durations of MAs and their impacts on the state are dependent or not. The parameter $full \in \{yes, no\}$ describes whether an MA necessarily brings the machine in perfect shape (yes) or not (no). Note that $full = no$ allows to bring the machine to a perfect state, but it is not required.

Note that all four manifestations are meaningful. Independent of $full$, $dep = no$ might be found in settings where MAs require a considerable repair or service time (overshadowing any variable time component), and $dep = yes$ is meaningful in settings where the variable time component has significant impact. On the other hand, $full = yes$ might be motivated by a substantial organizational burden of MAs, such that the machine state shall be maximized as a matter of policy whenever an MA is conducted, while $full = no$ allows flexibility, for example, when splitting an MA into multiple operations does not involve significant additional cost.

In the following, we use y and n for *yes* and *no* in the Graham notation.

2.4. Performance

The state a machine is in might influence the machine's performance. In this case, it is typically assumed that *deterioration slows down the machine*, and that the processing time of a job is a nonincreasing function of the state at the job's start time. Alternatively, *processing time of each job does not depend on the state of the machine*, which can be seen as a special case of the former.

We represent these two settings by $\eta = p_j$ and $\eta = p_{js}$ where p_j stands for processing times depending only on jobs and p_{js} represents that a job's processing time p_j may additionally depend on the state s at its start time.

In practice, the deterioration of the machine may also affect product quality (Xu et al. (2022)), which is not considered in this work.

Table 2 shows all expressions for the machine state (δ), the deterioration of a machine (ϵ), the maintenance (ζ), and the influence on the performance of the maintenance and machine (η).

Field	Value	Meaning
δ	$\underline{s}$	lower bound of machine state, $\underline{s} \in \{-\infty, 0\}$
	$\overline{s}$	upper bound of machine state, $\overline{s} \in [0, \infty]$
ϵ	ω	constant deterioration per job, independent of the job characteristics
	ω^p	deterioration correlating on the job's processing time
	ω_j	individual deterioration that depends on each job
ζ	dep	information about dependencies between the MA and the machine state, $dep \in \{yes, no\}$
	$full$	information if the machine is completely restored after the MA or not, $full \in \{yes, no\}$
	$-$	no type of MAs is considered
η	p_j	processing time p_j only depends on the characteristics of job j
	p_{js}	processing time p_j depends on the characteristics of job j and state s of machine

Table 2. Expressions of δ, ϵ, ζ, and η in the β-field of the Graham notation

In addition to the commonly used values in the literature for expressing scheduling problems in Graham notation, as summarized in Table 1, and our proposed four-field notation capturing general maintenance features within the β-field, Table 3 provides an overview of additional notation used in the upcoming sections to introduce scheduling problems from the literature involving maintenance. It should be noted that Table 3 is intended to serve as a reference for the reviewed literature, and we therefore refrain from providing further elaboration at this point. In addition, compared to the original sources, we have partly modified the used notation where necessary in order to ensure consistent notation

Field	Value	Meaning
β	B	batch
	d^{MA}	duration of an MA
	d_{il}^{MA}	duration of MA l on machine i
	$d^{MA}(x)$	duration of an MA depending on x
	$f(x)$	function depending on x
	fp	subset of jobs must be scheduled in fixed positions
	$fp-q$	subset of q jobs must be scheduled in fixed positions
	inv	inventory constrained scheduling
	m_i	specific characteristics of machine i
	N^{MA}	number of MAs
	$resource$	resource allocation restrictions
	S^{MA}	start time of an MA
	S_{psd}	past-sequence-related setup times, dependent on already scheduled jobs
	SP	series-parallel precedence constraints
	u_{ij}	resource allocation of job j on machine i
	v_{ij}	costs per time unit to process job j on machine i
γ	D	common due window for all jobs
	$TADC^i$	total absolute differences in completion times on machine i
	$TADW^i$	total absolute differences in waiting times on machine i
	TC^i	total completion time on machine i
	TL	total load as the sum of maximum completion times over all machines
	TW^i	total waiting time on machine i

Table 3. Specific entries in the β- and γ-field of the Graham notation applied in Sections 3 and 4

throughout this paper.

We categorize the scheduling problems found in the literature with regard to the maintenance features we have identified. Accordingly, we collate the literature relevant to the scope of this survey, providing complexity results, and present these findings in tables. Nevertheless, there is additional literature that also provides important insights regarding problems with state-dependent machine availability (see, e.g., Section 4.1). These papers either do not focus on the problems' complexity, though, or are related but finally too far away from our topic regarding its characteristics. They are discussed in the text, but are not included in the tables. Additionally, we introduce an example case in Section 3 to illustrate what machine environments involving maintenance might look like in practice. This example is adapted throughout the subsequent subsections of Sections 3 and 4 to illustrate the specific variants of maintenance features considered in the literature. Note that our example is purely illustrative and not based on any source, as the authors of the cited papers typically address abstract and stylized problems.

3. Processing times independent of the machine state

In this section, we address problem settings where the processing time does not depend on the machines state ($\eta = p_j$). Within this class of problems we distinguish between settings where all jobs deteriorate a machine equally ($\epsilon = \omega$) in Section 3.1, where deteriorations

are linear in the jobs' processing times ($\epsilon = \omega^p$) in Section 3.2, and where deteriorations are individual ($\epsilon = \omega_j$) in Section 3.3.

3.1. Fixed machine deterioration

In the paper by Briskorn et al. (2010), the authors consider an inventory constraint of a homogeneous good on classical scheduling problems. This means that each job, in addition to its classical characteristics, either delivers to or removes from inventory. The inventory level is bounded from below but not from above. If we consider the inventory level as the state of the machine, and the jobs that deliver to the inventory as maintenance, there is an obvious connection between the two problem settings. However, when comparing the results, one has to keep in mind that, at least in general, jobs that add to the inventory contribute to the value of the objective function, while an MA in our case does not. Nevertheless, some results are transferable, for example, by choosing weight $w_j = 0$ or due date $d_j = \infty$ for a job j representing maintenance. These transferable results are summarized in Table 4, together with the following results of this section.

The first paper, to the best of our knowledge, to explicitly consider the case of $\epsilon = \omega$ is by Drozdowski et al. (2017). They consider various single machine problems, always with a predetermined maximum machine state, and fixed-duration maintenance that fully recovers the machine. In particular, they focus on preemption, which causes a faster deterioration of the machine, since also every part of a job occurring as a result of preemptions causes the same level of deterioration on the machine as a job processed without any preemptions. Thus, problems that are easy to solve when preemption is not allowed become hard. Following this line of research, Hipp (2023) and Hipp and Jaehn (2024) add complexity results for the case of unique processing times with the number of tardy jobs and weighted sum of completion as the objective, respectively.

The recently published paper by Rudek (2026) is the sole study in this subsection that addresses a problem dealing with parallel machines, whereby the processing time of jobs depends on the state of the machine. In the paper, a machine's state is represented by a numerical value, with the lowest value representing the optimal state of the machine. The state increases by 1 with every job processed and can be restored by maintenance. This can easily be transferred to the case of $\delta = [0, \bar{s}]$.

There is a very small stream in the literature (Calinescu et al. (2017), Jaehn (2024)) that considers scheduling problems where the position of certain jobs in the sequence is fixed in advance. These types of jobs are motivated, among other things, by MAs that must be scheduled as late as possible. If we choose an appropriate deterioration rate and appropriate bounds on the machine's state, we can enforce that maintenance operations

are in certain positions in the schedule. Thus, they are no longer part of the actual scheduling process. However, an important difference from the other problems listed here is the fact that these fixed-position jobs contribute to the respective objective function. These problems can be characterized using our notation framework, but this requires a rather complex objective function. Therefore, in Table 4, we list these problems using the entry fp, which represents the fixed-position constraint of certain jobs.

As mentioned, let us introduce a running example case, a wood factory, that will be modified throughout the paper to reflect specific maintenance features. In this factory, we have a single cutting machine that can cut beams or boards of different sizes. To cut the wood into pieces, the machine has a blade that deteriorates by usage. To illustrate how *fixed machine deterioration*, in combination with *independent processing times* might look in this machine environment, we assume that the duration of the cutting process only depends on the characteristics of the board (p_j), and the deterioration of the blade is constant for each job (ω). By assuming that MAs are executed by exchanging the blade, which always consumes the same amount of time and leads to a total restoration of the machine's performance, we are close to the problems described by Drozdowski et al. (2017). If the MAs are predetermined in their position in the sequence, since the exchange of the blade might be subject to mandatory requirements, this case corresponds to the problems considered in Calinescu et al. (2017) and Jaehn (2024).

Problem	**Complexity**	**Notes**	**Source**
$1\|inv,[[0,\infty],\omega,(n,n),p_j]\,\|\sum C_j$	$O(n\log n)$		Briskorn et al. (2010), Theorem 3
$1\|p_j=p,inv,[[0,\infty],\omega,(n,n),p_j]\,\|\sum w_jC_j$	$O(n\log n)$		Briskorn et al. (2010), Theorem 3
$1\|inv,[[0,\infty],\omega,(n,n),p_j]\,\|\sum w_jC_j$	open		Briskorn et al. (2010), Table 1
$1\|d_j=d,inv,[[0,\infty],\omega,(n,n),p_j]\,\|\sum U_j$	$O(n^6)$		Briskorn et al. (2010), Theorem 9
$1\|p_j=p,inv,[[0,\infty],\omega,(n,n),p_j]\,\|\sum U_j$	$O(n^3)$		Briskorn et al. (2010), Theorem 8
$1\|inv,[[0,\infty],\omega,(n,n),p_j]\,\|\sum U_j$	open		Briskorn et al. (2010), Table 3
$1\|pmtn,[[0,\overline{s}],\omega,(n,y),p_j]\,\|C_{\max}$	$O(n)$		Drozdowski et al. (2017), Lemma 1
$1\|r_j,[[0,\overline{s}],\omega,(n,y),p_j]\,\|C_{\max}$	$O(n^2)$	for small $\overline{s}$, runtime is $O(n\log n)$	Drozdowski et al. (2017), Lemma 2
$1\|r_j,pmtn,[[0,\overline{s}],\omega,(n,y),p_j]\,\|C_{\max}$	$O(n^2)$	for small $\overline{s}$, runtime is $O(n\log n)$	Drozdowski et al. (2017), Lemma 3
$1\|\,[[0,\overline{s}],\omega,(n,y),p_j]\,\|L_{\max}$	$O(n\log n)$		Drozdowski et al. (2017), Lemma 4

Problem	Complexity	Notes	Source
$1\|pmtn, [[0,\overline{s}],\omega,(n,y),p_j]\|L_{\max}$	$O(n\log n)$		Drozdowski et al. (2017), Lemma 5
$1\|r_j, pmtn, [[0,\overline{s}],\omega,(n,y),p_j]\|L_{\max}$	strongly $\mathcal{NP}$-hard		Drozdowski et al. (2017), Lemma 6
$1\|p_j=p, r_j, [[0,\overline{s}],\omega,(n,y),p_j]\|\sum U_j$	$O(n^8)$		Hipp (2023)
$Pm\|[[0,\overline{s}],\omega,(n,y),p_{js}]\|L_{\max}$	strongly $\mathcal{NP}$-hard		Rudek (2026)
$1\|[[0,\overline{s}],\omega,(n,y),p_j]\|\sum w_jC_j$	$O(n\log n)$	MAs as late as possible, Smith's rule	Hipp and Jaehn (2024)
$1\|p_j=p, r_j, [[0,\overline{s}],\omega,(n,y),p_j]\|\sum w_jC_j$	$O(n^{18})$		Hipp and Jaehn (2024)
$1\|fp\|L_{\max}$	$O(n^3)$		Jaehn (2024), Theorem 2
$1\|fp\text{-}1\|\sum U_j$	$O(n^5)$		Jaehn (2024), Theorem 5
$1\|fp\|\sum U_j$	open		Jaehn (2024), Table 1
$1\|fp\|\sum C_j$	$O(n\log n)$		Jaehn (2024), Theorem 6
$1\|fp\text{-}1\|\sum w_jC_j$	FPTAS		Calinescu et al. (2017), Lemma 7

Table 4. Complexity status of problems with given processing times and fixed deterioration

3.2. Deterioration of machine state linear in processing time

To the best of our knowledge, only a few papers study models with constant processing times regarding the machine state ($\eta = p_j$) that cause machine deterioration according to ω^p. In these papers, whose results are listed in Table 5, ω^p is usually referred to as *sum-of-processing-times-based* or *workload-dependent* processing times or deterioration. Furthermore, they have in common that MAs depend on the machine state ($dep = yes$) and that maintenance fully restores the machine ($full = yes$).

Xu et al. (2015) consider a very special case of maintenance scheduling where exactly one MA is assigned to the machine. This MA is fixed regarding its start time, but its duration d^{MA} depends on the workload x before the maintenance. $d^{MA}(x)$ is a nondecreasing function of the machine's workload that determines the length of the MA. The authors consider two assumptions for the maintenance duration function, $(d^{MA})'(x) \geq 1$ and $0 < (d^{MA})'(x) < 1$, and present an exact algorithm and polynomial-time approximation scheme, respectively. In the course of the approach, a so-called (k,k)-exchange is applied, where k jobs of the job subset before the MA are exchanged with k jobs of the subset after the MA.

Almost the same problem as Xu et al. (2015), regarding the machine environment and maintenance characteristics, is studied by Luo and Liu (2017), but for minimizing total weighted flow time. Here, an arbitrary maintenance duration function $d^{MA}(x)$ is assumed instead of the given functions in Xu et al. (2015). They present a $(2+\epsilon)$-approximation and fully polynomial-time approximation scheme whose runtime depends, inter alia, on the completion time of binary search (see, e.g., Cormen et al. (2022)). In binary search, a sorted sequence is halved, testing for a given value. This procedure is repeated until the value is met.

A more general approach to workload-dependent maintenance scheduling is considered by Xu and Xu (2015) assuming job preemption and an arbitrary number of MAs. They formulate a pseudo-polynomial algorithm that is directly affected by the minimum workload interval of the machine (the workload limits t_{min} and t_{max} are given).

Lee and Kim (2017) deal with a two machine flow shop scheduling problem where only the machine in the first stage deteriorates and requires maintenance. Since the two-machine flow shop minimizing total tardiness is already strongly $\mathcal{NP}$-hard (see Koulamas (1994)), the authors analyze the problem to formulate dominance properties and a branch-and-bound algorithm including a modified version of the heuristic of Nawaz et al. (1983) to determine upper bounds for the algorithm.

In contrast to the aforementioned papers, the work of Hadidi et al. (2011) also includes deciding when to schedule MAs in order to prevent machine breakdowns. The objective is to minimize the sum of costs of holding (HC), repair (MRP), and preventive maintenance (PMC), that is,

$$1|[[0,\bar{s}],\omega^p,(y,y),p_j]|HC+MRC+PMC.$$

Even though they do not analyze the problem regarding its complexity, and therefore we have decided not to list this work in Table 5, an interesting insight into the practical relevance of such models is given. In another work, Hadidi et al. (2015) modify and extend this approach by, inter alia, looking at other decision policies for their integrated approach.

In the case of the cutting machine introduced in Section 3.1, we assume now that the processing time of a job depends on the number and length of cuts required to process the wooden boards. In consequence, it is conceivable that there is a direct, proportional impact of the processing time on the level of deterioration of the blade (ω^p), since longer processing times imply more elaborate cutting actions. Transferring the problems studied by Xu et al. (2015) and Luo and Liu (2017), the blade cannot be exchanged, but ground to resharpen it. This can only be done once throughout the entire planning horizon. How much the blade has to be ground to be restored, and as a result how long the MA lasts, depends on the workload before the MA.

Problem	Complexity	Notes	Source
$1\|N^{MA}=1, S^{MA}=\text{fixed}, (d^{MA})'(x)\geq 1, [[0,\bar{s}],\omega^p,(y,y),p_j]\|\sum C_j$	$\mathcal{O}(n^2)$	x: machine workload	Xu et al. (2015)
$1\|N^{MA}=1, S^{MA}=\text{fixed}, 0<(d^{MA})'(x)<1, [[0,\bar{s}],\omega^p,(y,y),p_j]\|\sum C_j$	PTAS, $\mathcal{O}(n^{2k+1})$	k: algorithm parameter	Xu et al. (2015)
$1\|pmtn,[[0,\bar{s}],\omega^p,(y,y),p_j]\|C_{max}$	$\mathcal{O}(\lceil\sum p_j/t_{min}\rceil)$	t_{min}: machine's minimum working interval	Xu and Xu (2015)
$F2\|m_1=[[0,\bar{s}],\omega^p,(y,y),p_j], m_2=\emptyset\|\sum T_j$	solution method: Branch-and-Bound		Lee and Kim (2017)
$1\|N^{MA}=1, S^{MA}=\text{fixed}, [[0,\bar{s}],\omega^p,(y,y),p_j]\|\sum w_jC_j$	$(2+\epsilon)$-approx., $\mathcal{O}(n\log f(S^{MA})/\epsilon^2 + T_f\log f(S^{MA})\log S^{MA}/\epsilon)$	T_f: runtime factor of binary search with nondecreasing function $f(\cdot)$ for the MA duration	Luo and Liu (2017), Theorem 1
$1\|N^{MA}=1, S^{MA}=\text{fixed}, [[0,\bar{s}],\omega^p,(y,y),p_j]\|\sum w_jC_j$	FPTAS, $\mathcal{O}(n^4\log f(S^{MA})/\epsilon^3 + T_f\log f(S^{MA})\log S^{MA}/\epsilon)$	T_f: runtime factor of binary search with nondecreasing function $f(\cdot)$ for the MA duration	Luo and Liu (2017), Theorem 2

Table 5. Complexity status of problems with given processing times and linear deterioration

3.3. Job individual machine deterioration

As already noted in Section 3.1, scheduling with inventory constraints bears strong parallels to the concepts of scheduling with MAs. We now want to present the remaining results provided in Briskorn et al. (2010) for problem settings with job individual machine deterioration. To repeat, the authors give a general definition of a scheduling problem with inventory constraints and due dates. Thereby the processing times of jobs do not depend on the machine state and arbitrary inventory modifications are allowed, indicating arbitrary deterioration and partial MAs in terms of maintenance scheduling. A given set of MAs with specified properties has to be scheduled. The authors consider the objective functions $\sum w_jC_j$, $L_{\max}$, and $\sum U_j$ and show that the general versions of the considered problems are all (strongly) $\mathcal{NP}$-hard. Hence, they consider problem variants assuming that, using our terminology, either all jobs or all MAs have certain characteristics in common. Note that to strengthen the parallels between inventory scheduling and maintenance scheduling one could prevent that maintenance is considered directly in the objective function by assuming constant weights $w_j = 0$ for jobs with positive inventory modifications.

The results for the general problem and some polynomially solvable problem variants maintaining $\epsilon = \omega_j$ can be found in Table 6. In Briskorn et al. (2013), the authors present exact algorithms to solve the general problem for the objective $\sum w_j C_j$. The general problem of minimizing maximum lateness $L_{\max}$ can be solved by a branch-and-bound algorithm as presented in Briskorn and Leung (2013). In Morsy and Pesch (2015), the authors present a 2-approximation algorithm for a problem variant of the general problem with objective $\sum w_j C_j$.

Different from the problem definition in Briskorn et al. (2010), in Davari et al. (2020), an inventory scheduling problem is formulated to minimize the makespan $C_{\max}$ while the inventory level may not exceed a maximum level and all jobs have release dates. Interpreted as a maintenance scheduling problem, a set of mandatory MAs, including their properties, is given and needs to be scheduled simultaneously with the jobs, where the maximum maintenance capacity is limited. The problem remains $\mathcal{NP}$-hard.

The paper by Bock et al. (2012) discusses the complexity of several scheduling problems that involve maintenance and individual deterioration. Thus, each job deteriorates the machine by a job-dependent amount ω_j, and MAs need to be conducted to ensure the machine's processing. Thereby MAs improve the machine state either to perfect shape or partially. The processing times of jobs do not depend on the machine state. The authors consider different objective functions and distinguish between the full maintenance case and a flexible setting where the machine is restored to a target state. The complexity results can again be found in Table 6. Grigoriu and Briskorn (2017) provide a 5/4-approximation algorithm for the single machine problem with full maintenance and makespan minimization introduced by Bock et al. (2012). Tian et al. (2019) extend the problem from a single machine to parallel machines with partial MAs. They propose a $(2\rho_{\max} + 3)$-approximation algorithm for $\rho_{\max} = \max\{\omega_j / p_j\}$ to minimize makespan.

Briskorn et al. (2024) consider a variant of the problem introduced in Bock et al. (2012) by limiting the problem to fixed job sequences and considering job individual machine deterioration which is subject to a discrete probability distribution. MAs can be scheduled at the beginning of the planning horizon, between two consecutive jobs, or after the completion of the last job in the sequence. A solution is a decision policy providing the decision when to schedule MAs. Thereby, either the positions in the sequence for maintenance are completely determined in advance independent of the realized deterioration, or the policy provides decisions about maintenance depending on the actual machine state. The authors show that both problem variants are $\mathcal{NP}$-hard and provide solution approaches using dynamic programming with pseudo-polynomial runtime.

In the exemplary wood factory, the deterioration of the blade is now individual for each job (ω_j). There might be different sizes of boards, but also different types of wood which lead to different levels of deterioration of the blade. As a result, each job stands for itself regarding its deterioration effect on the blade. Following the ideas of Briskorn et al. (2010),

the blade can be restored to different target states by grinding it to a certain extent. If the cutting process itself consumes for each job exactly the same amount of time, nevertheless the deterioration is individual, the problem environment coincides with that described by Bock et al. (2012).

Problem	**Complexity**	**Notes**	**Source**
$1\|inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|\, \sum C_j$	strongly $\mathcal{NP}$-hard		Briskorn et al. (2010), Theorem 1
$1\|inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|\, \sum w_j C_j$	strongly $\mathcal{NP}$-hard		Briskorn et al. (2010), Theorem 2
$1\|p_j = p, inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|\, \sum C_j$	$\mathcal{O}(n \log n)$	constant maintenance duration or increment of improvement	Briskorn et al. (2010), Theorem 3
$1\|inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|L_{\max}$	strongly $\mathcal{NP}$-hard		Briskorn et al. (2010), Theorem 4
$1\|d_j = d, inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|L_{\max}$	$\mathcal{O}(n^2(\sum p_j)^5 \sum \|\omega_j\|)$		Briskorn et al. (2010), Theorem 4
$1\|inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|L_{\max}$	$\mathcal{O}(n^2 \log(\sum p_j))$	constant maintenance duration or increment of improvement	Briskorn et al. (2010), Theorem 5
$1\|d_j = d, inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|\, \sum U_j$	$\mathcal{NP}$-hard		Briskorn et al. (2010), Theorem 6
$1\|inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|\, \sum U_j$	$\mathcal{O}(n^2 \sum p_j \sum \|\omega_j\|)$	equal due dates for maintenance and constant maintenance duration or increment of improvement	Briskorn et al. (2010), Theorem 7
$1\|p_j = p, inv, [[0,\infty], \omega_j, (n,n), p_j] \,\|\, \sum U_j$	$\mathcal{O}(n^3)$	equal due dates for maintenance and constant maintenance duration or increment of improvement	Briskorn et al. (2010), Theorem 8
$1\|d_j = d, p_j = p, inv,$ $[[0,\infty], \omega_j, (n,n), p_j] \,\|\, \sum U_j$	$\mathcal{O}(n^6)$	constant maintenance duration	Briskorn et al. (2010), Theorem 9
$1\|r_j, inv, [[0,\bar{s}], \omega_j, (n,n), p_j] \,\|C_{\max}$	strongly $\mathcal{NP}$-hard		Davari et al. (2020), Theorem 1

Problem	Complexity	Notes	Source
$1\|p_j = p, [[0,\bar{s}], \omega_j, (y,y), p_j] \,\|\, \sum C_j$	$\mathcal{NP}$-hard		Bock et al. (2012), Theorem 3
$1\|p_j = p, [[0,\bar{s}], \omega_j, (y,y), p_j] \,\|C_{\max}$	$\mathcal{NP}$-hard		Bock et al. (2012), Theorem 5
$1\|\, [[0,\bar{s}], \omega_j, (y,y), p_j] \,\|L_{\max}$	$\mathcal{O}(n^2(\bar{s})^2 \cdot \log(\sum(p_j + \omega_j)))$		Bock et al. (2012), Theorem 7
$1\|p_j = p, [[0,\bar{s}], \omega_j, (y,y), p_j] \,\|\, \sum U_j$	$\mathcal{NP}$-hard		Bock et al. (2012), Theorem 8
$1\|p_j < p_i \Rightarrow p_j + \omega_j \leq p_i + \omega_i$, $[[0,\bar{s}], \omega_j, (n,y), p_j] \,\|\, \sum C_j$	$\mathcal{O}(n \log n)$		Bock et al. (2012), Theorem 10
$1\|p_j < p_i \Rightarrow p_j + \omega_j \leq p_i + \omega_i$, $[[0,\bar{s}], \omega_j, (n,y), p_j] \,\|\, \sum U_j$	$\mathcal{O}(n \log n)$		Bock et al. (2012), Theorem 12
$1\|d_j = d, [[0,\bar{s}], \omega_j, (n,y), p_j] \,\|\, \sum U_j$	$\mathcal{O}(n \log n)$		Bock et al. (2012), Theorem 13
$1\|\, [[0,\bar{s}], \omega_j, (n,y), p_j] \,\|\, \sum U_j$	$\mathcal{NP}$-hard in the ordinary sense		Bock et al. (2012), Theorem 14

Table 6. Complexity status of problems with given processing times and individual deterioration

4. Processing times depending on the machine state

In this section, we address problem settings where the processing times depend on the machines' states ($\eta = p_{js}$). Within this class of problems we distinguish between settings where all jobs deteriorate a machine equally ($\epsilon = \omega$) in Section 4.1, and where deteriorations are linear in the jobs' processing times ($\epsilon = \omega^p$) in Section 4.2. In Section 4.1, we further differentiate the models with respect to the jobs' processing time characteristics and whether the deterioration is job-dependent or not. We do not explicitly consider the case where deterioration is individual ($\epsilon = \omega_j$), but the operating principles of the models in sections 4.1.1 and 4.1.2 are closely related to the idea of individual deterioration.

4.1. Fixed machine deterioration

The scheduling of jobs whose (deterministic) processing time is not a constant has been considered in various contexts in the literature. A classical scheduling problem considers related parallel machines, that is, machines that differ only in speed (Chekuri and Khanna (2001); Jansen et al. (2020)). Similarly, some models consider that the speed of a single machine can be adjusted, for example, in speed scaling (Albers et al. (2014); Bampis et al.

(2015)). In this context, there is a large body of literature dealing with a rather simple maintenance scheduling model, which has been proposed by Lee and Leon (2001), among others. Thereby, a single MA can potentially be scheduled within the entire planning horizon. It is assumed that the MA decreases the processing times of all subsequent jobs, but obstructs the machine, causing a delay in job processing equal to the duration of the maintenance. This trade-off needs to be evaluated and decisions need to be made regarding both, job sequencing and maintenance scheduling. These effects can be classified in the context of the machine's maintenance level as follows. The machine's maintenance level is binary and only becomes one after the MA has been performed. The processing of jobs has no impact on the level, that is, we have a constant deterioration of $\omega_j = 0$ per job j. Mosheiov and Sidney (2010) extend the problem to MAs whose duration increases with their start time. This assumption is consistent with the understanding of a deteriorating machine over time. All the problems studied are shown to be solvable in polynomial time. Other authors deal with this type of maintenance scheduling problems and include further properties such as random machine breakdowns (Lee and Lin (2001)) resulting in three possible maintenance levels (initial, after MAs and breakdowns), position-dependent MA duration, and due-window assignments (Mor and Mosheiov (2012); Mor and Mosheiov (2015)). The problem is extended to a parallel machine environment in Wang et al. (2011) but remains polynomially solvable for a fixed number of machines m. Yang et al. (2014) state the problem of potential single MAs for parallel machine setting. The processing times still depend on the fact whether an MA was previously scheduled on the machine and are further controllable by job compression. The problem can be solved in $\mathcal{O}(n^{m+3})$.

Motivated by machine deterioration, and thus approaching the scope of this paper, the processing time of a job may depend on the start time of the job (Cheng et al. (2004)). In this section, we consider situations in which the processing time of a job depends on its position in the schedule. In fact, there are only a few papers on this topic. A significant part of the literature concentrates on learning effects, which propose that the processing time of jobs decreases the later they are scheduled (Biskup (1999)). Conversely, another perspective in the literature explores deterioration effects, which propose that job processing time increases as the job position advances due to machine wear. Although learning effects are not in our scope, problems from both settings can be sorted into the model presented in this paper. Thereby, the state of the machine counts the number of jobs processed so far and thus constantly deteriorates per job. Furthermore, the literature is divided according to whether MAs are involved or not. Thereby, the impact of MAs on the machine state depends on the specific problem at hand.

Looking at our exemplary factory of Section 3, the time to cut the wood depends on the state of the machine, here of the blade (p_{js}). By executing MAs, the processing time can be shortened. If only one MA is allowed to be scheduled, this problem is related to the one of Lee and Leon (2001). By introducing the possibility of random machine breakdowns,

we address a related problem setting like the one of Lee and Lin (2001).

In most papers it is assumed that each job has a base processing time p_j which may be modified by its position and therefore depends on the state of the machine. The literature can be categorized more specifically based on whether the position's effect on a job's processing time is depending on the job. The remainder of this section will follow the proposed division.

4.1.1. Position-dependent, job-independent deterioration of machine state

First, we consider problem formulations with position-dependent processing times. Thus, the influence of the position on the processing time is job-independent. The first (to the best of our knowledge) to consider processing times that depend on the number of jobs processed is Gawiejnowicz (1996). He analyzes decreasing machine speed (deterioration), increasing machine speed (learning effect), and arbitrary speed functions. We restrict ourselves to the first. While these deterioration functions are very general (requiring only that processing times increase), Mosheiov (2005) considers a polynomial increase of a job's processing time with increasing positions in the schedule. If p_j is the processing time of job j if it were scheduled first, then its processing time at position r in the schedule is given by $p_j r^a$ for some $a > 0$. Maintenance operations are not taken into account, so according to our notation, the state of the machine never improves during the processing time. He makes use of knowledge from problems with learning functions (Mosheiov (2001)).

The additional consideration of precedence constraints is introduced by Gordon et al. (2008). They analyze not only the polynomial processing times $p_j r^a$ described above (with positive results only for $a = 1$), but also exponential processing times in the form of $p_j \gamma^{r-1}$, where $\gamma > 1$ is a parameter.

Kuo and Yang (2008) do not require, but allow, MAs to speed up processing times. They also introduce a constraint that limits the number of MAs. Since the number of MAs is naturally bounded by n (if no preemption is allowed), any algorithm solving a problem with such a constraint can be used up to n times to solve the corresponding unrestricted problem. Their result (see Table 7) is generalized by Rustogi and Strusevich (2012a). The latter distinguishes between group-dependent and group-independent deterioration, where a group refers to the jobs scheduled between two MAs. However, group-independent deterioration simply refers to the case where a machine is fully recovered after maintenance ($full = yes$). Group-dependent deterioration can then be modeled by a partial recovery of the machine.

Xue et al. (2014) consider general position-dependent processing times that may include learning effects and deterioration of a job within an instance. However, they are also restricted to problems that include only learning effects (not considered here) and deteri-

Problem	Complexity	Notes	Source
$1\vert\,[[0,\overline{s}],\omega,(n,y),p_{js}]\,\vert C_{\max}$	$\mathcal{O}(n\log n)$	for $\overline{s}\geq n$	Gawiejnowicz (1996), Theorem 2
$1\vert\,[[0,\overline{s}],\omega,(n,y),p_{js}]\,\vert C_{\max}$	$\mathcal{O}(n\log n)$	for $\overline{s}<n$	Gawiejnowicz (1996), Theorem 4
$1\vert\,[[-\infty,\infty],\omega,-,p_{js}=p_jr^a]\,\vert C_{\max}$	$\mathcal{O}(n\log n)$	$a>0$, LPT-rule	Mosheiov (2005), p. 885
$1\vert\,[[-\infty,\infty],\omega,-,p_{js}=p_jr^a]\,\vert\sum C_j$	$\mathcal{O}(n^3)$	$a>0$, solution is V-shaped	Mosheiov (2005), p. 884, Theorem 1
$1\vert SP,[[-\infty,\infty],\omega,-,p_{js}=p_jr]\,\vert C_{\max}$	$\mathcal{O}(n\log n)$		Gordon et al. (2008), Theorem 3
$1\vert SP,\left[[-\infty,\infty],\omega,-,p_{js}=p_j\gamma^{r-1}\right]\vert C_{\max}$	$\mathcal{O}(n\log n)$	$\gamma>1$	Gordon et al. (2008), Theorem 6
$1\vert prec,\left[[-\infty,\infty],\omega,-,p_{js}=p_j\gamma^{r-1}\right]\vert\sum C_j$	$\mathcal{O}(n\log n)$	$\gamma\geq 2$	Gordon et al. (2008), Theorem 7
$1\vert\,[[-\infty,\infty],\omega,(n,y),p_{js}=p_jr^a]\,\vert C_{\max}$	$\mathcal{O}(n^2)^*$	$a>0$, see next line for generalization	Kuo and Yang (2008), Theorem 3
$1\vert\,[[-\infty,\infty],\omega,(n,y),p_{js}]\,\vert C_{\max}$	$\mathcal{O}(n^2)$		Rustogi and Strusevich (2012a), p. 795
$1\vert\,[[-\infty,\infty],\omega,(n,n),p_{js}]\,\vert C_{\max}$	$\mathcal{O}(n^2)$		Rustogi and Strusevich (2012a), Theorem 3
$1\vert N^{MA}=\lfloor\frac{n}{k}\rfloor$, $[[0,\overline{s}],\omega,(n,y),p_{js}=p_jr^a]\,\vert$ $\alpha C_{\max}+\beta\sum E_j+\gamma\sum T_j$	$\mathcal{O}(n^2\log n)$	k: number of jobs between two MAs, $a>0$	Xue et al. (2014), Theorem 3

*The underlying paper states a runtime of $O(n\log n)$, but Rustogi and Strusevich (2012a) correct it to $O(n^2)$.

Table 7. Complexity status of problems with state-dependent processing times and fixed deterioration

oration, respectively. Unlike previous approaches, they minimize a linear combination of makespan, sum of earliness, and sum of tardiness.

The works of Montoya-Torres et al. (2021) and Montoya-Torres et al. (2023) are not listed in Table 7 because they do not provide complexity results. However, they present mixed integer programming models (MIPs) and heuristics for multi-machine problems with deterioration. Both papers justify their model by workers who represent the machines and whose productivity decreases due to fatigue or stress. Montoya-Torres et al. (2021) first consider the problem

$$Pm\vert\,[[-\infty,\infty],\omega,-,p_{js}=p_jr^a]\,\vert C_{\max}$$

with $a>0$ and proposes a MIP. Further MIPs are presented for extensions, namely different but related machine (or worker) deterioration factors $a_i>0$ for each machine i,

$$Pm\vert\,[[-\infty,\infty],\omega,-,p_{js}=p_jr^{a_i}]\,\vert C_{\max},$$

and breaks for workers, which corresponds to machine maintenance,

$$Pm|\,[[0,\bar{s}],\omega,[n,n],p_{js}=p_j r^{a_i}]\,|C_{\max}.$$

Montoya-Torres et al. (2023) analyze the problem

$$Pm|\,[[-\infty,\infty],\omega,-,p_{js}=\lceil p_j r^{a}\rceil]\,|T_{\max},$$

which is at least binary $\mathcal{NP}$-hard, since for infinitesimal a $P2||C_{\max}$, which is $\mathcal{NP}$-hard, can be reduced to it.

The cutting machine of Section 4.1 of our example is now specified in the way that the increase of processing times is only position dependent, like, for example, described by Mosheiov (2005). As more jobs the machine cuts, as longer each job lasts because the blade becomes dull. Adopting the approach of Kuo and Yang (2008), the blade deteriorates but not to a level that it cannot process jobs anymore. MAs can be executed to restore the blade, but it is not mandatory.

4.1.2. Position-dependent, job-dependent deterioration of machine state

Next, we focus on approaches where the actual processing times of jobs are influenced not only by the position, but also by job-dependent factors.

First, we consider problems with no MAs involved but with deteriorating machines and collect some complexity results in Table 8.

Again, there is a significant body of research that focuses primarily on learning effects. However, the work of Mosheiov and Sidney (2003) and Mosheiov and Sidney (2005), for example, provides approaches and results that can be used for research that takes into account deterioration effects. Mosheiov and Sarig (2008) address a scheduling problem with completely arbitrary position-dependent processing times that are given numerically for each job and each position, so there are no limitations to learning or deterioration effects, combined with a due window assignment problem. They propose an $\mathcal{O}(n^3)$ algorithm. Yu et al. (2014) provide an overview of the complexity of scheduling problem involving parallel machines that are either identical or unrelated and position-dependent processing times $p_{ijr} = f_{ij}(i,j,r)$ of job j scheduled in the r-th position on machine i. Thereby the authors consider different properties of the function determining the actual processing times f_{ij} depending on the allocation of job j to machine i. The primary objective in their considered scenarios is to reduce the total load, which is the sum of maximum completion times of all machines. The results are listed in Table 8.

In Zhao and Tang (2010), a very general interpretation of position-dependent and job-dependent processing times is given by $p_{js} = p_j r^{\alpha_j}$ for a job j scheduled at position r and

Problem	Complexity	Notes	Source
$1\|[[0,\bar{s}],\omega,-,p_{js}=p_j r^{\alpha_j}]\|C_{\max}$	$\mathcal{O}(n^3)$	$\alpha_j > 0$	Zhao and Tang (2010), Lemma 1
$Rm\|[[0,\bar{s}],\omega,-,p_{ijr}=f_{ij}(i,j,r)]\|TL$	$\mathcal{O}(mn^3)$	f_{ij} nondecreasing in r	Yu et al. (2014), Theorem 1
$Pm\|[[0,\bar{s}],\omega,-,p_{js}=f_j(j,r)]\|TL$	$\mathcal{O}(n^3)$	f_j nondecreasing in r	Yu et al. (2014), Theorem 2
$Rm\|[[0,\bar{s}],\omega,-,p_{ijs}=f_{ij}(i,j,r)]\|TL$	$\mathcal{O}(n^{m+2})$	f_{ij} non-monotonic	Yu et al. (2014), Theorem 4
$1\|S_{psd},[[0,\bar{s}],\omega,-,p_{js}=p_j r^{\alpha_j}]\|\sum(\alpha E_j+\beta T_j+\eta d_1+\delta D)$	$\mathcal{O}(n^3)$	$\alpha,\beta,\eta,\delta>0$	Yang and Yang (2012), Theorem 1
$Rm\|\text{resource},[[0,\bar{s}],\omega,-,p_{ijs}=p_{ij}r^{\alpha_{ij}}-b_{ij}u_{ij}]\|\alpha_1\sum C^i_{\max}+\alpha_2\sum TC^i+\alpha_3\sum TADC^i+\alpha_4\sum\sum v_{ij}u_{ij}$	$\mathcal{O}(n^{m+2})$	$\alpha_i \geq 0$	Hsu and Yang (2014), Theorem 1
$Rm\|\text{resource},[[0,\bar{s}],\omega,-,p_{ijs}=p_{ij}r^{\alpha_{ij}}-b_{ij}u_{ij}]\|\alpha_1\sum C^i_{\max}+\alpha_2\sum TW^i+\alpha_3\sum TADW^i+\alpha_4\sum\sum v_{ij}u_{ij}$	$\mathcal{O}(n^{m+2})$	$\alpha_i,\alpha_{ij}\geq 0$	Hsu and Yang (2014), Theorem 2
$Rm\|\text{resource},[[0,\bar{s}],\omega,-,p_{ijs}=(p_{ij}r^{\alpha_{ij}}/u_{ij})^k]\|\alpha_1\sum C^i_{\max}+\alpha_2\sum TW^i+\alpha_3\sum TADW^i+\alpha_4\sum\sum v_{ij}u_{ij}$	$\mathcal{O}(n^{m+2})$	$\alpha_i,\alpha_{ij}\geq 0$	Hsu and Yang (2014), Theorem 3
$Rm\|\text{resource},[[0,\bar{s}],\omega,-,p_{ijs}=(p_{ij}r^{\alpha_{ij}}/u_{ij})^k]\|\alpha_1\sum C^i_{\max}+\alpha_2\sum TW^i+\alpha_3\sum TADW^i+\alpha_4\sum\sum v_{ij}u_{ij}$	$\mathcal{O}(n^{m+2})$	$\alpha_i,\alpha_{ij}\geq 0$	Hsu and Yang (2014), Theorem 4

Table 8. Complexity status of problems with state-dependent processing times and job-dependent deterioration factors without MAs

a job-dependent deteriorating factor $\alpha_j > 0$. The authors consider both problems, with and without maintenance scheduling, and prove them to be polynomially solvable. Some complexity results for variants of the no-maintenance problem are collected in Table 8. There may be restrictions on the number of MAs that may be conducted, or the duration of an MA may depend on its start time or on how many MAs have already been conducted within the schedule. The latter two approaches reduce or increase the MA's duration, again incorporating deterioration or learning effects. As an example, Yang and Yang (2010a) deal with problem variants of the problem introduced by Zhao and Tang (2010) that differ in the MA duration. More results and problem variants can be found in Table 9.

Problem	Complexity	Notes	Source
$1\|N^{MA}=1,d^{MA}(t)=b+ct,[[0,\bar{s}],\omega,(n,y),p_{js}=p_j r^{a_j}]\|\sum(\alpha E_j+\beta T_j+\gamma q)$	$\mathcal{O}(n^4)$	$b>0,c\geq 0$, $\alpha,\beta,\gamma\geq 0$, q: common flow allowance	Ji et al. (2014), Theorem 1
$1\|d^{MA}=t_0,[[0,\bar{s}],\omega,(n,y),p_{js}=p_j r^{\alpha_j}]\|C_{\max}$	$\mathcal{O}(n^4)$	t_0: basic MA duration, $\alpha_j>0$	Zhao and Tang (2010), Theorem 1
$1\|d^{MA}_l=t_0 l^b,[[0,\bar{s}],\omega,(n,y),p_{js}=p_j r^{\alpha_j}]\|C_{\max}$	$\mathcal{O}(n^4)$	t_0: basic MA duration, $\alpha_j>0$, $b<0$	Yang and Yang (2010a), Theorem 1

Problem	Complexity	Notes	Source
$1\|d_l^{MA} = t_0 + \beta t_l, [[0,\bar{s}],\omega,(n,y), p_{js} = p_j r^{\alpha_j}]\|C_{\max}$	$\mathcal{O}(n^5)$	t_0: basic MA duration, t_l: runtime before the $l-$th MA, $\alpha_j > 0$, $\beta \geq 0$	Yang and Yang (2010a), Theorem 3
$Rm\|N^{MA} \leq k, d_{il}^{MA} = \alpha_i + \beta_i t_{il}, [[0,\bar{s}],\omega,(n,y), p_{jir} = p_{ji} f_{ji}(r)]\|\sum C_j$	$\mathcal{O}(n^{m+k+2})$	k: maximum number of MAs, t_{il}: runtime before the $l-$th MA on machine i, $\alpha_i > 0$, $\beta_i \geq 0$, $f_{ji}(r) \geq 1$	Yang (2013), Theorem 1
$1\|N^{MA} \leq k_0, d_l^{MA} = a + bt_l, [[0,\bar{s}],\omega,(n,y), p_{js} = p_j r^{\alpha_j}]\|\sum C_j$	$\mathcal{O}(n^{k_0+3})$	t_l: machine runtime between the $(l-1)$-th and the $l-$th MA, $k_0 \leq n-1$, α_j, $a > 0$, $b \geq 0$	Yang and Yang (2010b), Theorem 1
$1\|d_l^{MA} = t_l, [[0,\bar{s}],\omega,(n,y), p_{js} = p_j g_j(r)]\|C_{\max}$	$\mathcal{O}(n^4)$	$g_j(r)$: deterioration factor of job j in position r	Rustogi and Strusevich (2012b), Theorem 6
$1\|N^{MA} = \lfloor \frac{n}{k} \rfloor, d_l^{MA} = at_l + b, [[0,\bar{s}],\omega,(n,y), p_{js} = f_j(p_j, r)]\|$ $\alpha C_{\max} + \beta \sum E_j + \gamma \sum T_j$	$\mathcal{O}(n^4)$	t_l: machine runtime between the $(l-1)$-th and the $l-$th MA, k: number of jobs between two MAs, $\alpha, \beta, \gamma > 0$, $a \geq 0$	Xue et al. (2014), Theorem 1

Table 9. Complexity status of problems with state-dependent processing times and job-dependent deterioration factors with MAs

Adopting the problem setting of Zhao and Tang (2010), our problem setting in the factory differs from the former one in Section 4.1.1 in that the connection between the processing time and a job is now individual, as well as the deterioration of the blade. The requirements of each board, together with its position, determines the processing time and the deterioration of the blade. MAs are not mandatory, but can be executed to restore it.

4.2. Deterioration of machine state linear in processing time

When dealing with problems assuming machine state-dependent processing times, we can find only a few works that also deal with maintenance scheduling (see Table 10). In

Zhang et al. (2017), processing time p_j of job j is time-dependent (deterioration rate β_j multiplied by time t) and can be interpreted as state-dependent only if there are no idle times on the machine. This type of deterioration related processing time is not part of our scope. Nevertheless, this study is interesting when considering state-dependent deterioration because maintenance scheduling is workload-dependent with workload $x = \sum p_j$ of jobs scheduled before the MA and convex nondecreasing function $f(x)$. They prove the $\mathcal{NP}$-hardness of this single machine layout for minimizing makespan and total completion time.

Problem	**Complexity**	**Notes**	**Source**
$1\|[[-\infty,\infty],\omega^p,-,$ $p_{js}=(1+\sum_{k=1}^{r-1}p_k)^A]\|C_{max}$	$\mathcal{O}(n\log(n))$		Gordon et al. (2008), p. 362
$1\|prec,[[-\infty,\infty],\omega^p,-,$ $p_{js}=(1+\sum_{k=1}^{r-1}p_k)^1]\|C_{max}$	$\mathcal{O}(n)$		Gordon et al. (2008), Theorem 4
$1\|SP,[[-\infty,\infty],\omega^p,-,$ $p_{js}=(1+\sum_{k=1}^{r-1}p_k)^2]\|C_{max}$	$\mathcal{O}(n\log(n))$		Gordon et al. (2008), Theorem 5
$1\|prec,[[-\infty,\infty],\omega^p,-,$ $p_{js}=p_j(1+\sum_{k=1}^{r-1}p_{\pi(k)})^A\|C_{max}$		Polynomial solvable	Dolgui et al. (2012)
$1\|[[-\infty,\infty],\omega^p,-,p_{js}=$ $(1+\sum_{l=1}^{r-1}p_{[l]}^G)p_j^G]\|\sum C_j^A : C_{max}^B \leq U$	$\mathcal{O}(n_A\log n_A+$ $n_B)$	2-agent model (G=A,B)	Liu et al. (2013), Theorem 1
$1\|[[-\infty,\infty],\omega^p,-,p_{js}=$ $(1+\frac{\sum_{l=1}^{r-1}p_{[l]}^G}{\sum_{l=1}^{n}p_l^G})p_j^G]\|\sum C_j^A : C_{max}^B \leq U$	$\mathcal{O}(n_A\log n_A+$ $n_B)$	2-agent model (G=A,B)	Liu et al. (2013), Theorem 1
$1\|[[0,\bar{s}],\omega^p,(y,y),p_{js}=p_j+\beta_j t]\|C_{max}$	Strongly $\mathcal{NP}$-hard	3-partition	Zhang et al. (2017), Theorem 3.2
$1\|N^{MA}=\text{fixed},[[0,\bar{s}],\omega^p,(y,y),$ $p_{js}=p_j+\beta_j t]\|C_{max}$	$\mathcal{NP}$-hard in the ordinary sense	Subset product problem, see Ji et al. (2006)	Zhang et al. (2017), Theorem 3.3
$1\|[[0,\bar{s}],\omega^p,(y,y),p_{js}=p_j+\beta_j t]\|\sum C_j$	Strongly $\mathcal{NP}$-hard	3-partition	Zhang et al. (2017), Theorem 5.1
$1\|N^{MA}=1,[[0,\bar{s}],\omega^p,(y,y),$ $p_{js}=p_j+\beta_j t]\|\sum C_j$	$\mathcal{NP}$-hard in the ordinary sense	Subset product problem, see Ji et al. (2006)	Zhang et al. (2017), Theorem 5.2

Problem	Complexity	Notes	Source
$1\|[[0,\bar{s}],\omega^p,(y,y),p_{js}=p_j+\max\{\alpha_j^D a_j-\alpha_j^U(L-l_j),0\}]\|\mu^E\sum p_j+\mu^R\sum x_i+\mu^T\sum T_j$	$\mathcal{NP}$-hard	Heuristic: simulated annealing, α_j^D: deterioration rate of machine, α_j^U: increasing rate by toolchange, $L \leq l_j$: tool lifetime; cost coefficient of energy μ^E, new tools μ^R, and of tardiness μ^T, x_i: binary variable for tool replacement	Salama and Srinivas (2021)

Table 10. Complexity status of problems with state-dependent processing times and linear deterioration

Another work dealing with a kind of maintenance scheduling and state-dependent processing times is conducted by Salama and Srinivas (2021). They assume a tool change that has to be performed in order to execute jobs. These tools are deteriorated according to the scheduled jobs' processing times. Furthermore, the actual processing time, which is limited by a lower bound, is affected by a deterioration rate of the machine α_j^D (slowing down jobs) and the actual state of the tools α_j^U (speeding up jobs). The objective is a combination of energy, tool change, and tardiness costs. After formulating a mixed integer linear programming model (MILP), in a second step two simulated-annealing based heuristics are generated (with an adaptive neighborhood search added to the second heuristic) and compared.

The paper of Gordon et al. (2008) is already covered in Section 4.1 as a paper that does not actually deal with maintenance scheduling. However, in addition to the layouts mentioned above, the authors also study problems of processing time-dependent deterioration. The actual processing time depends entirely on the sum of processing times of the jobs already processed – in the paper called *cumulative deterioration*. Minimizing makespan in this layout is examined with and without precedence constraints. The same layout is tackled by Dolgui et al. (2012). In their work, they analyze the characteristics and add, for example, learning effects to the problem layout. In the course of their research, they formulate polynomial algorithms, but do not give specific runtime analyses for their approaches.

Liu et al. (2013) present another work assuming workload-dependent processing times without any type of maintenance scheduling. A two-agent approach is studied, where the agents (machines) compete for jobs. Two different types of processing time based equations are considered to determine the actual processing time, a purely cumulative

approach and an approach that includes the total processing times.

An interesting and noteworthy problem of Seif et al. (2018) deals with a permutation flow shop problem assuming a fuzzy bi-objective function of minimizing total tardiness $\sum T_j$ and maintenance costs (second term in γ-field). The maintenance costs consist of the costs SP_{ik} to require resources for processing the MA k on machine i and costs WF_k of performing an MA of duration e_{ik}. If an MA is performed, the binary variable $y_{irk} = 1$. Processing time of the job on position r in the sequence p_{ir} is based on the actual machine state h by including factor λ to p_{ir}. In conclusion, we obtain the scheduling problem

$$Fm|prmu, [[0, \bar{s}], \omega^p, (y, y), p_{js} = \lambda^h \cdot p_{ir}]| \sum_j T_j, \sum_i \sum_j \sum_k y_{irk}(SP_{ik} + e_{ik}WF_k).$$

The authors formulate a fuzzy multi-objective linear programming based on the work of Liang (2006) and solve it with a standard genetic algorithm. It is not listed in Table 10 because no statements are made about the runtime complexity to solve the problem.

Problem	**Complexity**	**Notes**	**Source**
$1\|S_{psd}, [[-\infty, \infty], \omega^p, -, p_{js} = p_j(1 + \sum_{l=1}^{r-1} \log p_{[l]})^a]\|C_{max}$	$\mathcal{O}(n \log(n))$	a: deterioration index, $a > 1$	Cheng et al. (2011), Theorem 1
$1\|S_{psd}, [[-\infty, \infty], \omega^p, -, p_{js} = p_j(1 + \sum_{l=1}^{r-1} \log p_{[l]})^a]\|\sum C_j$	$\mathcal{O}(n \log(n))$	a: deterioration index, $a > 1$	Cheng et al. (2011), Theorem 2
$1\|S_{psd}, [[-\infty, \infty], \omega^p, -, p_{js} = p_j(1 + \sum_{l=1}^{r-1} \log p_{[l]})^a]\|\sum C_j^2$	$\mathcal{O}(n \log(n))$	a: deterioration index, $a > 1$	Cheng et al. (2011), Theorem 3
$1\|S_{psd}, agreeable, [[-\infty, \infty], \omega^p, -, p_{js} = p_j(1 + \sum_{l=1}^{r-1} \log p_{[l]})^a]\|\sum T_j$	$\mathcal{O}(n \log(n))$	a: deterioration index, $a > 1$	Cheng et al. (2011), Theorem 4
$1\|S_{psd}, agreeable, [[-\infty, \infty], \omega^p, -, p_{js} = p_j(1 + \sum_{l=1}^{r-1} \log p_{[l]})^a]\|L_{max}$	$\mathcal{O}(n \log(n))$	a: deterioration index, $a > 1$	Cheng et al. (2011), Theorem 5
$1\|B, [[0, \bar{s}], \omega^p, (y, y), p_{js} = p_{ij}(1 + \sum_{k=1}^{r-1} p_{i[k]})^{\alpha_i}]\|C_{max}$	$\mathcal{O}(n \log(n))$	α_i: aging factor of batch i	Cheng et al. (2018)
$1\|B, [[0, \bar{s}], \omega^p, (y, y), p_{js} = p_{ij}(1 + \sum_{k=1}^{r-1} p_{i[k]})^{\alpha_i}]\|\sum C_j$	$\mathcal{O}(n \log(n))$	α_i: aging factor of batch i, $\alpha_i \geq 1$	Cheng et al. (2018)
$1\|B, [[0, \bar{s}], \omega^p, (y, y), p_{js} = p_{ij}(1 + \sum_{k=1}^{r-1} p_{i[k]})^{\alpha_i}]\|\sum C_j$	open; $\mathcal{O}(n^2 \log(n))$, $\mathcal{O}(2^n)$	α_i: aging factor of batch i, $0 < \alpha_i < 1$, solution approaches: SPT & V-shaped-based heuristics	Cheng et al. (2018)

Table 11. Complexity status of problems with batching, state-dependent processing times and linear deterioration

The case outlined by Salama and Srinivas (2021) can be directly assigned to the exemplary

factory. The tool corresponds to the blade of the cutting machine. If we assume to have more than one machine, the paper of Liu et al. (2013) can be applied. Here, we can assume to have several cutting machines. Since such machines cause fixed costs even by having them ready, it is desired to exploit the capacity of each machine – the agents described in Liu et al. (2013).

In addition to these works, two papers, shown in Table 11, include batch or group scheduling characteristics in their scheduling models. Cheng et al. (2011) combine classical batch scheduling with setups between batches. Sum-of-processing-time-based deteriorating jobs, but no MAs, are considered. Cheng et al. (2018) add maintenance scheduling to this type of problem, where the actual processing time is affected only by the total processing times of the jobs scheduled in the same batch.

5. Summary of the results and future research directions

We use some descriptive statistics to gain a clear picture of the appropriateness of our classification, the distribution of the problems found in the literature (including gaps), and the computational complexity of these problems. This paper summarizes a total of 92 problems in Tables 4 to 11.

Regarding the state representation δ, most problems consider machine states within the lower and upper bounds $[0, \bar{s}]$, as seen in Table 12.

δ	$[-\infty, \infty]$	$[0, \infty]$	$[0, \bar{s}]$	Miscellaneous
Number of problems	19	16	53	5

Table 12. Distribution of the analyzed problems regarding state representation

All of the variants also appear in the literature. The five problems assigned to *miscellaneous* involve the *fixed position constraint* (see Table 4). In our framework, these problems could be denoted circumstantially, resulting in the categories $[0, \bar{s}]$ or $[0, \infty]$.

The very same five problems are not assigned in Table 13, where we consider different specifications of deteriorating machines. However, these could have been assigned to $\epsilon = w$ as well, that is, to the case in which each job deteriorates the machine by a constant w. This is by far the most prominent case found in the literature, but time-dependent deterioration w^p and individual deterioration w_j are also well represented.

ϵ	w	w^p	w_j	Miscellaneous
Number of problems	44	25	19	5

Table 13. Distribution of the analyzed problems regarding deterioration

When it comes to the distinction of how maintenance affects the machines, the classification

is no longer as balanced as before, especially since there are now many problems in the miscellaneous category, see Table 14. This is due to the numerous problems from the field of time-dependent and position-dependent processing times. While these problems do not originally focus on state-dependent machine availability, they are closely related and therefore worth mentioning in this paper.

ζ	(y,y)	(y,n)	(n,y)	(n,n)	Miscellaneous
Number of problems	18	0	27	18	30

Table 14. Distribution of the analyzed problems regarding maintenance

It is notable that we did not find any problem in the literature addressing the cases where maintenance duration depends on the state, but where maintenance operations are not required to restore a machine to the perfect state ($\zeta = (y,n)$). This is surprising given the case's high practical relevance. For instance, all types of refueling, recharging, or other types of replenishing energy storages fall under this category. The duration of the maintenance operation, that is, refueling, depends on the state of the machine, that is, the tank's fill level. Of course, in most applications, refueling to the maximum is not required. Therefore, we view this as an area of research that has been overlooked in the literature.

η	p_j	p_{js}	Miscellaneous
Number of problems	45	48	0

Table 15. Distribution of the analyzed problems regarding performance

In terms of how a machine's state affects processing times, both variants are represented equally in the literature, see Table 15. In summary, this paper discusses 93 problems that encompass a wide range of challenges related to state-dependent machine availability. It is worth mentioning that many of these problems originate from related fields, such as position-dependent processing times and inventory scheduling. Polynomial time algorithms have been proposed for 61 of these problems, although the runtime of some includes the number of machines in the exponent. The high number of polynomial solvable problems is not an indication on the complexity of the problems, but rather shows that many easily solvable cases are discussed in the literature.

We see four directions in which future research could go. The first three reveal gaps in the existing literature, while the fourth introduces new ideas on how to address practical problems in models.

As mentioned above, the literature has not yet considered the case of $\zeta = (y,n)$. Let us address this issue more generally. In industry, state dependence in scheduling is relevant in all areas where time-based approaches do not adequately address the needs of resource utilization. These are often not appropriately covered in research. For instance, the deterioration of wheels and engines on aircraft landing gear typically does not depend primarily on time-based factors driven by processing times, here flight duration. Wear

and tear of the engine and wheels depend on the number of flights since the use of the landing gear is the same regardless of the flight duration (see Sriram and Haghani (2003) and Ruther (2013)). As a result, time-based maintenance planning is potentially associated with inefficient use of resources and leads to non-optimal maintenance cycles. This issue can be tackled with state-dependent modeling approaches.

Specifically, work such as that of Hadidi et al. (2011) or Seif et al. (2018) illustrates the practical relevance of studying scheduling problems with machine deterioration linear in (literally or closely related to) processing time. The results in Tables 5, 10, and 11 show that the complexity of problems with ω^p may change dramatically compared to traditional scheduling models. This should be another starting point for future research, since to our knowledge only very specific scheduling layouts have been studied. Therefore, it is difficult to make general statements or see patterns in this class of problems. In the most general layouts, normal processing times (independent of any function) are given with MAs that fully restore the machine, that is,

$$1|[[0,\bar{s}],\omega^p,(y,y),p_j]|\gamma,$$

and

$$1|[[0,\bar{s}],\omega^p,(y,y),p_{js}]|\gamma.$$

These problems may not be as complex as ones presented in the tables regarding their model characteristics, but they can provide insight into the general influence of job scheduling with ω^p on maintenance planning.

Furthermore, layouts where the decision maker can manage the desired machine state after an MA, that is, $\zeta = (y, n)$, represent interesting features not only for research, but especially for practice by increasing the possibilities to control the machine state. The papers in Sections 3.3 and 4.2 provide an introduction to this field, but also include specifics, such as the function that determines the machine deterioration.

The second future research direction is based on the observation that most of the literature dealing with state-dependent processing times p_{js} considers processing times that depend either on the position of the job in the sequence, including potentially job-dependent factors (see Section 4.1), or on the start time of the job (see Section 4.2). Nevertheless, another notable consideration of state-dependent processing times, which is quite conceivable in a practical context and has not yet been extensively covered in the literature, is the consideration of individual deterioration ω_j per job j, which is neither dependent on the job's position nor on its start time. As a result, each job deteriorates the machine to an individual degree without any connection to the other jobs. The more the machine deteriorates, the longer it takes to process the jobs because the system slows down or because corrections to the end products are necessary. Such a problem can be discussed when the machine is used to process jobs that are technically or materially different. Thereby,

the deterioration of the machine is measured variably in terms of the machine state. For example, it can be imagined that a state represents the condition of the machine relatively, that is, in percent, varying between 0% and 100%, and each job reduces the state of the machine by a certain amount x_j%. For sure, the accuracy of the deteriorations to expect and if these are estimations or actual figures depends on the given information in the practical setting. Nevertheless, both scenarios are reasonable since one can either imagine that the actual deterioration is known due to technical information or it depends on further (possibly random) properties of the machine or the material which opens a large field for further research. These layouts can be represented by the classification

$$1|[[0, \bar{s}], \omega_j, \zeta, p_{js}]|\gamma,$$

leaving open which objective and maintenance properties are considered.

Third, several of the studies deal with both time-based and state-based characteristics that influence the machine availability (see, e.g., the works of Xue et al. (2014), Xu et al. (2015), Luo and Liu (2017), or Zhang et al. (2017)). Here, it can be asked how the different features affect the functionality and performance of the respective layout and whether interdependencies or trade-offs can be found. Furthermore, very specific objectives are pursued in the context of certain problems, such as those of Yu et al. (2014), Xue et al. (2014), or Hsu and Yang (2014). Analyzing these layouts by assuming typical completion time and due date related objective functions of scheduling research, such as total completion time, maximum lateness, etc., might help to understand the specifics of these problems and gain deeper insights. In general, the studies presented in this survey mainly aim at investigating the complexity of scheduling problems in order to understand the structure of problems. In a next step, sophisticated algorithms, either exact or heuristic, can help to improve the resource utilization in solving these problems.

Finally, we see that more complex practical decision-making situations, especially those involving stochastic processes, are rarely reflected in the literature. For instance, the question of whether to repair or replace a machine often arises. Although replacement puts the machine in perfect condition, it is usually more expensive and time-consuming. On the other hand, repair success is uncertain, so the resulting state of the machine is stochastic. Furthermore, the combination of preventive maintenance and reactive actions (e.g., after a breakdown) has not been adequately considered.

6. Conclusion

In this paper, we address traditional machine scheduling problems that are characterized by machine and/or job features leading to machine availability constraints on the execution of jobs. In contrast to the majority of works dealing with machine availability constraints, we concentrate on non-time related factors as the main cause of machine unavailability. We summarize these models including any type of non-time related machine availability constraint under the term *state-dependent machine availability*. In the course of this work, we categorize various sources of machine deterioration regarding the characteristics of jobs' processing times and their effect on machine wear and tear, and how the machine can be improved by performing maintenance activities (MAs). For this reason, we extend the three-field notation introduced by Graham et al. (1979) to display the model characteristics that differ by the machine state, deterioration, MAs, and the machine performance.

While our goal is to present and relate the scope of state-dependent machine availability to other scheduling problems with restrictions of the machine state, our approach is to analyze the papers especially in terms of the runtime complexity to solve the problems. Most of the studied problems deal with single machine layouts. It is remarkable that the assumption of any type of state-dependent machine availability regularly leads to changes in runtime complexity compared to the respective layouts with continuous machine availability. In real-world production environments, the machine state often has an impact on the performance level of the machine since performing jobs deteriorates the machine and so affects its speed when executing subsequent jobs. In research, two main streams can be identified: Models where jobs' processing times are given regardless of the machine state, and models where processing times depend on the machine state. Furthermore, several types of deterioration can be found, which differ in their form of job dependency. Although most studies assume fixed deterioration (each job deteriorates the machine equally), or machine deterioration linear in job's processing times, some works deal with jobs that individually deteriorate the machine state, that is, no direct processing time-to-deterioration-dependence is given. Overall, the research field of state-dependent machine availability in scheduling still has tremendous potential not only to fill theoretical gaps, but also to generate approaches for practical applications in production and logistics environments.

References

Ahmad, R., Kamaruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial application. *Computers & Industrial Engineering*, *63*(1), 135–149.

Albers, S., Müller, F., Schmelzer, S. (2014). Speed scaling on parallel processors. *Algorithmica*, *68*(2), 404–425.

Bampis, E., Kononov, A., Letsios, D., Lucarelli, G., Nemparis, I. (2015). From preemptive to non-preemptive speed-scaling scheduling. *Discrete Applied Mathematics*, *181*, 11–20.

Biskup, D. (1999). Single-machine scheduling with learning considerations. *European Journal of Operational Research*, *115*(1), 173–178.

Błażewicz, J., Dell'Olmo, P., Drozdowski, M., Mączka, P. (2003). Scheduling multiprocessor tasks on parallel processors with limited availability. *European Journal of Operational Research*, *149*(2), 377–389.

Błażewicz, J., Ecker, K., Pesch, E., Schmidt, G., Weglarz, J. (2019). Handbook on Scheduling. Springer.

Bock, S., Briskorn, D., Horbach, A. (2012). Scheduling flexible maintenance activities subject to job-dependent machine deterioration. *Journal of Scheduling*, *15*(5), 565–578.

Briskorn, D., Choi, B.C., Lee, K., Leung, J., Pinedo, M. (2010). Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research*, *207*(2), 605–619.

Briskorn, D., Gönsch, J., Thiemeyer, A. (2024). Scheduling maintenance activities subject to stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *319*(1), 62–78.

Briskorn, D., Jaehn, F., Pesch, E. (2013). Exact algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling*, *16*, 105–115.

Briskorn, D., Leung, J.Y. (2013). Minimizing maximum lateness of jobs in inventory constrained scheduling. *Journal of the Operational Research Society*, *64*(12), 1851–1864.

Calinescu, G., Jaehn, F., Li, M., Wang, K. (2017). An FPTAS of minimizing total weighted completion time on single machine with position constraint, in: The 28th International Symposium on Algorithms and Computation (ISAAC 2017), Schloss Dagstuhl, Leibniz-Zentrum für Informatik, Wadern, Germany. pp. 19:1–19:13.

Chattopadhyay, D. (2001). Production and maintenance planning for electricity generators: modeling and application to indian power systems. *International Transactions in Operational Research*, *8*(4), 465–490.

Chekuri, C., Khanna, S. (2001). A ptas for minimizing weighted completion time on uniformly related machines, in: Automata, Languages and Programming: 28th International Colloquium, ICALP 2001 Crete, Greece, July 8–12, 2001 Proceedings 28, Springer. pp. 848–861.

Cheng, M., Xiao, S., Luo, R., Lian, Z. (2018). Single-machine scheduling problems with a batch-dependent aging effect and variable maintenance activities. *International Journal of Production Research*, *56*(23), 7051–7063.

Cheng, T., Lee, W.C., Wu, C.C. (2011). Single-machine scheduling with deteriorating jobs and past-sequence-dependent setup times. *Applied Mathematical Modelling*, *35*(4), 1861–1867.

Cheng, T.E., Ding, Q., Lin, B.M. (2004). A concise survey of scheduling with time-dependent processing times. *European Journal of Operational Research*, *152*(1), 1–13.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. (2022). Introduction to Algorithms. MIT press, Cambridge, MA.

Costa, A., Cappadonna, F.A., Fichera, S. (2016). Minimizing the total completion time on a parallel machine system with tool changes. *Computers & Industrial Engineering*, *91*, 290–301.

Davari, M., Ranjbar, M., De Causmaecker, P., Leus, R. (2020). Minimizing makespan on a single machine with release dates and inventory constraints. *European Journal of Operational Research*, *286*(1), 115–128.

Dolgui, A., Gordon, V., Strusevich, V. (2012). Single machine scheduling with precedence constraints and positionally dependent processing times. *Computers & Operations Research*, *39*(6), 1218–1224.

Drozdowski, M., Jaehn, F., Paszkowski, R. (2017). Scheduling position-dependent maintenance operations. *Operations Research*, *65*(6), 1657–1677.

Evazabadian, F., Arvan, M., Ghodsi, R. (2019). Short-term crude oil scheduling with preventive maintenance operations: a fuzzy stochastic programming approach. *International Transactions in Operational Research*, *26*(6), 2450–2475.

Gawiejnowicz, S. (1996). A note on scheduling on a single processor with speed dependent on a number of executed jobs. *Information Processing Letters*, *57*(6), 297–300.

Gawiejnowicz, S. (2020). A review of four decades of time-dependent scheduling: Main results, new topics, and open problems. *Journal of Scheduling*, *23*(1), 3–47.

Gawiejnowicz, S. (2026). Theory and methodology of time-dependent scheduling: Past, present and future. *European Journal of Operational Research*, *329*(3), 719–746.

Geurtsen, M., Didden, J.B., Adan, J., Atan, Z., Adan, I. (2023). Production, maintenance and resource scheduling: A review. *European Journal of Operational Research*, *305*(2), 501–529.

Gordon, V.S., Potts, C.N., Strusevich, V.A., Whitehead, J.D. (2008). Single machine scheduling models with deterioration and learning: handling precedence constraints via priority generation. *Journal of Scheduling*, *11*, 357–370.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Annals of Discrete Mathematics. Elsevier, Amsterdam. volume 5, pp. 287–326.

Grigoriu, L., Briskorn, D. (2017). Scheduling jobs and maintenance activities subject to job-dependent machine deteriorations. *Journal of Scheduling*, *20*(2), 183–197.

Hadidi, L.A., Al-Turki, U.M., Rahim, M.A. (2011). An integrated cost model for production scheduling and perfect maintenance. *International Journal of Mathematics in Operational Research*, *3*(4), 395–413.

Hadidi, L.A., Al-Turki, U.M., Rahim, M.A. (2015). Practical implications of managerial decisions to integrate production scheduling and maintenance. *International Journal of System Assurance Engineering and Management*, *6*, 224–230.

Hipp, A. (2023). Scheduling position-dependent maintenance operations in single machine layouts, in: International Conference on Operations Research, Springer. pp. 25–31.

Hipp, A., Jaehn, F. (2024). Single machine scheduling problems with position-dependent maintenance, Working Paper. Helmut Schmidt University. Hamburg, Germany.

Hsu, C.J., Yang, D.L. (2014). Unrelated parallel-machine scheduling with position-dependent deteriorating jobs and resource-dependent processing time. *Optimization Letters*, *8*, 519–531.

Jaehn, F. (2024). Scheduling with jobs at fixed positions. *European Journal of Operational Research*, *318*(2), 388–397.

Jansen, K., Klein, K.M., Verschae, J. (2020). Closing the gap for makespan scheduling via sparsification techniques. *Mathematics of Operations Research*, *45*(4), 1371–1392.

Ji, M., He, Y., Cheng, T.E. (2006). Scheduling linear deteriorating jobs with an availability constraint on a single machine. *Theoretical Computer Science*, *362*(1-3), 115–126.

Ji, P., Li, G., Huo, Y., Wang, J.B. (2014). Single-machine common flow allowance scheduling with job-dependent aging effects and a deteriorating maintenance activity. *Optimization Letters*, *8*, 1389–1400.

Kendall, D.G. (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, , 338–354.

Koulamas, C. (1994). The total tardiness problem: review and extensions. *Operations Research*, *42*(6), 1025–1041.

Kubzin, M.A., Strusevich, V.A. (2006). Planning machine maintenance in two-machine shop scheduling. *Operations Research*, *54*(4), 789–800.

Kuo, W., Yang, D.L. (2008). Minimizing the makespan in a single-machine scheduling problem with the cyclic process of an aging effect. *Journal of the Operational Research Society*, *59*(3), 416–420.

Lee, C.Y., Leon, V.J. (2001). Machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, *128*(1), 119–128.

Lee, C.Y., Lin, C.S. (2001). Single-machine scheduling with maintenance and repair rate-modifying activities. *European Journal of Operational Research*, *135*(3), 493–513.

Lee, J.Y., Kim, Y.D. (2017). Minimizing total tardiness in a two-machine flowshop scheduling problem with availability constraint on the first machine. *Computers & Industrial Engineering*, *114*, 22–30.

Liang, T.F. (2006). Distribution planning decisions using interactive fuzzy multi-objective linear programming. *Fuzzy Sets and Systems*, *157*(10), 1303–1316.

Liu, P., Yi, N., Zhou, X., Gong, H. (2013). Scheduling two agents with sum-of-processing-times-based deterioration on a single machine. *Applied Mathematics and Computation*, *219*(17), 8848–8855.

Low, C., Ji, M., Hsu, C.J., Su, C.T. (2010). Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Applied Mathematical Modelling*, *34*(2), 334–342.

Luo, W., Liu, F. (2017). On single-machine scheduling with workload-dependent maintenance duration. *Omega*, *68*, 119–122.

Ma, Y., Chu, C., Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, *58*(2), 199–211.

Miwa, M., Oyama, T. (2018). An optimal track maintenance scheduling model analysis taking the risk of accidents into consideration. *International Transactions in Operational Research*, *25*(5), 1465–1490.

Montoya-Torres, J.R., Botta-Genoulaz, V., Materzok, N., Gíslason, Þ.P., Mendiela, S. (2021). Modeling the parallel machine scheduling problem with worker-and position-dependent processing times, in: Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems: IFIP WG 5.7 International Conference, APMS 2021, Nantes, France, September 5–9, 2021, Proceedings, Part III, Springer. pp. 351–359.

Montoya-Torres, J.R., Moreno-Camacho, C.A., Vélez-Gallego, M.C. (2023). Variable neighbourhood search for job scheduling with position-dependent deteriorating processing times. *Journal of the Operational Research Society*, *74*(3), 873–887.

Mor, B., Mosheiov, G. (2012). Scheduling a maintenance activity and due-window assignment based on common flow allowance. *International Journal of Production Economics*, *135*(1), 222–230.

Mor, B., Mosheiov, G. (2015). Scheduling a deteriorating maintenance activity and due-window assignment. *Computers & Operations Research*, *57*, 33–40.

Morsy, E., Pesch, E. (2015). Approximation algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling*, *18*, 645–653.

Mosheiov, G. (2001). Scheduling problems with a learning effect. *European Journal of Operational Research*, *132*(3), 687–693.

Mosheiov, G. (2005). A note on scheduling deteriorating jobs. *Mathematical and Computer Modelling*, *41*(8-9), 883–886.

Mosheiov, G., Sarig, A. (2008). A due-window assignment problem with position-dependent processing times. *Journal of the Operational Research Society*, *59*(7), 997–1003.

Mosheiov, G., Sidney, J.B. (2003). Scheduling with general job-dependent learning curves. *European Journal of Operational Research*, *147*(3), 665–670.

Mosheiov, G., Sidney, J.B. (2005). Note on scheduling with general learning curves to minimize the number of tardy jobs. *Journal of the Operational Research Society*, *56*, 110–112.

Mosheiov, G., Sidney, J.B. (2010). Scheduling a deteriorating maintenance activity on a single machine. *Journal of the Operational Research Society*, *61*(5), 882–887.

Nawaz, M., Enscore Jr, E.E., Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, *11*(1), 91–95.

Pei, J., Zhou, Y., Yan, P., Pardalos, P.M. (2023). A concise guide to scheduling with learning and deteriorating effects. *International Journal of Production Research*, *61*(6), 2010–2031.

Perez-Gonzalez, P., Framinan, J.M. (2018). Single machine scheduling with periodic machine availability. *Computers & Industrial Engineering*, *123*, 180–188.

Rudek, R. (2026). A fast local search for the identical parallel machine scheduling problem with the position-based deteriorating effect and maintenance. *International Transactions in Operational Research*, *33*(5), 3226–3257.

Rustogi, K., Strusevich, V.A. (2012a). Simple matching vs linear assignment in scheduling models with positional effects: A critical review. *European Journal of Operational Research*, *222*(3), 393–407.

Rustogi, K., Strusevich, V.A. (2012b). Single machine scheduling with general positional deterioration and rate-modifying maintenance. *Omega*, *40*(6), 791–804.

Ruther, S. (2013). Integrated aircraft routing, crew pairing, and tail assignment, Unpublished doctoral dissertation, University of Newcastle, Newcastle, Australia.

Salama, M., Srinivas, S. (2021). Adaptive neighborhood simulated annealing for sustainability-oriented single machine scheduling with deterioration effect. *Applied Soft Computing*, *110*, 107632.

Sanlaville, E., Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, *35*(9), 795–811.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, *121*(1), 1–15.

Seif, J., Yu, A.J., Rahmanniyay, F. (2018). Modelling and optimization of a bi-objective flow shop scheduling with diverse maintenance requirements. *International Journal of Production Research*, *56*(9), 3204–3225.

Sriram, C., Haghani, A. (2003). An optimization model for aircraft maintenance scheduling and reassignment. *Transportation Research Part A: Policy and Practice*, *37*(1), 29–48.

Strusevich, V.A., Rustogi, K., et al. (2017). Scheduling with Time-Changing Effects and Rate-Modifying Activities. Springer, Cham.

Tian, H., Yu, S., Luo, W. (2019). Parallel machine scheduling on jobs and partial maintenance activities due to job-dependent machine deteriorations. *American Journal of Mathematical and Management Sciences*, *38*(3), 250–260.

Wang, J.J., Wang, J.B., Liu, F. (2011). Parallel machines scheduling with a deteriorating maintenance activity. *Journal of the Operational Research Society*, *62*(10), 1898–1902.

Xu, D., Wan, L., Liu, A., Yang, D.L. (2015). Single machine total completion time scheduling problem with workload-dependent maintenance duration. *Omega*, *52*, 101–106.

Xu, D., Xu, Z., Xu, L., Yu, X. (2022). Short-term scheduling with machine calibration. *International Transactions in Operational Research*, *29*(5), 3031–3050.

Xu, Z., Xu, D. (2015). Single-machine scheduling with preemptive jobs and workload-dependent maintenance durations. *Operational Research*, *15*, 423–436.

Xue, P., Zhang, Y., Yu, X. (2014). Single-machine scheduling with piece-rate maintenance and interval constrained position-dependent processing times. *Applied Mathematics and Computation*, *226*, 415–422.

Yang, D.L., Cheng, T., Yang, S.J. (2014). Parallel-machine scheduling with controllable processing times and rate-modifying activities to minimise total cost involving total completion time and job compressions. *International Journal of Production Research*, *52*(4), 1133–1141.

Yang, S., Yang, D.L. (2012). Single-machine scheduling problems with past-sequence-dependent delivery times and position-dependent processing times. *Journal of the Operational Research Society*, *63*(11), 1508–1515.

Yang, S.J. (2013). Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. *Applied Mathematical Modelling*, *37*(5), 2995–3005.

Yang, S.J., Yang, D.L. (2010a). Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. *Omega*, *38*(6), 528–533.

Yang, S.J., Yang, D.L. (2010b). Minimizing the total completion time in single-machine scheduling with aging/deteriorating effects and deteriorating maintenance activities. *Computers & Mathematics with Applications*, *60*(7), 2161–2169.

Yu, X., Zhang, Y., Huang, K. (2014). Multi-machine scheduling with general position-based deterioration to minimize total load revisited. *Information Processing Letters*, *114*(8), 399–404.

Zhang, X., Yin, Y., Wu, C.C. (2017). Scheduling with non-decreasing deterioration jobs and variable maintenance activities on a single machine. *Engineering Optimization*, *49*(1), 84–97.

Zhang, Y., Wang, Y., Bai, X., Liu, Y. (2026). Designing a robust sustainable service network for agricultural machinery maintenance under demand uncertainty. *International Transactions in Operational Research*, *33*(3), 1927–1970.

Zhao, C.l., Tang, H.y. (2010). Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan. *Applied Mathematical Modelling*, *34*(3), 837–841.

B. Article II – Scheduling maintenance activities subject to stochastic job-dependent machine deterioration

Title: Scheduling maintenance activities subject to stochastic job-dependent machine deterioration
Authors: Briskorn, D., Gönsch, J., and Thiemeyer, A.
Journal: European Journal of Operational Research
Status: Published
Citation: Briskorn, D., Gönsch, J., Thiemeyer, A. (2024). Scheduling maintenance activities subject to stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *319*(1), 62-78. https://doi.org/10.1016/j.ejor.2024.06.030

The following pages present a reprinted version of the above article, reformatted to match the layout of this dissertation.

Copyright:
© 2026 The Authors. Published by Elsevier B.V. This is an open access article distributed under the terms of the Creative Commons Attribution (CC BY) License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Scheduling maintenance activities subject to stochastic job-dependent machine deterioration

Dirk Briskorn[1], Jochen Gönsch[2], Antonia Thiemeyer[1]

[1]: Bergische Universität Wuppertal, Schumpeter School of Business and Economics

[2]: Universität Duisburg-Essen, Mercator School of Management

This research has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 452470135.

Abstract

This paper considers a maintenance scheduling problem on a single machine with a fixed job sequence involving job-dependent, stochastic deterioration. If the machine breaks down, then a time-consuming emergency maintenance activity needs to be conducted. Instead, planned maintenance activities can be conducted between jobs in order to improve the machine's state and, thus, prevent the machine from breaking down. We distinguish two types of machines differing in the flexibility regarding the point of time when the maintenance activities need to be scheduled. The goal is to minimize the total expected completion time of jobs. We formalize two optimization problems, analyze their computational complexities, develop exact solution approaches, and conduct a computational study. Here, we show that our approaches can solve large instances to optimality within short run times and we analyze the advantage gained by planning maintenance activities depending on the actual state of the machine.

Keywords: Dynamic Programming · Stochastic job-dependent machine deterioration · Maintenance activities · Single machine

1. Introduction

In most machine scheduling papers the authors assume continuously available machines over the whole planning horizon (see Brucker (2004) and Pinedo (2009)). However, in real world settings the machine needs to be maintained from time to time and is not available during these maintenance activities. Thus, the production flow is directly affected by the

deterioration of the manufacturing system. This is why scheduling problems with machine availability restrictions came into the focus of research in the past decades.

In this paper we consider a variant of the problem introduced by Bock et al. (2012). The condition of a single machine is measured by a *maintenance level* (ML) and is represented by an integer number from the interval $[0, l^{\max}]$, where $l^{\max}$ is an integer value greater than zero, that is $l^{\max} \in \mathbb{N}_{>0}$. The higher the ML is, the better is the machine's condition. If the ML amounts to the maximum value $l^{\max}$, the machine is in perfect shape. On the other hand, a ML of zero while processing a job signals a machine breakdown, which can only be resolved by time-intensive repair work.

Two different operations have a direct impact on the ML of the machine. On the one hand, the ML drops while jobs are processed according to the stochastic deterioration associated with the jobs. On the other hand, the ML can be increased by performing time-consuming *maintenance activities* (MAs). In our setting any MA improves the ML to the maximum $l^{\max}$. We distinguish between two different types of MAs, the *planned maintenance activities* (PMAs) which shall be scheduled and the *emergency maintenance activities* (EMAs) which arise when the machine breaks down. Between each pair of consecutive jobs at most one PMA can be scheduled. Similarly, at most one PMA can be scheduled before the first job and at most one PMA can be scheduled after the last job. There are no restrictions on the total number of PMAs that shall be conducted. The duration of a PMA depends on the current condition of the machine when the PMA starts. The worse the condition of the machine is, the longer the PMA takes. An EMA does essentially the same but necessarily starts with a ML at zero and might be even more time-consuming since it needs to be conducted spontaneously and, hence, not all the materials or personnel are available right away. During any MA, whether PMA or EMA, no jobs can be processed. As a consequence, the execution of maintenance work delays the production flow, which has an impact, among others, on the completion times of all following individual jobs.

In this paper, two different optimization problems are considered in which the scheduling of PMAs has to be done for a fixed job sequence with the goal to minimize the expected total completion time. Basically, the two optimization problems differ in whether decisions about PMAs need to be taken at the beginning of the planning horizon or the decisions concerning PMAs are made after each job depending on the actual ML. For the second version the information about the current ML is needed at any time. This is why we refer to the two versions as the optimization problem on the *non-transparent machine* and on the *transparent machine*, respectively.

While a fixed job sequence seems to yield a very narrow problem setting, we actually encountered a similar setting at a pharmaceutical company in Germany where production campaigns are planned months, sometimes years, in advance. Due to the huge setup effort between production of different goods the sequence of production lots is fixed and lots are

produced over weeks, sometimes months. The production system's deterioration is subject to stochasticity. Therefore, integrating PMAs to weigh delays caused by PMAs against larger delays caused by EMAs would be interesting. Even if we encounter a problem setting where the job sequence is not fixed we can benefit from an approach solving our problem fast. In a decomposition approach a job sequence can be determined on the upper level, e.g. using a meta-heuristic searching good sequences. On the lower level, we then need to evaluate candidate job sequences which can be accomplished by solving the problem at hand. Of course, if we employ such a decomposition approach we rely on an efficient approach solving our problem fast enough to allow repetitive application.

The integration of MAs into machine scheduling problems has attracted significant research interest in recent decades. In fact, there are already a number of surveys that give an overview over related problems. The earliest surveys covering limited machine availability periods due to preventive MAs (deterministic), machine breakdowns (stochastic) or other arbitrary availability constraints are provided by Sanlaville and Schmidt (1998), Schmidt (2000) and Lee et al. (1997, Section 1.2). These surveys cover results for a single machine and multiple machines and distinguish between preemptive and non-preemptive jobs. More recently Ma et al. (2010) and Strusevich et al. (2017) provided surveys dealing with the same topics. All these surveys have in common that unavailability of machines due to MAs is not affected by properties of jobs, but MAs must be scheduled according to availability constraints like, for example, time windows. Ahmad and Kamaruddin (2012) give an overview of literature that covers settings where unavailability of machines may occur due to machine breakdowns or high failure rates. In both cases MAs are conducted to restore the machine and the production quality. Thereby the authors distinguish between time-based maintenance, assuming the failure behavior in time of the equipment is predictable, and condition-based maintenance, assuming the machine's condition can be concluded by technical monitoring parameters, such as e.g. vibration, temperature and noise levels. The survey differs from the previous ones as they have a focus on the applicability to real-world problems. Thereby the deterioration on the machine is not considered as a job-dependent parameter, which distinguishes our model from the literature listed. To the best of our knowledge, Geurtsen et al. (2023) provide the most recent survey reviewing scheduling problems involving MAs, resource constraints, or a combination of both. Among others, they consider rate-modifying activities which are close to our MAs in that they improve the machine's state (see Lee and Leon (2001) for an introduction). For the sake of a unique terminology we will speak of MAs in the following even if the authors speak of rate-modifying activities. The processing time of a job may depend on the total processing time of jobs scheduled either since the start of the planning horizon or after the last MA. In our model, while processing times of jobs are given as input and cannot be modified the impact of a job on the machine's state is stochastic and not necessarily correlated with its processing time.

Machine scheduling models accounting for MAs differ with respect to multiple features. In the following, we give an overview of these features starting with the machines' speeds which may be influenced by MAs. Afterwards, we discuss MAs which ensure availability of machines and restrictions on MAs' durations and their number.

Generally, MAs may improve the condition of the machine in order to remedy deterioration potentially cancelling out learning effects. First, we can find rather simple interdependencies between the schedule and a machine's speed. For example, in Lee and Lin (2001) the authors consider a single-machine scheduling problem involving one potential MA and at most one machine breakdown. The processing time of jobs depends on whether or not a MA has been carried out previously in the schedule. Thereby they assume an arbitrary distribution function for the probability of a machine breakdown. One of their results for the problem of minimizing expected total completion times is, that the SPT sequence is an optimal schedule if the distribution function of a machine breakdown is concave. Alternatively, the processing times of the jobs can also depend on their position in the schedule, see, e.g., Gordon et al. (2008). For example, Zhao and Tang (2010) consider a scheduling problem on a single machine to schedule jobs and an arbitrary number of MAs to minimize the makespan. The processing times of the jobs depend on their positional distance to the previous MA (or the beginning of the planning horizon). A schedule with minimum makespan is found in polynomial time. However, many models assume a finer dependency between the machine's state and its performance. Often, then, jobs' processing times are assumed to depend on their start times or the elapsed timespan between the last MA and their start times. Lodree Jr and Geiger (2010) propose an optimal policy for scheduling a single MA to minimize makespan when the processing times of the jobs are given as the product of their start time and a position-dependent deterioration factor. Zhang et al. (2017) consider the problem of scheduling a set of jobs whose processing times are a non-decreasing linear function of their start time, respectively the elapsed time since the last MA, and an arbitrary number of MAs can be conducted. The authors show that the problem to minimize the makespan is NP-hard. Variations and combinations of different factors with an impact on the processing times are also possible and can be found in the literature. E.g., Yang (2010) deals with a problem of scheduling at most one MA on a single machine taking start-time-dependent learning effects and position dependent deterioration into account. Thereby, the aim is to minimize the makespan, the total completion time, or the total absolute deviation of completion times. It can be shown that all considered problems, differing exclusively in their objective function, can be solved in polynomial time. In summary, a job's processing time may be influenced by the deteriorating system and may depend on its position in the sequence, its start time (cf. Gawiejnowicz (2020)) or any other characteristics of the job schedule accounting for deterioration of the machine or learning effects. Moreover, processing times may also depend on resource allocations, see Shabtay and Steiner (2007) and Vickson (1980). In our model, however, the processing times of jobs are given as input and cannot be modified.

Besides the consideration of MAs modifying the machines' speeds, there is also a large body of literature dealing with MAs which are necessary for keeping machines available at all. In most machine scheduling models accounting for the necessity of having MAs we usually find little flexibility for scheduling them. Often, the time windows (defined by start time and end time) to carry out MAs are fixed in advance, see Lee (2004), Gawiejnowicz and Kononov (2010), and Mellouli et al. (2009) for example, or MAs need to be carried out periodically, see, e.g., Xu et al. (2020) and Liao and Chen (2003). Other papers define a lower and upper bound for the time that is allowed to elapse between two MAs, see Chen (2008), Xu et al. (2009), and Sun and Li (2010). In both cases the motivation for these restrictions is to ensure that a MA is carried out before any events occur that shall be prevented, as for example machine breakdowns. Often, the machine's state does not need to be observed explicitly. In our model, we have complete freedom to decide whether and when to schedule PMAs, that is we can willingly accept the risk of machine breakdowns. The only driver to have PMAs at all is their impact on the completion time of jobs.

Models in the literature differ in durations of MAs, as well. Some authors assume fixed MA durations, see, e.g., Mor and Mosheiov (2012), while others deal with durations to be determined as part of the solution, for example Mosheiov and Sidney (2010) and Luo et al. (2015). In Yang and Yang (2010), e.g., durations may depend on MAs' positions in the schedule or the elapsed time since the last MA was completed or both. In a first setting it is assumed that the duration of a MA increases with time passed since the last MA. This goes along with the understanding that the more the machine was used, the more it is worn down, and the longer it takes to restore its condition, regardless of how many MAs have been carried out in the past. In a second setting a learning effect is captured. The more often the machine has been maintained in the past, the more efficient MAs are now and they take less time. Both settings, however, do not explicitly account for the state of the machine. More general approaches allow MA durations depending on the condition of the machine which is not necessarily determined by the time elapsed or the number of MAs performed, see, e.g., Bock et al. (2012) and Grigoriu and Briskorn (2017). Then, the state of the machine is specified explicitly. Accurately tracking the machine's state is also highly relevant in practice with regard to the research field of predictive maintenance. In many areas of engineering, research is being carried out on sensor technologies in order to be able to read out the condition of machines and schedule MAs based on this data or modeling and simulation processes based on historical data, especially to prevent machine breakdowns (cf. Mobley (2002)). In our model, we follow Bock et al. (2012) and Grigoriu and Briskorn (2017) and assume that the MAs' durations depend on the machine's state which in turn depends on the earlier scheduled jobs and MAs.

Further, the number of MAs to be performed can be restricted. Some authors allow only one MA to be conducted over the whole planning horizon, see, e.g., Mosheiov and Sarig (2009) and Kubzin and Strusevich (2006), others allow a larger number, for example Yang

(2013). In both cases the considerations can be motivated by budget restrictions. We do not have any restriction on the total number of MAs in our problem setting.

The problem closest to the ones treated in this paper has been introduced by Bock et al. (2012) and considers a single machine and job-dependent deterministic deterioration. Machine breakdowns have to be prevented categorically by scheduling PMAs timely. Two different types of PMAs, full and partial MAs, are distinguished. Thereby, a full MA increases the ML to the maximum and a partial MA increases the ML by an arbitrary, feasible increment. The duration of the MA linearly increases with the improvement of the ML. Various objective functions are considered, namely total completion time, maximum lateness, and number of tardy jobs. The complexity analysis revealed that the scheduling problems restricted to full MAs are all NP-hard, except when the deterioration is equal among all jobs. In this case, the optimization problem can be solved in polynomial time for all objective functions considered. On the other hand, almost all problem variants allowing partial MAs can be solved in polynomial time. Luo et al. (2019) later complemented the complexity results by showing that minimization of total completion time with partial MAs is NP-hard. Grigoriu and Briskorn (2017) focus on makespan minimization in the setting with full MAs introduced by Bock et al. (2012). They provide a 5/4 linear-time approximation algorithm for a single machine and, in addition, extend the setting to parallel machines, analyze approximability, and propose heuristics.

In the paper at hand we take the problem setting with full MAs from Bock et al. (2012) and restrict ourselves to a fixed job sequence, but extend the problem to job-dependent stochastic deterioration following discrete probability distributions. The objective is to minimize the total expected completion time. Machine breakdowns do not have to be avoided categorically but we might suffer from longer durations of resulting EMAs. Specifically, our contributions are the following.

- We introduce the problem setting with a fixed job sequence and stochastic deteriorations caused by jobs and formalize it.
- We prove that our problems are binary NP-hard even though the job sequence is fixed.
- We develop exact solution approaches for both machine types based on dynamic programming. Both algorithms run in pseudo-polynomial time and solve even large instances in less than 3.1 s of run time.
- We conduct a computational study evaluating the two solution approaches and the impact of the machine type on the objective value.

The paper is organized as follows: In Section 2 the problems are formally defined and we provide the analysis of complexity. In Sections 3 and 4 we present our solution approaches based on dynamic programming (cf. Powell (2011)). Section 5 provides the computational study and, finally, Section 6 concludes the paper and gives an outlook for further extensions

of the considered problems including open research questions.

2. Maintenance planning for a given job sequence

This section introduces the optimization problems tackled in this paper. We start with a precise problem definition in Section 2.1 while the computational complexity is analyzed in Section 2.2. Section 2.3 states some theoretical results for the relationship between both optimization problems.

2.1. Problem definition

We consider a single machine with a maximum maintenance level (ML) $l^{\max}$ and an initial ML l_0, $0 \leq l_0 \leq l^{\max}$. The machine is either transparent or not which gives rise to two problem variants. A set of jobs $J = \{1, \ldots, n\}$ is processed in ascending order of the job indices. The ML is affected over time by jobs being processed and non-preemptive maintenance activities (MA) being conducted.

Each job $j \in J$ has processing time p_j and causes the ML to decrease by $\tilde{D}_j$. Thereby, $\tilde{D}_j$ is a discrete random variable which can take the values $\Delta_j := \{\delta_j^1, \ldots, \delta_j^{K_j}\} \subset \mathbb{N}$ underlying the discrete probability distribution $P_j(\tilde{D}_j = \delta_j^k) = q_j^k \in (0, 1]$ for $k = 1, \ldots, K_j$. Here, K_j is the number of possible machine deteriorations caused by $j \in J$. The realization $\tilde{D}_j = \delta_j^k$ indicates that job j causes a reduction in the ML by δ_j^k and has probability q_j^k. Naturally, we have $\sum_{k=1}^{K_j} q_j^k = 1$ for each $j \in J$. We assume that the deterioration occurs while processing the job but do not rely on a particular model how deterioration is distributed over the processing interval. We assume all parameters except the probabilities to be integers. For the sake of clarity, we denote all random variables with a tilde.

We denote the dependent discrete random variable representing the ML at the completion of job $j \in J$ as $\tilde{L}_j$. Here, for simplicity of notation $\tilde{L}_0$ reflects the certain initial ML of l_0. The corresponding probability distribution arises fully from the deterioration probabilities of all jobs processed so far and the PMA decisions. A solution M is a $(l^{\max}+1) \times n$ matrix in $\{0, 1\}$. Entry $m_{l,j}$ signals whether (1) or not (0) an PMA is planned immediately before job j, $j = 1, \ldots, n$, if the ML $\tilde{L}_{j-1}$ after completing job $j-1$ equals l, $l = 0, \ldots, l^{\max}$. If the machine is transparent we can choose each entry individually. If the machine is not transparent, we cannot determine our decision based on the current ML and, therefore, have to choose $m_{l,j} = m_j$ for all $l = 0, \ldots, l^{\max}$, i.e. independent of l. Thus, we simplify notation for a non-transparent machine such that a solution is represented by a binary vector M with n entries with entry m_j signaling whether or not a PMA is planned

J	Set of jobs $\{1, \ldots, n\}$
$\tilde{C}_j$	Completion time of job $j \in J$
$\tilde{D}_j$	Deterioration of job $j \in J$
Δ_j	Set of possible deteriorations of job $j \in J$
δ_j^k	k-th deterioration of job $j \in J$ for $k = 1, \ldots, K_j$
p_j	Processing time of job $j \in J$
d^{ema}	Duration of an EMA
$E_j^{\text{ema}}(L)$	Expected number of EMAs during job j started with ML $0 \leq L \leq l^{\max}$
K_j	Number of possible machine deteriorations caused by job $j \in J$
$\tilde{L}_j$	ML at the completion of job j
$l^{\max}$	Maximum ML
l_0	Initial ML
n	Number of jobs
q_j^k	Probability of $\tilde{D}_j = \delta_j^k$
m_j	Binary decision variable for MPP-N indicating if there is a PMA scheduled before job j
$m_{l,j}$	Binary decision variable for MPP-T indicating if there is a PMA scheduled before job j at ML $0 \leq l \leq l^{\max}$
Z	Total expected completion time

Table 1. Notation for MPP-N and MPP-T.

immediately before job j.

We assume that a PMA raises the ML to $l^{\max}$ with a rate of 1. Thus, a PMA planned immediately before job $j \in J$ takes $l^{\max} - \tilde{L}_{j-1}$ time units and leaves the machine with ML $l^{\max}$. If the ML drops to zero during processing and before completion of a job an EMA becomes due. An EMA raises the ML to $l^{\max}$ and takes a given duration $d^{\text{ema}} \geq l^{\max}$. The excess duration of $d^{\text{ema}} - l^{\max}$ as compared to a PMA also improving the ML from 0 to l^{max} is due to equipment or personnel being required at the machine unexpectedly. Note that if δ_j^k with $\delta_j^k > l^{max}$ materializes, then EMAs cannot be avoided entirely and even multiple may be necessary during the processing of job j.

Let $\tilde{C}_j$ be the random variable representing the completion time of job $j \in J$ and let $\tilde{C}_0 = 0$ for ease of notation. Note that $\tilde{C}_j$ as well as $\tilde{L}_{j-1}$ depends on the solution at hand, but we have omitted an additional M in the index for reasons of readability. We then can express $\tilde{C}_j$ as the sum of $\tilde{C}_{j-1}$, a random variable describing the duration of a PMA between $j-1$ and j, j's deterministic processing time, and a random variable describing the total duration of EMAs during the processing of j. We seek to determine a solution that minimizes the expected total completion time $E[\sum_{j=1}^n \tilde{C}_j] = \sum_{j=1}^n E[\tilde{C}_j]$ of jobs where $E[\tilde{C}_j]$ can in turn be expressed as the sum of expected values of each of the summands specified above. In order to outline how a solution determines the expected total completion time we, first, derive expected numbers of EMAs depending on the ML reached when starting a job.

Job $j \in J$ starts immediately after the previous job if no PMA is planned in between.

Otherwise, it starts immediately after the preceding PMA. In both cases, j's completion is delayed if EMAs occur during j's processing. Obviously, the number of EMAs during job j depends on the ML L the job is started with and the realized deterioration. We denote the expected number of EMAs during job j started with ML L as $E_j^{\text{ema}}(L)$. It is given by

$$E_j^{\text{ema}}(L) = \sum_{k=1,\dots,K_j} q_j^k \cdot \max\{0, \lceil (\delta_j^k - L)/l^{\max} \rceil\}.$$

Note that if δ_j^k with $\delta_j^k > L$ materializes, then $\lceil (\delta_j^k - L)/l^{\max} \rceil > 0$ EMAs must compensate for the total deficiency $\delta_j^k - L > 0$ of ML. If, however, δ_j^k with $\delta_j^k \leq L$ materializes, then no EMA is necessary.

With $E_j^{\text{ema}}(L)$ we are now equipped to derive $E[\tilde{C}_j]$. For a solution M we get for a non-transparent machine (1) and a transparent machine (2), respectively,

$$E[\tilde{C}_j] = E[\tilde{C}_{j-1}] + \underbrace{m_j \cdot (l^{\max} - E[\tilde{L}_{j-1}])}_{\text{expected duration of PMA before } j} + p_j + \underbrace{\sum_{l=0}^{l^{\max}} q_{j-1,l} \cdot E_j^{\text{ema}}(l + m_j \cdot (l^{\max} - l)) \cdot d^{\text{ema}}}_{\text{expected number of EMAs during } j} \tag{1}$$

$$E[\tilde{C}_j] = E[\tilde{C}_{j-1}] + \underbrace{E[m_{\tilde{L}_{j-1},j} \cdot (l^{\max} - \tilde{L}_{j-1})]}_{\text{expected duration of PMA before } j} + p_j + \underbrace{\sum_{l=0}^{l^{\max}} q_{j-1,l} \cdot E_j^{\text{ema}}(l + m_{l,j} \cdot (l^{\max} - l)) \cdot d^{\text{ema}}}_{\text{expected number of EMAs during } j} \tag{2}$$

where $q_{j-1,l}$ is the probability that job $j-1$ completes with a ML of l if $j > 1$ and PMAs are conducted according to M. It holds $q_{0,l_0} = 1$ and $q_{0,l} = 0$ for each $l \neq l_0$. Note that $q_{j-1,l}$ itself depends on M. In both variants, the expected completion time $E[\tilde{C}_{j-1}]$ of job $j-1$ and the processing time p_j contribute. Furthermore, for a non-transparent machine, the expected duration of a PMA before j does not depend on the current ML (which is unknown), but its expectation. For a transparent one, it depends on the current ML $\tilde{L}_{j-1}$. Note again that random variable $\tilde{L}_{j-1}$ does not only depend on $\tilde{D}_1, \dots, \tilde{D}_{j-1}$ but also on the realized PMAs before job $j-1$ according to M. Note, furthermore, that no term in (1) or (2) depends on assumptions on how the deterioration, that is the decrement of the ML caused by a job is distributed over time during its processing.

To conclude, our maintenance planning problem with a non-transparent machine (MPP-N) and a transparent machine (MPP-T), respectively, is to find a solution with minimum total expected completion time of jobs, which is given by $Z = E[\sum_{j=1}^{n} \tilde{C}_j] = \sum_{j=1}^{n} E[\tilde{C}_j]$. Table 1 summarizes the notation.

In particular, the fixed PMA after the last job has no influence on the objective function,

as it does not delay the completion of any job.

Furthermore, we consider a decision version of the problems (MPP-N-DEC and MPP-T-DEC) asking if a total expected completion time of no more than a given constant Γ is possible.

We can see that processing times of jobs contribute a constant of $\sum_{j\in J}(n+1-j)\cdot p_j$ to the total expected completion time of each solution. We, hence, in the remainder of the paper assume that $p_j = 0$ for each $j \in J$.

Example 1. We consider an instance I where a sequence of three jobs, $J = \{1, 2, 3\}$, is processed on a machine with $l^{\max} = 6$, $l_0 = 4$ and $d^{\text{ema}} = 25$. Job parameters are given in Table 2.

j	p_j	k	δ_j^k	$P_j(\tilde{D}_j = \delta_j^k) = q_j^k$
1	0	1	2	0.7
		2	4	0.3
2	0	1	1	0.4
		2	3	0.2
		3	4	0.3
		4	9	0.1
3	0	1	2	0.2
		2	5	0.8

Table 2. Deterioration parameters for Example 1.

MAs can be scheduled before each of the three jobs in the sequence. A feasible solution for MPP-N is given by the binary vector $M^N = (1, 0, 1)$ representing the solution of scheduling two MAs, one before the first and one immediately before the third job. We now want to calculate the value of the objective function Z for this feasible solution.
As an example we want to calculate the probability $q_{1,4}$ which is among others needed to calculate the expected completion times (see (1)). Again, $q_{1,4}$ is the probability that the random variable $\tilde{L}_1$ will take the value 4 under application of the PMA schedule given by M^N. Since a PMA is performed right before job 1, the ML at the start of job 1 is certainly $l^{\max}$. The possible deterioration during the processing of job 1 is either $\delta_1^1 = 2$ or $\delta_1^2 = 4$. Deterioration δ_1^1 occurs with a probability of $q_1^1 = 0.7$ and reduces the ML by 2 units to a ML of 4. There is no other scenario resulting in $\tilde{L}_1 = 4$, and therefore, it holds that $q_{1,4} = 0.7$. Using analogous calculations we obtain the expected completion times

$$E[\tilde{C}_1] = 0 + 1 \cdot (l^{\max} - l_0) + 0 + E_1^{\text{ema}}(l^{\max}) \cdot d^{\text{ema}} = 2 + 0 \cdot 25 = 2,$$
$$E[\tilde{C}_2] = 2 + 0 + 0 + (0.7 \cdot E_2^{\text{ema}}(4) + 0.3 \cdot E_2^{\text{ema}}(2)) \cdot d^{\text{ema}} = 2 + (0.07 + 0.21) \cdot 25 = 9 \text{ and}$$
$$E[\tilde{C}_3] = 9 + 1 \cdot (l^{\max} - (0.09 \cdot (5 + 4) + 0.28 \cdot 3 + 0.33 \cdot 1 + 0.21 \cdot 0)) + 0 + 0 \cdot d^{\text{ema}} = 13.02.$$

They sum up to the objective value $Z = E[\sum_{j=1}^{3} \tilde{C}_j] = \sum_{j=1}^{3} E[\tilde{C}_j] = 24.02$.

An equivalent solution to M^N for the problem MPP-T is given by the 7×3 matrix M^T holding $(M^T)_{l,1} = 1$, $(M^T)_{l,2} = 0$ and $(M^T)_{l,3} = 1$ for all $l = 1, \ldots, l^{\max} + 1$.

2.2. Computational complexity

In the section at hand we show that both MPP-N and MPP-T are NP-hard. We start by restating Even-Odd-Partition which is known to be NP-complete, see Garey and Johnson (1979) and introducing a new problem Even-Odd-Counting.

Even-Odd-Partition: Given integers $a_1, \ldots, a_{2m}$ with $a_{k-1} < a_k$ for $k = 2, \ldots, 2m$ and with total value $2B$, is there a subset of m of these numbers with total value of B such that for each $k = 1, ..., m$ exactly one of the pair $\{a_{2k-1}, a_{2k}\}$ is in the subset?

Even-Odd-Counting: Given integers $a_1, \ldots, a_{2m}$ with $a_{k-1} < a_k$ for $k = 2, \ldots, 2m$ and with total value $2B$ and integer $\chi \leq 2^m$, are there at least χ distinct subsets with total value of B such that for each subset and each $k = 1, ..., m$ exactly one of the pair $\{a_{2k-1}, a_{2k}\}$ is in the subset?

Even-Odd-LE-Counting: Given integers $a_1, \ldots, a_{2m}$ with $a_{k-1} < a_k$ for $k = 2, \ldots, 2m$ and with total value $2B$ and integers $\psi \leq 2^m$ and $R \leq 2B$, are there at least ψ distinct subsets with total value of at most R such that for each subset and each $k = 1, ..., m$ exactly one of the pair $\{a_{2k-1}, a_{2k}\}$ is in the subset?

Note that restricting $\chi \leq 2^m$ and $\psi \leq 2^m$ is reasonable since the answer is obviously no otherwise.

Lemma 1. *Even-Odd-LE-Counting is NP-hard.*

Proof. The proof is by reduction from Even-Odd-Partition and proceeds as follows. First, we provide a Karp reduction from Even-Odd-Partition to Even-Odd-Counting. In a second step, we present a Cook reduction from Even-Odd-Counting to Even-Odd-LE-Counting.

For a given instance I of Even-Odd-Partition we consider an instance I' of Even-Odd-Counting with the same numbers and $\chi = 1$. The reduction is polynomial since $\log \chi \leq m$. Obviously, the answer to I is yes if and only if the answer to I' is yes. Hence, Even-Odd-Counting is NP-hard.

Now, consider an instance I' of Even-Odd-Counting. We consider the following algorithm solving Even-Odd-Counting and using calls of a subroutine solving Even-Odd-LE-Counting. We consider instances Even-Odd-LE-Counting having the same numbers $a_1, \ldots, a_{2m}$ and $R = B$ in a binary search. In the course of the search we vary the value of ψ which enables us to derive the exact number K_B of distinct subsets with

total value of at most $R = B$ (K_B is the largest value of ψ for which the subroutine solving Even-Odd-LE-Counting returns yes). Note that this requires $O(m)$ calls of the subroutine since $\psi \leq 2^m$. We repeat the above with $R = B - 1$ which yields the exact number K_{B-1} of distinct subsets with total value of at most $R = B - 1$. Again, $O(m)$ calls of the subroutine suffice. Now, $K_B - K_{B-1}$ is the exact number of distinct subsets with total value of exactly B. The answer to I' is yes if and only if $K_B - K_{B-1} \geq \chi$. The above algorithm runs in polynomial time if the subroutine runs in polynomial time and, thus, Even-Odd-LE-Counting is NP-hard unless $P = NP$. □

Now, we are prepared to show our first main result concerning MPP-N and MPP-N-DEC by reduction from Even-Odd-LE-Counting which is NP-hard due to Lemma 1.

Theorem 1. *MPP-N-DEC is NP-hard even if $K_j \leq 2$ for each $j \in J$.*

Proof. For the reduction we restrict ourselves to instances of Even-Odd-LE-Counting with $\psi > 0$, $\sum_{k=1}^{m-1} a_{2k} < R$, and $a_{2m} \leq R$. Hence, we should argue that this special case is NP-hard, as well. First, the answer to instances with $\psi \leq 0$ is trivially yes. Second, for an arbitrary instance we can add $2B$ to each number and replace R by $2mB + R$ and obtain an equivalent instance with

$$\sum_{k=1}^{m-1} (2B + a_{2k}) = 2(m-1)B + \sum_{k=1}^{m-1} a_{2k} = 2mB - 2B + \sum_{k=1}^{m-1} a_{2k} < 2mB \leq 2mB + R$$
$$\text{and } a_{2m} + 2B < 4B \leq 2mB \leq 2mB + R$$

In the following we, thus, assume that $\psi > 0$, $\sum_{k=1}^{m-1} a_{2k} < R$, and $a_{2m} \leq R$ for given instance I of Even-Odd-LE-Counting.

We construct an instance I' of MPP-N-DEC as follows. We have $n = m$ jobs. We set $p_j = 0$, $K_j = 2$, $\delta_j^1 = a_{2j-1}$, $\delta_j^2 = a_{2j}$, and $q_j^1 = q_j^2 = 0.5$ for each $j = 1, \ldots, m$. Furthermore, we set $l^{\max} = d^{\text{ema}} = (2^{m+1} + 1) \cdot R$, $l_0 = R$, and $\Gamma = (2^m - \psi)/2^m \cdot l^{\max}$.

We claim that the answer to I' is yes if and only if the answer to I is yes.

Suppose a solution M to I' with a total expected completion time of no more than $\Gamma = (2^m - \psi)/2^m \cdot l^{\max}$ exists.

Having a PMA before job j raises the expected completion time of $n + 1 - j$ jobs by at least $l^{\max} - l_0 = 2^{m+1} \cdot R$. Hence, the total expected completion time for a solution with a PMA is at least

$$\begin{aligned} & 2^{m+1} \cdot R = l^{\max} - R = l^{\max} - l^{\max}/(2^{m+1} + 1) \\ > \; & l^{\max} - l^{\max}/2^m = (2^m - 1)/2^m \cdot l^{\max} \geq (2^m - \psi)/2^m \cdot l^{\max} = \Gamma. \end{aligned}$$

Hence, M does not have a PMA.

Note that each subset of numbers in I directly corresponds to a materialization of deteriorations of all jobs, each with a probability of $1/2^m$. No such materialization causes an EMA before the start of job n since $\sum_{k=1}^{m-1} a_{2k} < R$ and, thus, only job n can have a positive expected completion time. If it has a positive completion time it has completion time $d^{\text{ema}} = l^{\max}$ since $\delta_n^2 = a_{2m} \leq R < l^{\max}$ and, thus, job n cannot cause more than one EMA.

Now it is not hard to see that the total expected completion time does not exceed $(2^m - \psi)/2^m \cdot l^{\max}$ if and only if there are at least ψ distinct subsets with total value of at most R.

Assume now that I is a yes-instance. It follows from the above that the minimum total expected completion time for I' does not exceed $\Gamma = (2^m - \psi)/2^m \cdot l^{\max}$. □

Our second main result concerns MPP-T and MPP-T-DEC. Again, we use a reduction from Even-Odd-LE-Counting which is NP-hard due to Lemma 1.

Theorem 2. *MPP-T-DEC is NP-hard even if $K_j \leq 2$ for each $j \in J$.*

Proof. Given an instance I of Even-Odd-LE-Counting with $\psi > 0$ and $\sum_{k=1}^{m-1} a_{2k} < R$ (as justified in the proof of Theorem 1) we construct an instance I' of MPP-T-DEC as follows.

We have $n = m + 1$ jobs. We set $p_1 = 0$, $K_1 = 1$, $\delta_1^1 = 2^{m+1} \cdot R$, and $q_1^1 = 1$ and $p_j = 0$, $K_j = 2$, $\delta_j^1 = a_{2j-1}$, $\delta_j^2 = a_{2j}$, and $q_j^1 = q_j^2 = 0.5$ for each $j = 2, \ldots, m$ and $p_{m+1} = 0$, $K_{m+1} = 1$, $\delta_{m+1}^1 = 2B - R$, and $q_{m+1}^1 = 1$. Furthermore, we set $l^{\max} = (2^{m+1} + 1) \cdot 2B$, $l_0 = 2B$, $d^{\text{ema}} = 2 \cdot l^{\max}$ and $\Gamma = (2^{n-1} - \psi)/2^{n-1} \cdot l^{\max}$.

We claim that the answer to I' is yes if and only if the answer to I is yes.

Suppose a solution M to I' with a total expected completion time of no more than Γ exists. Note that having a PMA before job j, $j \leq n - 1$, raises the expected completion time of $n + 1 - j$ jobs by at least $l^{\max} - l_0$. If the PMA is delayed by one job irrespective of the actual deterioration caused by j the expected completion time of $n - j$ jobs is increased by at most $l^{\max}$ instead. Since

$$\begin{aligned} & (n+1-j) \cdot (l^{\max} - l_0) = (n+1-j) \cdot 2^{m+1} \cdot 2B = (n-j) \cdot 2^{m+1} \cdot 2B + 2^{m+1} \cdot 2B \\ > \; & (n-j) \cdot 2^{m+1} \cdot 2B + 2B = (n-j) \cdot l^{\max} \end{aligned}$$

delaying the PMA decreases the objective value. Note that j cannot cause an EMA since $l_0 = 2B$ and delaying the PMA cannot lead to longer MAs later on. Hence, there is a solution M' to I' with a total expected completion time of no more than Γ such that PMAs are conducted only immediately before job n. We refer to such a solution M with

$m_{l,j} = 0$ for each $j < n$ and $l \in \{0, \ldots, l^{\max}\}$ as delayed and restrict ourselves to delayed solutions in the following.

Consider a delayed solution with a total expected completion time of no more than Γ. Note that EMAs can only be triggered by job n and since job n has a certain deterioration of δ_n^1 we can prevent them from becoming due once we know $\tilde{L}_{n-1}$. Also, since $d^{\text{ema}} = 2 \cdot l^{\max}$ it cannot increase the total expected completion time if we have a PMA in case of $\tilde{L}_{n-1} < \delta_n^1$ instead of accepting a certain EMA. Hence, if there is a delayed solution with a total expected completion time of no more than Γ, then the delayed solution with $m_{l,n} = 1$ if and only if $l < \delta_n^1 = 2B - R$ for each $l \in \{0, \ldots, l^{\max}\}$ has a total expected completion time of no more than Γ. We refer to this solution as the dominant solution.

Note that each subset of numbers in I directly corresponds to a materialization of deteriorations of jobs $1, \ldots, n-1$, each with a probability of $1/2^{n-1}$. The dominant solution has a PMA before the start of job n for such a materialization if its total deterioration is above R since then $\tilde{L}_{n-1} < l_0 - R = 2B - R = \delta_n^1$. So, the probability that a PMA is arranged is $\Delta^{>R}/2^{n-1}$ where $\Delta^{>R}$ is the number of subsets in I with a total value exceeding R. The PMA's duration is at least $l^{\max} - l_0$ and at most $l^{\max}$ and, thus, the corresponding total expected completion time is at least $\Delta^{>R}/2^{n-1} \cdot (l^{\max} - l_0)$ and at most $\Delta^{>R}/2^{n-1} \cdot l^{\max}$ (since job n cannot trigger an EMA after starting at ML $l^{\max}$ since $\delta_n^1 \leq l^{\max}$). Note that for $\Delta^{>R} < 2^m$ we have

$$\begin{aligned}
&(\Delta^{>R} + 1)/2^{n-1} \cdot (l^{\max} - l_0) = \Delta^{>R}/2^{n-1} \cdot l^{\max} + 1/2^{n-1} \cdot l^{\max} - (\Delta^{>R} + 1)/2^{n-1} \cdot l_0) \\
\geq\ & \Delta^{>R}/2^{n-1} \cdot l^{\max} + 1/2^{n-1} \cdot l^{\max} - l_0 = \Delta^{>R}/2^{n-1} \cdot l^{\max} + 1/2^{n-1} \cdot ((2^n + 1) \cdot l_0) - l_0 \\
>\ & \Delta^{>R}/2^{n-1} \cdot l^{\max} + l_0 - l_0 > \Delta^{>R}/2^{n-1} \cdot l^{\max}
\end{aligned}$$

and, thus, the total expected completion time is at most $\Gamma = (2^{n-1} - \psi)/2^{n-1} \cdot l^{\max}$ if and only if $\Delta^{>R} \geq \psi$.

Assume now that I is a yes-instance. It follows from the above that the minimum total expected completion time for I' does not exceed $\Gamma = (2^{n-1} - \chi')/2^{n-1} \cdot l^{\max}$. □

2.3. On the relation between transparent and non-transparent machines

In this section we analyze the ratio between the minimum total expected completion times $Z^N(I)$ and $Z^T(I)$ of MPP-N and MPP-T respectively, for the same instance I. The ratio $Z^T(I)/Z^N(I)$ measures how much we can decrease the objective value by employing a transparent machine rather than a non-transparent one. We denote the set of all instances by $\mathcal{I}$.

Corollary 1. *$Z^N(I) \geq Z^T(I)$ for each instance $I \in \mathcal{I}$.*

The corollary follows directly from the fact that each solution for I according to MPP-N is a solution for I according to MPP-T.

Lemma 2. $\inf\{Z^T(I)/Z^N(I) \mid I \in \mathcal{I}\} = 0$.

Proof. We prove the lemma by a constructive argument and consider instance I_q, $q \in [0,1]$, which is defined as follows. We have $J = \{1,2\}$ (with $p_1 = p_2 = 0$) and $l_0 = l^{\max} = 2$. The deteriorations of the jobs are given by $\delta_1^1 = l^{\max}$, $\delta_1^2 = 1$ with $q_1^1 = q$, $q_1^2 = 1-q$ and $\delta_2^1 = 1$ with $q_2^1 = 1$. Finally, the duration of an EMA is specified as $d^{\text{ema}} = l^{\max} + \frac{1}{q}$.

It can be easily seen that the first job does not cause an EMA and $\tilde{L}_1$ takes value $l^{\max} - 1$ with probability $1-q$ and value 0 with probability q.

We derive an optimum solution M^N to I_q according to MPP-N first. Clearly, we can choose $m_1^N = 0$ since $l_0 = l^{\max}$ and the completion time of job 1 is 0. If we choose $m_2^N = 1$, both the expected duration of the PMA and the total expected completion time is $q \cdot l^{\max} + (1-q) \cdot 1$. If we choose $m_2^N = 0$, an EMA is due with probability q and the total expected completion time is $q \cdot d^{\text{ema}} = q \cdot l^{\max} + 1$. Thus, $M^N = (0,1)$ is an optimum solution and $Z^N(I_q) = q \cdot l^{\max} + 1 - q$.

Next, we consider an optimum solution M^T to I_q according to MPP-T. It is not hard to see that we prevent EMAs in any optimum solution and can choose $m_{l,j}^T = 0$ if $l > 0$ or $j = 1$ and $m_{0,2}^T = 1$ since

- a MA is due if and only if $\tilde{L}_1 = 0$ and
- we prefer to have a PMA rather than an EMA since $l^{\max} < d^{\text{ema}}$.

The minimum total expected completion time, thus, is $Z^T(I_q) = q \cdot l^{\max}$.

Concluding, we obtain a ratio of

$$\frac{Z^T(I_q)}{Z^N(I_q)} = \frac{q \cdot l^{\max}}{q \cdot l^{\max} + 1 - q} = \frac{l^{\max}}{l^{\max} + \frac{1}{q} - 1}. \tag{3}$$

which approaches 0 if q goes to zero. □

Lemma 2 states that there are instances so that the relative advantage of the transparent machine is arbitrarily high by employing a transparent machine rather than a non-transparent one.

3. Solution approach for MPP-N

We develop a solution approach for MPP-N based on dynamic programming. First, we will propose the structure of the *dynamic program* (DP), that is the set of states, the set of transitions, and the Bellman function (depending on, among others, costs associated with transitions), in Section 3.1. Note that a deterministic DP, where some input values are expectations, suffices. Then, in Section 3.2, we will develop a pseudo-polynomial time approach to determine the transitions' costs in this DP.

3.1. Deterministic dynamic program

The basic idea of the DP is to see a solution as a sequence of batches. A *batch* is a maximum sequence of jobs so that no PMA is planned in between these jobs. A solution M corresponds to the set of batches where job 1 is a first job in a batch and job $j > 1$ is a first job in a batch if and only if $m_j = 1$. Thus, if $m_j = m_{j'} = 1$ with $j' > j$ and $m_{j''} = 0$ for each $j < j'' < j'$, then jobs j to $j' - 1$ form a batch. We let $\mathcal{B}^m = (B_1^m, \ldots, B_{r^m}^m)$ be the sequence of $r^m \in \mathbb{N}$ batches implied by M. Additionally to the batch sequence $\mathcal{B}^m$ the information if there is a PMA planned before the first job is needed for the full solution information.

For a solution M with f^r and l^r being the first and the last job in batch B_r^m, $r = 1, \ldots, r^m$, we can rewrite the objective value as

$$\begin{aligned} Z = \sum_{j=1}^{n} E[\tilde{C}_j] &= \sum_{r=1}^{r^m} \sum_{j=f^r}^{l^r} E[\tilde{C}_j] \\ &= \underbrace{m_1 \cdot n \cdot (l^{\max} - l_0)}_{\text{delay of jobs by PMA before job 1}} + \sum_{r=1}^{r^m} ((n - l^r) \cdot E[\tilde{H}_r] + \sum_{j=f^r}^{l^r} E[\tilde{C}'_j]). \quad (4) \end{aligned}$$

Here, $\tilde{H}_r$ is the random variable representing the duration of batch B_r^m including EMAs and a following PMA (if there is one). Note that $\tilde{H}_{r^m}$ does not influence the objective value since $n - l^r = 0$. For the other batches $\tilde{H}_r$ is the length of the timespan between start times of jobs f^r and f^{r+1}. Note, furthermore, that $\tilde{H}_1$ depends on whether or not an PMA is planned before job 1. Finally, the random variable corresponding to the difference between the start time of the first job in j's batch and j's completion time is given as $\tilde{C}'_j$. The formulation in (4), then, allows to identify, first, the delay of all jobs caused by an initial PMA and, second,

$$\Omega(f^r, l^r) := \left((n - l^r) \cdot E[\tilde{H}_r] + \sum_{j=f^r}^{l^r} E[\tilde{C}'_j] \right) \quad (5)$$

as the additive contribution of batch B_r^m to the objective value. The first part of (5) reflects the delay of all jobs in following batches due to the duration of batch B_r^m including EMAs and a following PMA while the second part accounts for the delay of completion of each job in batch B_r^m after starting the first job in the batch. Note that it is open at the moment how to determine $E[\tilde{H}_r]$ and $E[\tilde{C}'_j]$. We will detail this in Section 3.2.

Now we are ready to state the DP. Slightly abusing notation, we denote states by $j \in \{0, \ldots, n+1\}$, i.e. the state space encompasses a state for each job, an initial state and a terminal state. Being in state j, $0 < j < n+1$ means all decisions about PMAs prior to the processing of job j are already made, and job j is the first job in a batch, i.e. there is a PMA immediately before j. State 0 is an initial state reflecting the situation where no decision has been made yet while state $n+1$ is a terminal state where all batches are determined. We have a transition from each state j to each state j' with $j < j'$. A transition from j to j' corresponds to the decision to schedule the next PMA after job j immediately before j', that is, jobs j to $j'-1$ form a batch (i.e. $m_j = m_{j'} = 1, m_{j''} = 0, j < j'' < j'$).

Note that each path from the initial state 0 to the terminal state $n+1$ represents a solution M of the problem. To obtain a fully specified DP, we must assign costs to the transitions such that the cost of each path equals the objective value of the corresponding solution. The intuitive way to do so is to assign to each transition the incremental change in objective value that corresponds to the decision represented by this transition. The additive separation of the objective in Equation (4) provides the blueprint to do so. The expected cost $c(j, j')$ of the transition from j to j' equals

$$c(j,j') \quad = \quad \begin{cases} n \cdot (l^{\max} - l_0), & j = 0,\ j' = 1 \\ \Omega(1, j'), & j = 0,\ j' > 1 \\ \Omega(j, j'), & j > 0,\ j' > j. \end{cases} \tag{6}$$

The first case in (6) reflects the duration of a PMA before job $j = 1$. The other two cases reflect the delay of all following jobs of a corresponding batch consisting of job 1 to $j'-1$ if $j = 0$ and j to $j'-1$ if $j > 0$ plus the total completion time of all jobs in the batch. We say that each state j has value $d(j)$, which is the expected cost of the cheapest path from the initial state to j, i.e. the minimum contribution to the objective caused by partitioning jobs 1 to $j-1$ into batches.

We set $d(0) = 0$ and for all other states we formulate the following Bellman equation

$$d(j) = \min \left\{ d\left(j'\right) + c(j', j) | 0 \leq j' < j \right\}. \tag{7}$$

The problem then is to determine $d(n+1)$. The sequence of transitions implying $d(n+1)$ represents the decisions about a PMA before job 1 and a partition of the jobs into batches.

Regarding computational effort, we have $O(n)$ states and $O(n^2)$ transitions. Once all

transitions' costs are determined we can, thus, determine an optimum solution in $O(n^2)$. We will see in Section 3.2 that transitions' costs can be determined in pseudo-polynomial time which renders our approach to run in pseudo-polynomial time.

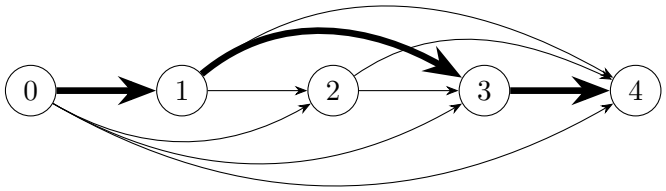


Figure 1. DP with solution $M^N = (1, 0, 1)$.

Example 2. We consider again instance I introduced in Example 1. The given instance parameters define the DP visualized by the network in Figure 1.

The bold printed transitions correspond to the feasible solution $M^N = (1, 0, 1)$. Remember, this solution implies the batch configuration $\mathcal{B}^m = (\{1, 2\}, \{3\})$ with a PMA planned before job 1. For the determination of the optimal solution the transition costs are needed, whose calculation is illustrated in Section 3.2.

3.2. Calculating transition costs

In the section at hand we address how to derive the expected cost $c(j, j')$ of a transition from state j to state j'. As outlined in Section 3.1 this is trivial for $c(0, 1) = n \cdot (l^{\max} - l_0)$ and, thus, we focus on the case with $j' > 1$.

To do so, we need to capture how the machine's ML evolves during the processing of jobs j to $j' - 1$ because of deteriorations and what jobs are delayed by the resulting EMAs. Again, we use dynamic programming and define a – now stochastic – DP (not to be confused with the DP in Section 3.1). More technically, the expected cost is calculated using the following lattice containing only stochastic nodes without any decision nodes. We consider a state (i, l) for each $i = \max\{j, 1\}, \ldots, j'$ and $l = 0, \ldots, l^{\max}$. State (i, l) represents the machine having a ML of l when completing job i. Additionally, we have an initial state $(\max\{j, 1\} - 1, \bar{l})$ and a terminal state $(j' + 1, l^{\max})$. Here, $\bar{l}$ is the ML before starting the processing of jobs in the batch and equals l_0 if $j = 0$ and $l^{\max}$ otherwise.

We have a transition from each state (i, l), $i < j'$, to each state $(i + 1, l')$. With each such transition we associate

- a transition probability $q(i, l, l')$ with

$$q(i, l, l') = \sum_{\substack{k=1,\ldots,K_{i+1}: \\ l'=(l-\delta^k_{i+1}) \mod l^{\max} \wedge (l-\delta^k_{i+1}) \neq l^{\max}}} q^k_{i+1} \qquad \text{if } l' < l^{\max} \text{ and}$$

$$q(i,l,l') = \sum_{k=1,\ldots,K_{i+1}: l^{\max} = l - \delta^k_{i+1}} q^k_{i+1} \qquad \text{if } l' = l^{\max} \text{ and}$$

- an expected duration $d(i,l,l')$ with

$$d(i,l,l') = \frac{1}{q(i,l,l')} \sum_{\substack{k=1,\ldots,K_{i+1}: (l'=l=l^{\max} \wedge \delta^k_{i+1}=0) \vee \\ (l' = (l-\delta^k_{i+1}) \mod l^{\max} \wedge (l-\delta^k_{i+1}) \neq l^{\max})}} q^k_{i+1} \cdot \max\{0, \lceil (\delta^k_{i+1} - l)/l^{\max} \rceil \cdot d^{\text{ema}}\},$$

  that is the expected total duration of EMAs during the processing of $i+1$ given again that the ML was l when completing i and is l' when completing $i+1$.

The definition of $q(i,l,l')$ distinguishes between $l' < l^{\max}$ and $l' = l^{\max}$. Note that we can reach $l' = l^{\max}$ with positive probability only if $l = l^{\max}$. The definition of $d(i,l,l')$ integrates both cases.

Furthermore, we have a transition from each state (j',l) to the terminal state $(j'+1, l^{\max})$. With such a transition we associate transition probability of $q(j',l,l^{\max}) = 1$ and an expected duration of $d(j',l,l^{\max}) = l^{\max} - l$ reflecting that a PMA with duration $l^{\max} - l$ follows at the end of the batch. Recall that in the case of the last batch this duration does not delay any job and, hence, we can assume a PMA to follow the last batch, as well.

The value of each state (i,l) is a tuple consisting of

- the probability $q(i,l)$ that it is reached conditional on the initial state and
- the expected contribution $\Omega^X(i,l)$ of jobs $j,\ldots,i$ to $\Omega(j,j')$ conditioned on the ML after job i is l.

For the initial state (root node), we have $(\max\{j,1\}-1, \bar{l})$ with $q(\max\{j,1\}-1,\bar{l}) = 1$ and $\Omega^X(\max\{j,1\}-1,\bar{l}) = 0$.

For each state (i,l), $\max\{j,1\} - 1 < i \leq j'$, we have recursions

- $q(i,l) = \sum_{l'=0}^{l^{\max}} q(i-1,l') \cdot q(i-1,l',l)$ and
- $\Omega^X(i,l) = 1/q(i,l) \sum_{l'=0}^{l^{\max}} (\Omega^X(i-1,l') + (n-i+1) \cdot d(i-1,l',l)) \cdot q(i-1,l') \cdot q(i-1,l',l).$

Note that by considering the impact on the completion time of all jobs that follow $i-1$ in $\Omega^X(i,l)$ we account for both parts in (5), the delay of jobs in the next batch and the delay of jobs $i,\ldots,j'$ in the batch under consideration. For the terminal state $(j'+1, l^{\max})$ we have

- $q(j'+1, l^{\max}) = \sum_{l'=0}^{l^{\max}} q(j',l') = 1$ and
- $\Omega^X(j'+1, l^{\max}) = \sum_{l'=0}^{l^{\max}} q(j',l') \cdot (\Omega^X(j',l') + (n-j') \cdot (l^{\max} - l')).$

Thus, we have $c(j, j') = \Omega(j, j') = \Omega^X(j' + 1, l^{\max})$. We have $O(n \cdot l^{\max})$ states and $O(n \cdot (l^{\max})^2)$ transitions in our DP and, thus, we derive $c(j, j')$ in pseudo-polynomial time. Complementing Theorem 1 the complexity status of MPP-N-DEC is settled.

Corollary 2. *MPP-N-DEC is binary NP-hard.*

In the generic construction given above we have a computational overhead as there are probably states and transitions which are never reached. In order to decrease the computational burden the states and transitions starting from them are generated in increasing job-indices which allows to avoid states that are not actually reached. Furthermore, it is not hard to see that we can derive cost $c(j, j')$ for all $j' > j$ by considering the DP for $c(j, n + 1)$ only since the latter contains all states and transitions of the DP for $c(j, j')$ (except for the final states). The overall run time complexity for yielding all transitions' costs is, then, $O(n^2 \cdot (l^{max})^2)$.

Example 3. We consider again instance I from Example 1. Exemplarily, we calculate the transition cost $c(1, 3)$, that is, the expected contribution to the objective of the batch consisting of job 1 and 2 with a scheduled PMA before job 1 (without this PMA before job 1, it would be denoted by $c(0, 3)$). For the calculation, we set up the DP in Figure 2.

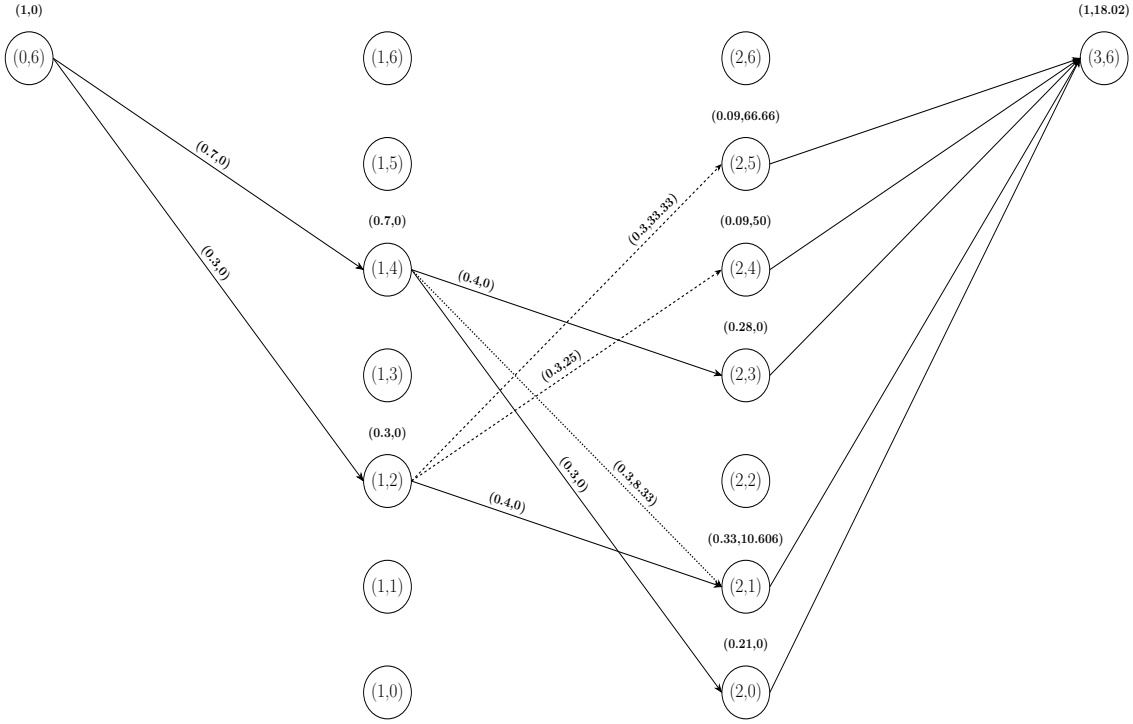


Figure 2. DP for transition costs $c(1, 3)$.

For readability reasons, only the transitions with a probability greater than zero are illustrated, as only these are relevant for the calculations. Please note that for a given state,

different realizations of deteriorations may bring the system to an identical next state, albeit with different numbers of EMAs. Thus, a transition may cover zero, one, or multiple EMAs. Accordingly, transitions that only reflect deteriorations triggering at least one EMA are depicted as dashed lines. Transitions where at least one deterioration triggers EMAs and at least one does not, are represented as dotted lines. Transitions without EMAs are drawn as continuous lines.

To make the DP setup clearer, we elaborate on the transitions between stage 0 and stage 1. The initial ML corresponds to $l^{\max} = 6$ and the processing of the first job reduces the ML by either 2 or 4. This results in the transitions between the states $(0, 6)$ and $(1, 6-2) = (1, 4)$ respectively $(1, 6-4) = (1, 2)$ with the transition probabilities $q(0, 6, 4) = q_1^1 = 0.7$ and $q(0, 6, 2) = q_1^2 = 0.3$. The deterioration does not exceed the ML in both cases and, thus, the expected durations are $d(0, 6, 4) = d(0, 6, 2) = 0$.

In Figure 2 the transition probabilities and durations are noted as tuples at the transitions. These numbers build the basis for the calculation of the state values which are noted as tuples above the states.

As an example, let us calculate the value of the state $(2, 1)$. The state $(2, 1)$ symbolizes that the second job was executed last and that the present ML is 1. This state can be reached in different ways, which correspond to the incoming edges of $(2, 1)$. Either the ML was at level 4 before the processing of job 2 started and a deterioration of 3 or 9 occurred, or a ML of 2 was decreased by a deterioration of 1. The probability to end up in this state sums up to $q(2, 1) = 0.33$.

An EMA occurs if the previous ML was 4 and the deterioration is 9. We have the transition duration $d(1, 4, 1) = 25/3$ which results in the value

$$\begin{aligned}\Omega^X(2,1) &= 1/q(2,1)\Big((\Omega^X(1,4) + (n-2+1)\cdot d(1,4,1))\cdot q(1,4)\cdot q(1,4,1) \\ &\quad + (\Omega^X(1,2) + (n-2+1)\cdot d(1,2,1))\cdot q(1,2)\cdot q(1,2,1)\Big) \\ &= 1/0.33\cdot((0+(3-2+1)\cdot 25/3)\cdot 0.7\cdot 0.3 + (0+2\cdot 0)\cdot 0.3\cdot 0.1) = 10.606.\end{aligned}$$

These values can be interpreted as follows. The probability that the maintenance level after job 2 equals 1 is 0.33. Then, the expected contribution to the objective function (due to EMAs during jobs 1 to 2) is 10.606. The cost of transition $(1, 3)$ is derived as $c(1, 3) = \Omega^X(3, 6) = 18.02$.

4. Solution approach for MPP-T

We develop a solution approach for MPP-T based on a stochastic DP with $n+1$ stages where each stage $j = 1, \ldots, n$ corresponds to job j in the sequence and stage $n+1$ is the terminal stage. Each stage except the terminal stage consists of a set of states (or decision nodes) in which decisions concerning PMAs need to be made and stochastic nodes representing the stochastic deteriorations caused by jobs. In total, stage $j = 1, \ldots, n$ represents the evolution of the machine's ML between the completion of job $j-1$ (the start of the planning horizon if $j = 1$) and the completion of job j. This evolution can be divided into the decision regarding a PMA before job j (outgoing transitions from states) and the deterioration caused by job j which potentially triggers EMAs (outgoing transitions from stochastic nodes).

The set of states in stage j, $j = 2, \ldots, n+1$, is given by $D_j := \{[j,l] | l \in \{0, \ldots, l^{\max}\}\}$. State $[j,l]$ represents the machine having an ML of l when completing job $j-1$ (before a potential PMA in between $j-1$ and j if $j < n+1$). The only state $[1, l_0]$ in stage $j = 1$ represents the machine having the initial maintenance level l_0.

The set of stochastic nodes in stage j, $j = 1, \ldots, n$, is given by $S_j := \{(j,l) | l \in \{0, \ldots, l^{\max}\}\}$. Thereby, the stochastic node (j,l) represents the machine having an ML of l right before the processing of job j where new information arrives as the random deterioration realizes (i.e. after a potential PMA immediately before job j). There are no stochastic nodes in stage $n+1$.

Transitions exist exclusively from states to stochastic nodes of the same stage and from stochastic nodes to states of the subsequent stage.

- We have a transition from state $[j,l]$ to stochastic node (j,l') if and only if $l' = l$ or $l' = l^{\max}$. If $l' = l$, the transition from $[j,l]$ to (j,l'), represents that no PMA occurs in between jobs $j-1$ and j if $j > 1$ and before job 1 if $j = 1$. If $l' = l^{\max}$, the transition from $[j,l]$ to (j,l') represents that a PMA occurs in between jobs $j-1$ and j if $j > 1$ and before job 1 if $j = 1$. Note that both cases coincide if $l = l^{\max}$ and we are free to interpret the transition either as having a PMA (with duration 0) or as having no PMA (resulting in an ML of $l^{\max}$, nevertheless).
- There is a transition from stochastic node (j,l) to state $[j+1, l']$ if and only if
  - $l = l' = l^{\max}$ and there is a $k = 1, \ldots, K_j$ with $\delta_j^k = 0$ or
  - $l' < l^{\max}$ and $l' = (l - \delta_j^k) \bmod l^{\max}$ for at least one $k = 1, \ldots, K_j$.

  The transition represents the ML to be l' when completing j while it is l when starting j. Note that not only deterioration but also EMAs may influence the ML while processing j. Note, furthermore, that different deteriorations caused by j are represented by the same transition if they differ by an integer multiple of $l^{\max}$ since

the ML is the same when completing j (but different numbers of EMAs have been carried out).

In the next step we would like to specify the stage-wise contribution function of the DP. This specifies the additive contribution to the objective function of a stage in the DP, i.e. of a job. For this we consider transitions between two states. A transition between the two states $[j,l]$ and $[j+1,l']$ is a concatenation of two transitions of the just introduced transition types which are defined by the decision $m_{l,j}$ made in the state $[j,l]$ and the actually occurred realization of the stochastic deterioration leading to state $[j+1,l']$.

In order to identify an additive contribution to the objective value by PMAs or EMAs we, first, reformulate the objective function. Here, random variable $\tilde{B}_j$, $j=1,\ldots,n$, represents the time that elapses between the completion of job $j-1$ and the completion of job j, that is $\tilde{B}_j = \tilde{C}_j - \tilde{C}_{j-1}$ if $j>1$ and $\tilde{B}_1 = \tilde{C}_1$. Furthermore, random variable $\tilde{B}_{j,l,m_{l,j}}$ also represents the time between the completion of $j-1$ and j, but given that $\tilde{L}_{j-1}=l$ and the decision is $m_{l,j}$. Now consider the following reformulation of the objective function.

$$
\begin{aligned}
Z &= \sum_{j=1}^{n} E[\tilde{C}_j] = \sum_{j=1}^{n}\sum_{j'=1}^{j} E[\tilde{B}_{j'}] = \sum_{j'=1}^{n} \underbrace{(n-j'+1)\cdot E[\tilde{B}_{j'}]}_{\text{contribution of job } j' \text{ to objective value}} \\
&= \sum_{j'=1}^{n} (n-j'+1) \sum_{l=0}^{l^{\max}} q_{j'-1,l} \cdot E[\tilde{B}_{j',l,m_{l,j'}}] \\
&= \sum_{j'=1}^{n} \underbrace{\sum_{l=0}^{l^{\max}} q_{j'-1,l} \cdot \underbrace{(n-j'+1)\cdot E[\tilde{B}_{j',l,m_{l,j'}}]}_{\substack{\text{contribution of job } j' \text{ to objective} \\ \text{value if } \tilde{L}_{j'-1}=l \text{ dependent on } m_{j,l}}}}_{\text{contribution of job } j' \text{ to objective value dependent on } m_{j,l}}
\end{aligned}
$$

Note that $q_{j'-1,l}$ is the probability that the ML when completing job $j'-1$ is l conditional on a solution M. The reformulation allows us to express the objective value as the sum of independent additive contributions of jobs dependent on the solution M at hand. This contribution can be interpreted as the total expected delay of jobs $j',\ldots,n$ caused by EMAs while processing job j' and (possibly) a PMA immediately before processing job j'.

It remains to detail $E[\tilde{B}_{j,l,m_{l,j}}]$ which can be derived as

$$E[\tilde{B}_{j,l,m_{l,j}}] = m_{l,j} \cdot \underbrace{\left(\underbrace{(l^{\max} - l)}_{\text{duration of PMA}} + \underbrace{d^{\text{ema}} \cdot \underbrace{\sum_{k=1}^{K_j} q_j^k \cdot \max\{\lceil (\delta_j^k - l^{\max})/l^{\max} \rceil, 0\}}_{\text{expected number of EMAs if } m_{l,j}=1}}_{\text{expected total duration of EMAs if } m_{l,j}=1} \right)}_{E[\tilde{C}_j - \tilde{C}_{j,1}] \text{ if } \tilde{L}_{j-1}=l \text{ and } m_{l,j}=1}$$
$$+(1 - m_{l,j}) \cdot \underbrace{\left(d^{\text{ema}} \cdot \underbrace{\sum_{k=1}^{K_j} q_j^k \cdot \max\{\lceil (\delta_j^k - l)/l^{\max} \rceil, 0\}}_{\text{expected number of EMAs if } \tilde{L}_{j-1}=l \text{ and } m_{l,j}=0} \right)}_{\substack{\text{expected total duration of EMAs if } \tilde{L}_{j-1}=l \text{ and } m_{l,j}=0 \\ E[\tilde{C}_j - \tilde{C}_{j,1}] \text{ if } \tilde{L}_{j-1}=l \text{ and } m_{l,j}=0}}. \quad (8)$$

Note that since $m_{l,j} \in \{0, 1\}$ only one of the summands can have positive value. If $m_{l,j} = 1$, then the expected timespan between completion of job $j-1$ and completion of job j is composed of the duration of a PMA and the total duration of potential EMAs. If $m_{l,j} = 0$, only potential EMAs are accounted for. From (8) we can identify the stagewise contribution in our DP:

- The stagewise expected contribution from $[j, l]$ which corresponds to $m_{l,j} = 0$ equals $(n - j + 1) \cdot E[\tilde{B}_{j,l,0}]$.
- The stagewise expected contribution from $[j, l]$ which corresponds to $m_{l,j} = 1$ equals $(n - j + 1) \cdot E[\tilde{B}_{j,l,1}]$.

It remains to define the Bellman equation $d(j, l)$ evaluating each state $[j, l] \in D_j$, $j = 1, \ldots, n+1$, by the minimum sum of total timespan between completion of job $j-1$ and completions of jobs $j, \ldots, n$ given that the ML is l when completing $j-1$. This total timespan is due to PMAs and EMAs some time after the completion of $j-1$. We set $d(n+1, l) = 0$ for each state $[n+1, l] \in D_{n+1}$ and formulate the Bellman equation

$$d(j, l) = \min \left\{ (n - j + 1) E[\tilde{B}_{j,l,m_{l,j}}] + \sum_{k=1}^{K_j} q_j^k \cdot d\left(j + 1, \hat{l}(j, l, m_{l,j}, k)\right) \;\middle|\; m_{l,j} \in \{0, 1\} \right\} \quad (9)$$

for each state $[j, l] \in D_j$, $j = 1, \ldots, n$, where $\hat{l}(j, l, m_{l,j}, k)$ is the ML when completing j given $m_{l,j}$, an ML of l when completing $j-1$, and an actual deterioration of δ_j^k of job j.

Intuitively speaking, choosing $m_{l,j} = 0$ corresponds to choosing the outgoing transition of $[j, l]$ reflecting no PMA between $j-1$ and j while $m_{l,j} = 1$ corresponds to choosing the outgoing transition of $[j, l]$ reflecting a PMA.

An optimum solution to MPP-T-DEC can then be derived as the optimum choice of $m_{l,j}$ according to (9) for each state $[j,l] \in D_j$, $j = 1,\dots,n$ with the optimal objective value $Z = d(1,l_0)$. We have $O(n \cdot l^{\max})$ states, $O(n \cdot l^{\max})$ stochastic nodes, and $O(n \cdot (l^{\max})^2)$ transitions in our DP and, thus, we can evaluate $d(j,l)$ for each state $[j,l] \in D_j$, $j = 1,\dots,n$, in pseudo-polynomial time. Thus, complementing Theorem 2 the complexity status of MPP-T-DEC is settled.

Corollary 3. *MPP-T-DEC is binary NP-hard.*

Example 4. We consider again the instance introduced in Example 1. The job sequence consists of $n = 3$ jobs and we therefore have $n + 1 = 4$ stages including the terminal stage.

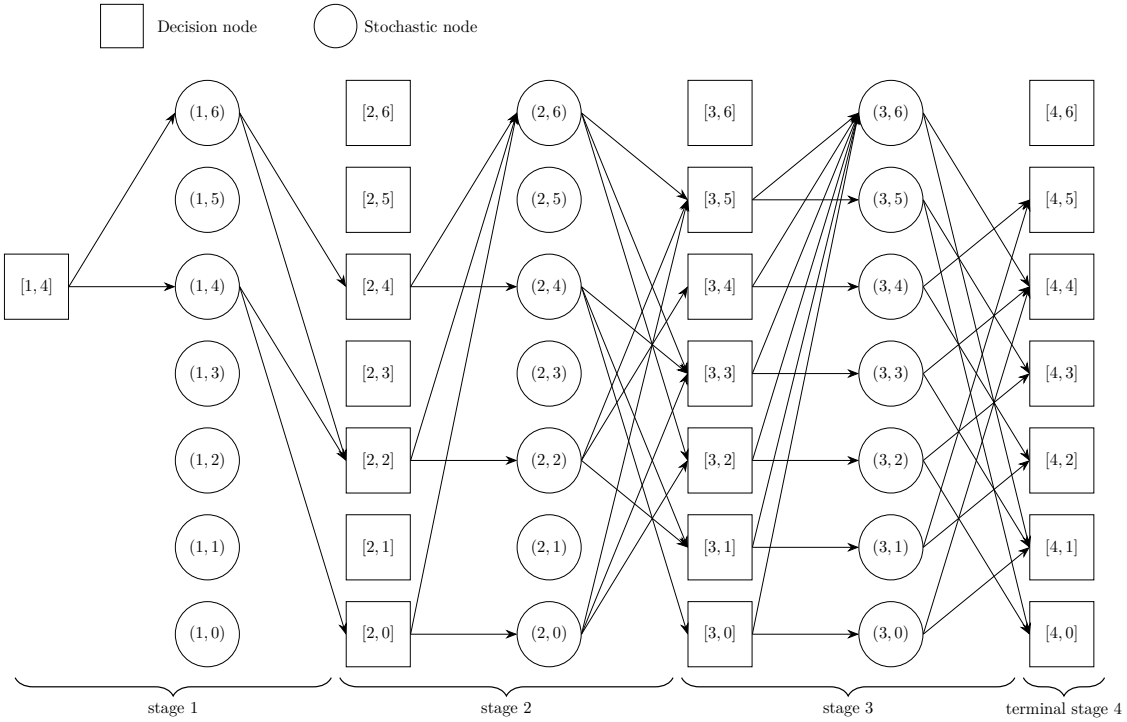


Figure 3. DP.

Figure 3 shows the DP as a network with squares, circles, and arrows representing states, stochastic nodes, and transitions. The network is reduced to transitions from states and nodes which can be actually reached directly or indirectly from state $[1,4]$. E. g., since there is no transition from $[1,4]$ to $(1,3)$ we do not depict transitions starting from $(1,3)$. State $[2,4]$ represents the ML to be 4 when completing job $j = 1$. The two transitions from $[2,4]$ to $(2,6)$ and from $[2,4]$ to $(2,4)$ represent a PMA to be carried out in between jobs 1 and 2 which raises the ML to 6 and to keep the ML unchanged, respectively. Stochastic node $(2,2)$ represents the ML to be 2 when starting job $j = 2$. This can only occur if the ML was already 2 after job 1 was finished and no PMA was executed. Lets have a closer look on the outgoing transitions from $(2,2)$. As job 2 can cause a deterioration of $\delta_2^1 = 1$,

$\delta_2^2 = 3$, $\delta_2^3 = 4$ or $\delta_2^4 = 9$, there are three transitions from state $(2,2)$ to the decision states in stage 3, namely to $[3,1]$, $[3,4]$, and $[3,5]$. Note that $\delta_2^2 = 3$ and $\delta_2^4 = 9 = \delta_2^2 + l^{\max}$ are represented by the same transition to $[3,5]$. Both transitions to $[3,4]$ and $[3,5]$ involve EMAs while the transition to $[3,1]$ does not.

The expected contribution always depends on the considered state $[j,l]$ and the decision $m_{l,j}$ in that state. Job $j=3$ can cause either a deterioration of $\delta_3^1 = 2$ or $\delta_3^2 = 5$. Therefore, an EMA occurs if no PMA is done and the deterioration δ_3^2 occurs. For state $[3,3]$, we then have

$$\begin{aligned} E[\tilde{B}_{3,3,1}] &= 1 \cdot \left((6-3) + d^{\text{ema}} \cdot \sum_{k=1}^{2} q_3^k \cdot \max\{\lceil (\delta_3^k - 6)/l^{\max} \rceil, 0\} \right) + (1-1) \cdot [\cdots] \\ &= 3 \\ E[\tilde{B}_{3,3,0}] &= 0 \cdot [\cdots] + (1-0) \cdot [d^{\text{ema}} \cdot (0.2 \cdot 0 + 0.8 \cdot 1)] = 0.8 \cdot 25 = 20 \end{aligned}$$

The above expressions indicate the time expected to elapse while job 3 is being processed, ML 3 is present and a PMA is being executed ($m_{3,3} = 1$) or no PMA is being executed ($m_{3,3} = 0$). Thus, the expected contribution depends on the decision $m_{3,3}$ and is then either $(3-3+1) \cdot 3 = 3$, if $m_{3,3} = 1$, or $(3-3+1) \cdot 20 = 20$, if $m_{3,3} = 0$. The Bellman function provides the state's value

$$\begin{aligned} d(3,3) = \min \quad & \{(3-3+1) E[\tilde{B}_{3,3,m_{3,3}}] + \sum_{k=1}^{K_3} q_3^k \cdot d\left(4, \hat{l}(3,3,m_{3,3},k)\right) \mid m_{3,3} \in \{0,1\}\} \\ = \min \quad & \{\underbrace{3 + 0.2 \cdot d(4,4) + 0.8 \cdot d(4,1)}_{m_{3,3}=1}, \underbrace{20 + 0.2 \cdot d(4,1) + 0.8 \cdot d(4,4)}_{m_{3,3}=0}\} = 3 \end{aligned}$$

due to the boundary conditions $d(4,l) = 0$, $\forall\, l = 0, \ldots, l^{\max}$. As a result, it can be stated that when the state $[3,3]$ is reached, a PMA provides the best decision and thus $m_{3,3} = 1$ is set. In this way, all stages of the DP can be evaluated backward and the optimal decision policy can be determined.

It can be seen in Example 4 that not necessarily all stochastic nodes in stage 1 can be reached from $[1,4]$. Nodes that cannot be reached are not needed for the evaluation of the DP. We observe the same for states and nodes in later stages. Therefore, we define subsets $\hat{D}_j \subseteq D_j$ and $\hat{S}_j \subseteq S_j$ containing all states and nodes that actually can be reached. Building the DP from left to right and only considering states and nodes that can be reached, can reduce computational overhead.

In Figure 3 the difference between sets D_j and $\hat{D}_j$ as well as between S_j and $\hat{S}_j$ can be easily seen. States and nodes that have no incoming transition and no outgoing transition are not in $\hat{D}_j$ or $\hat{S}_j$.

In particular, it suffices to apply the Bellman equation in (9) only to the states $[j,L] \in \hat{D}_j$, which can reduce the computational effort.

The (re-)formulation of the DP as a linear program (LP) is a common approach allowing to solve the problem with a standard solver (cf. Powell (2011)). For the purpose of this work the LP corresponding to the DP just presented, will be formulated and used in Section 5 to compare the performance of the DP with the LP solver's.

At this point we only want to state the LP and roughly explain the idea. The structure of the DP is transferred by setting the Bellman value of the states in the DP as the decision variables in the LP. The minimization in the definition of the Bellman equation is conducted by defining suitable constraints and the direction of optimization. Note that those constraints imply upper bounds on the Bellman value $d(j,l)$ of each state (j,l) if a PMA is conducted (first constraint) and if no PMA is conducted (second constraint), respectively. By maximizing the value of $d(j,l)$ the option resulting in the lower value is implicitly chosen. Note, furthermore, that having an exponential number of variables allows us to use continuous variables only despite the MPP-T-DEC being binary NP-hard. In the literature, the Bellman value of the initial state is often found as the objective function to be maximized, here $\max d(1,l_0)$. Note that, alternatively, the sum of the Bellman values of all states can be equivalently chosen as the objective function. We use the second variant as it reduces symmetry and the solution can be found more efficiently by the solver software using this formulation. The total expected completion time Z corresponds now to the value of the decision variable $d(1,l_0)$. This leads to the following LP.

$$\max \sum_{\substack{j=1,\dots,n \\ l=0,\dots,l^{\max}}} d(j,l)$$

s.t.

$$d(j,l) \leq (n-j+1)E[\tilde{B}_{j,l,1}] + \sum_{k=1}^{K_j} q_j^k \cdot d(j+1, \hat{l}(j,l,1,k)) \;\forall j = 1,\dots,n,\; \forall l = 0,\dots,l^{\max}$$

$$d(j,l) \leq (n-j+1)E[\tilde{B}_{j,l,0}] + \sum_{k=1}^{K_j} q_j^k \cdot d(j+1, \hat{l}(j,l,0,k)) \;\forall j = 1,\dots,n,\; \forall l = 0,\dots,l^{\max}$$

$$d(n+1,l) = 0 \qquad \forall l = 0,\dots,l^{max}$$

5. Computational study

This section is dedicated to evaluate a series of computational tests for both optimization problems, MPP-N and MPP-T. In Section 5.1, we will provide a detailed description of our instance generator. Then, in Section 5.2 we will give an overview of the results for both optimization problems. In particular, we address run times of our approaches and,

in more detail, the ratio between the minimum total expected completion times obtained for different machine types. It should be noted that the input for both optimization problems does not differ. This allows us to compare the results on a transparent and a non-transparent machine directly using the same instance set.

All computational tests have been executed on a computer with 64 GB of RAM, an i7-12700K CPU @5 GHz running Ubuntu 22.04. All algorithms have been implemented in Java 18 and CPLEX 22.1.1 is used as LP solver.

5.1. Instances

Since there is no established instance set for MPP-N and MPP-T we develop our own instances. We distinguish between four instance classes, which differ in their underlying probability distributions of the jobs' deterioration.

In the following, we illustrate the instance generation procedure. While we vary the number n of jobs, the initial maintenance level l_0, the duration d^{ema} of an EMA, and the distributions of deteriorations, we fix $p_j = 0$ (as discussed before) and $l^{\max} = 50$. Note that we can account for all relations between deteriorations, l_0, and $l^{\max}$ by varying the former two only. The values for $l^{\max}$, l_0, n, and d^{ema} we used in a full-factorial manner are listed in Table 3. The parameters have been selected such that they represent different scenarios while all possible parameter values stand in a comprehensible relation to the maximum maintenance level of $l^{\max} = 50$. Thereby, we aim to develop instances that depict various conceivable real world scenarios. Note that considering only one value for $l^{\max}$ is no restriction since we can scale $l^{\max}$ and Δ_j simultaneously in order to obtain instances with the same characteristics and different values of $l^{\max}$.

Parameter	**Values**
$l^{\max}$	$\{50\}$
l_0	$\{0, 10, 20, 30, 40, 50\}$
n	$\{5, 10, 50, 100, 200, 500\}$
d^{ema}	$\{50, 75, 150, 500\}$
Π	$\{10, 20, 25, 30, 40\}$
K	$\{11, 31, 51\}$
q_B, q_I	$[i \cdot 0.1, (i+1) \cdot 0.1]$ with $i = 0, \ldots, 9$
q_{BE}	$[0, 0.05]$, $[0.05, 0.1]$, $[0.1, 0.2]$, $[0.2, 0.5]$, $[0.5, 1]$

Table 3. Parameter sets for instance generation.

The individual jobs per instance are then specified by the underlying discrete probability distribution of deterioration. In all instances we have an underlying interval of values in $[\Pi - (K-1)/2, \Pi + (K-1)/2]$ defined by its center Π and its odd width K. All deteriorations supported by jobs' distributions will be from this interval and, hence, by

varying Π and K we can steer the level and the spread of deteriorations. We restrict ourselves to parameter configurations that satisfy $\Pi - \frac{K-1}{2} \geq 0$. Thereby we ensure that only deteriorations of at least zero are considered. On the other hand, there is an explicit upper bound on deteriorations, that is given through the values of Π and K as detailed in Table 3. Thus, we allow deteriorations that exceed the maximum maintenance level $l^{\max}$. It remains to define the associated probabilities.

For the probability distribution of deteriorations supporting values in $[\Pi - (K-1)/2, \Pi + (K-1)/2]$, we employ three well-known discrete probability distributions, namely the Binomial distribution, the uniform distribution, and the Bernoulli distribution (cf. Pinsky and Karlin (2010)). Furthermore, we propose a fourth distribution, referred to as Inverted Binomial distribution.

- The *Binomial distribution* is defined by its distribution parameters $q_B \in [0,1]$ and $n_B \in \mathbb{N}$. It has a characteristic bell-shaped probability function on the support set $\{0, 1, \ldots, n_B\}$. For each job $j \in J$ of a given instance we draw $q_{(B,j)}$ randomly from $[i \cdot 0.1, (i+1) \cdot 0.1]$, for an $i \in \{0, \ldots, 9\}$, and set $n_B = K - 1$. Shifting the support set of $B(K-1, q_{(B,j)})$ to $\{\Pi - \frac{K-1}{2}, \ldots, \Pi + \frac{K-1}{2}\}$ we obtain a distribution that is suitable to reflect a job whose most probable deterioration is close to the expected value of deterioration and the probability for other deteriorations decreases the more the deterioration deviates from the expected value. Note that using this distribution we obtain $K_j = K$ for each job $j \in J$. We draw $q_{(B,j)}$ from the same interval for all jobs of one instance but vary i throughout our computational study.
- We use the *uniform distribution* $U(\Pi - \frac{K-1}{2}, \ldots, \Pi + \frac{K-1}{2})$ rendering each deterioration in the interval equally likely. Note that Π and K fully specify the distribution and, thus, jobs are identical in this setting. Again, we have $K_j = K$ for each job $j \in J$.
- The *Bernoulli distribution* is defined by its distribution parameter $q_{BE} \in [0,1]$. It has support set $\{0,1\}$ with 0 and 1 having probability $(1-q_{BE})$ and q_{BE}, respectively. For each job $j \in J$ we draw $q_{(BE,j)}$ randomly from an interval of values. Shifting the support set of $Bern(q_{(BE,j)})$ from $\{0,1\}$ to $\{\Pi - \frac{K-1}{2}, \Pi + \frac{K-1}{2}\}$ we obtain a distribution with $K_j = 2$ possible deteriorations for each job whose difference is specified by K. We draw $q_{(BE,j)}$ from the same interval for all jobs of an instance but vary the interval throughout our computational study.
- The *Inverted Binomial distribution* arises from a Binomial distribution again with parameters $q_I \in [0,1]$ and $n_I \in \mathbb{N}$. The motivation for this additional deterioration distribution is that we also want to reflect effects in which a job triggers both low and high deterioration with similar frequency. Moderate deterioration, on the other hand, is less likely to occur. Such a distribution is obtained by mirroring the curve of binomial distribution $B(K-1, q_{(I,j)})$ along the x-axis and shifting it along the

y-direction. Finally, normalisation is conducted in order to obtain a permissible probability distribution again. Formally, we get the probability

$$P_I(x) = \frac{(1 - P_B(x))}{\sum_{k=0}^{n} (1 - P_B(x))}$$

for $x = 0, \ldots, n$ where P_B and P_I are the the probability functions of the Binomial and Inverted Binomial distribution. Again, a shift of the support results in a distribution function of deteriorations with support in the required interval which is noted as $I(K-1, q_{(I,j)})$. Again, we obtain $K_j = K$ for each job $j \in J$.

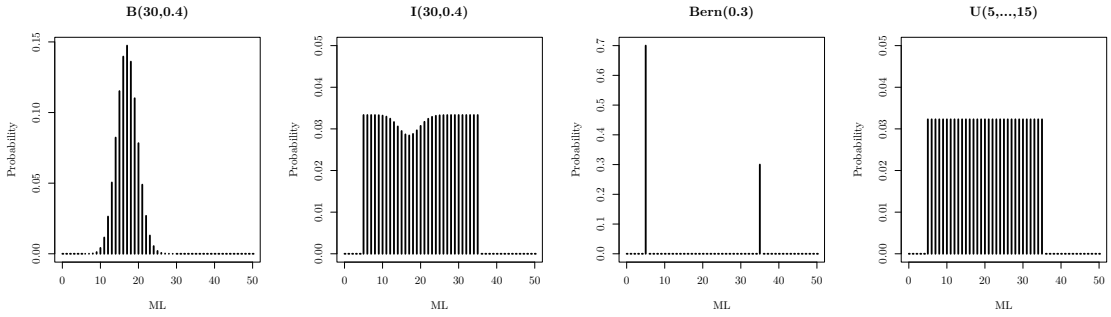


Figure 4. Deterioration distributions with common parameters $l^{\max} = 50$, $\Pi = 20$ and $K = 31$.

Figure 4 shows four deterioration distributions with common parameters $l^{\max} = 50$, $\Pi = 20$ and $K = 31$ and underlying distributions $B(30, 0.4)$, $I(30, 0.4)$, $Bern(0.3)$, and $U(5, \ldots, 35)$.

We design our test set by combining values of $l^{\max}$, l_0, n, and d^{ema}, types of distribution, and intervals to draw the distribution-specific parameters from (if applicable) as listed in Table 3 in a full-factorial manner (with the exception that we exclude combinations of Π and K leading to negative deteriorations). For each valid combination of values, type of distribution (except uniform distribution), and interval we generate $N = 10$ instances by randomly drawing the distribution-specific parameters for each job individually from the specified intervals. Note that for the uniform distribution we obtain identical instances and, hence, only $N = 1$ instance is generated. In total we generate 433,728 instances. In the following we consider average values for the N instances of each of the 44,928 instance classes based on the same parameter configuration.

Depending on the underlying distribution type we refer to the sets of all generated instance classes as $\mathcal{I}_B$ (Binomial), $\mathcal{I}_U$ (uniform), $\mathcal{I}_{BE}$ (Bernoulli) and $\mathcal{I}_I$ (inverted Binomial). The union of these sets $\mathcal{I}_{CS}$ builds the basis for the whole computational study.

5.2. Numerical results

First, we consider run times of our solution approaches in Section 5.2.1. Afterwards, we analyze the ratio $R(I) := Z^T(I)/Z^N(I)$ for any instance class $I \in \mathcal{I}_{CS}$ in Section 5.2.2.

5.2.1. Run times

Average run times and maximum run times for solving MPP-N and MPP-T are summarized in Table 4. Average run times and maximum run times are given for each distribution type and distinguishing transparent and non-transparent machines. Furthermore, the run times for solving MPP-T using the LP formulation proposed in Section 4 are given.

Average run time				**Maximum run time**			
	MPP-N	MPP-T (DP)	MPP-T (LP)		MPP-N	MPP-T (DP)	MPP-T (LP)
$\mathcal{I}_B$	0.36383s	0.00096s	0.67109 s	$\mathcal{I}_B$	2.819s	0.008s	83.742s
$\mathcal{I}_I$	0.36337s	0.00096s	0.09376s	$\mathcal{I}_I$	2.813s	0.008s	0.424s
$\mathcal{I}_U$	0.36646s	0.00097s	0.08608s	$\mathcal{I}_U$	2.815s	0.015s	0.439s
$\mathcal{I}_{BE}$	0.19136s	0.00011s	0.03251s	$\mathcal{I}_{BE}$	1.004s	0.001s	0.088s
$\mathcal{I}_{CS}$	0.33059s	0.0008s	0.30374s	$\mathcal{I}_{CS}$	2.819s	0.015s	83.742s

Table 4. Average run times and maximum run times.

The maximum run time over all instance classes using the presented solution approaches based on DPs is below 3s (recall that we have instance classes with up to $n = 500$ jobs) with an average of 0.1656s. In more detail, average run times for instances from instance classes with up to $n = 10$ jobs and $n = 50$ jobs are less than 0.01s and 0.04s, respectively. Having in mind a potential application in a decomposition approach solving a more involved problem (as sketched in Section 1) we conclude that our DP approaches are promising candidates for solving the subproblem.

As can be seen in Table 4, the average run time to find a solution for MPP-T is less than 0.015s using the implemented DP to solve instances in an arbitrary instance class. The run times for solving the LP formulation using CPLEX are significantly higher. These observations clearly show that using the LP does not provide any advantage over the DP. In particular, it is noticeable that the LP has difficulty solving instances with binomially distributed deteriorations.

The run times for MPP-T are significantly shorter than for MPP-N: The average run time for MPP-N is about 0.33059s while the average run time for MPP-T is 0.0008s. This remarkable difference can be explained by the different solution methods and the high effort of calculating all edge costs for MPP-N, see Section 3.2. Lowest average run times can be observed for $\mathcal{I}_{BE}$. This effect can be explained by the fact that independent from other instance parameters each job has only two possible deteriorations to be considered.

The complexity results in Section 2.2 already suggest that the number of jobs in the instance class also has a major influence on the run time. We, therefore, take a closer look on the relationship between the number of jobs n and the run times. Figure 5 shows the

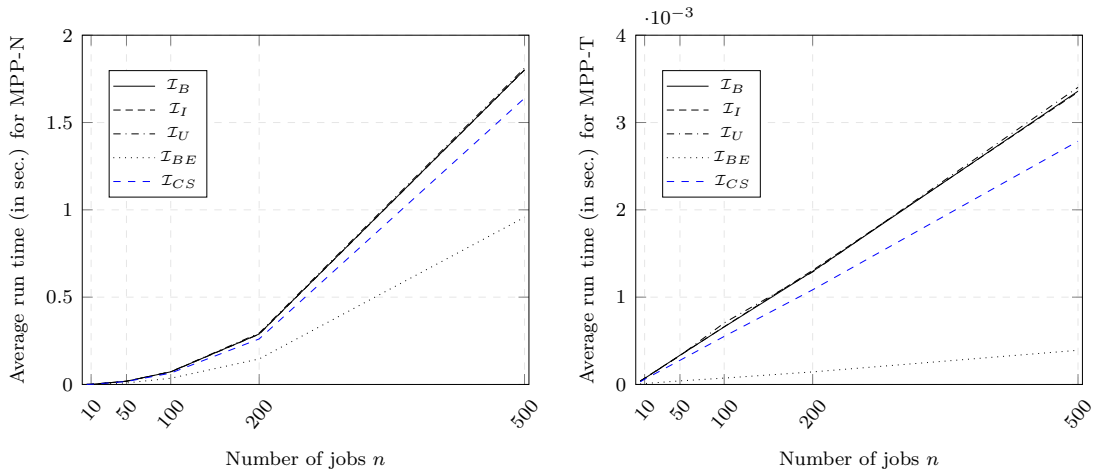


Figure 5. Average run times depending of the number of jobs.

average run times depending on the number of jobs in the instance class. As one would expect, the run times increase with increasing n in all cases and we observe an almost linear increase of average run times when solving MPP-T for increasing n. Note that the number of states and transitions scales linearly in n in the stochastic DP developed in Section 4 while the number of transitions is in $O(n^2)$ in the DP proposed in Section 3.

5.2.2. Ratio range

Since our solution approaches solve MPP-N and MPP-T to optimality we can empirically evaluate the average ratio $Z^T(I)/Z^N(I)$ for each instance class $I \in \mathcal{I}_{CS}$. We have seen in Section 2.3 that the advantage of a transparent machine over a non-transparent one can be arbitrarily large but we really had to finetune instances to let the advantage grow indefinitely. Our aim here is to see what parameters might drive the advantage in our test set. This kind of analysis might inform decisions in real world applications whether or not to invest in a transparent machine (revealing the current state). Table 5 provides a detailed overview of the ratios of all examined instance classes differentiated according to deterioration distribution type.

Upon examining the outcomes, it becomes apparent that many instance classes have exceedingly high ratios, meaning that utilizing the transparent machine will not lead to any noteworthy improvements in the objective function for these instance classes. In total, the ratio of 20,267 instance classes, which corresponds to roughly 45.1% of the dataset, equals 1. In these scenarios, the optimum decisions for the transparent machine are consistently

Instance class	# instance classes I with ratio $R(I)$										
	< 0.1	< 0.2	< 0.3	< 0.4	< 0.5	< 0.6	< 0.7	< 0.8	< 0.9	< 0.95	≤ 1
$\mathcal{I}_B$	0	0	0	0	7	14	25	34	79	220	17,280
	0%	0%	0%	0%	0.0405%	0.0810%	0.1447%	0.1968%	0.4572%	1.2732%	100%
$\mathcal{I}_I$	0	0	0	0	0	2	12	25	83	228	17,280
	0%	0%	0%	0%	0%	0.0116%	0.0694%	0.1447%	0.4803%	1.3194%	100%
$\mathcal{I}_U$	0	0	0	0	0	0	1	3	9	23	1,728
	0%	0%	0%	0%	0%	0%	0.0579%	0.1736%	0.5208%	1.331%	100%
$\mathcal{I}_{BE}$	0	0	0	1	7	13	23	42	116	283	8,640
	0 %	0 %	0 %	0.0116%	0.0810%	0.1505%	0.2662%	0.4861%	1.3426%	3.2755%	100%
$\mathcal{I}_{CS}$	0	0	0	1	14	29	61	104	287	754	44,928
	0 %	0 %	0 %	0.0022%	0.0312%	0.0645%	0.1358%	0.2315%	0.6388%	1.6782%	100%

Table 5. Overview of the cumulative distribution of ratios.

the same as for the non-transparent machine.

In contrast, however, there are also instance classes with a significantly lower ratio. For example, a ratio of 0.419 results from an instance class in $\mathcal{I}_B$ which is determined by the parameter configuration $n = 5$, $l_0 = 30$, $d^{\text{ema}} = 500$. The deteriorations of jobs are lying in the interval $[5, 15]$ ($K = 10$, $\Pi = 11$) and are binomially distributed with parameter $q_B \in [0, 0.1]$. This parameter configuration provides a class of instances with a high probability of very low deterioration per job. Higher deteriorations are possible but have a very low probability. If the higher deterioration occurs for each job in the sequence, the ML is exceeded and a MA becomes due. An EMA has a very long duration and, therefore, the risk of conducting one should be kept as low as possible. This setting resembles the structure we used to prove Lemma 2 and highlights sufficient conditions for a clear advantage of the transparent machine. On the transparent machine preventive PMAs are only carried out when the unlikely event of high deterioration has occurred. Conversely, the non-transparent machine lacks this information and schedules PMAs right before processing each job in order to prevent the overwhelming delay caused by an EMA, thereby causing delays in processing the job sequence.

The lowest average ratio of 0.378 is observed for the instance class with $n = 5$, $l_0 = 50$, $d^{\text{ema}} = 500$, and a Bernoulli distribution with parameters $K = 10$, $\Pi = 11$ and $q_{BE} \in [0.1, 0.2]$. In such instances each job reduces the ML of the machine by either 5 or 15, with the high deterioration having a much lower probability. Due to the high initial ML, maintenance becomes necessary if and only if the high deterioration occurs more than two times. In this setting, the transparent machine allows to situationally maintain if the high deterioration has actually already occurred twice. The non-transparent machine, however, gets maintained regularly since the high deterioration could have already occurred twice after the last PMA as a precaution to avert an expensive EMA.

In the following we more thoroughly investigate which parameters drive the ratio. First, we focus on parameters not related to the distribution of deterioration in Section 5.2.3. Afterwards, we consider parameters of the distributions in Section 5.2.4.

5.2.3. Distribution-independent parameters

We consider the influence of individual parameters on the ratio, namely the number n of jobs, the initial ML l_0, and the duration d^{ema} of an EMA.

- **Number of jobs** (n): The influence of the number of jobs on the average ratio is shown in Figure 6. It can be clearly seen that both, the average ratio and the minimum ratio, increases with increasing n for all considered deterioration distribution types. So it seems that for instance classes with many jobs there is less opportunity to profitably exploit the transparent machine.

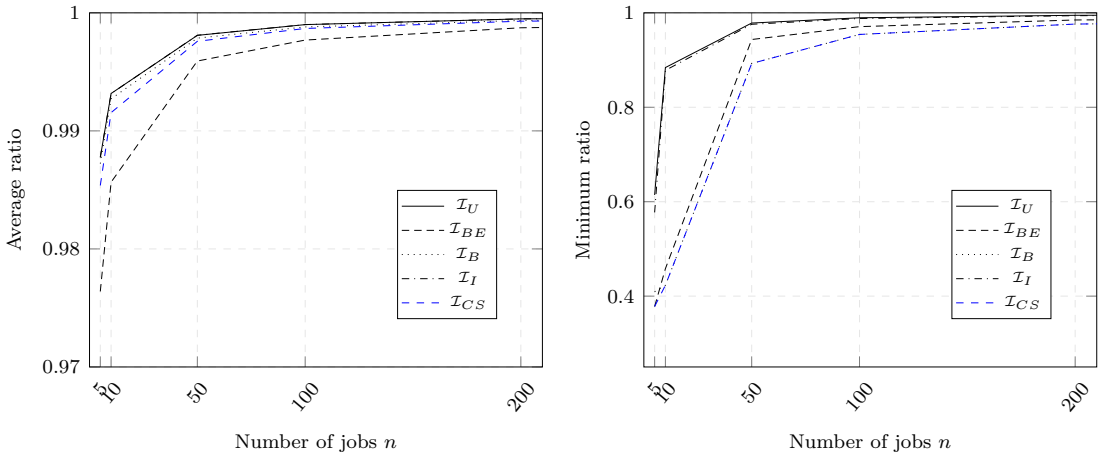


Figure 6. Average and minimum ratios depending on the number of jobs.

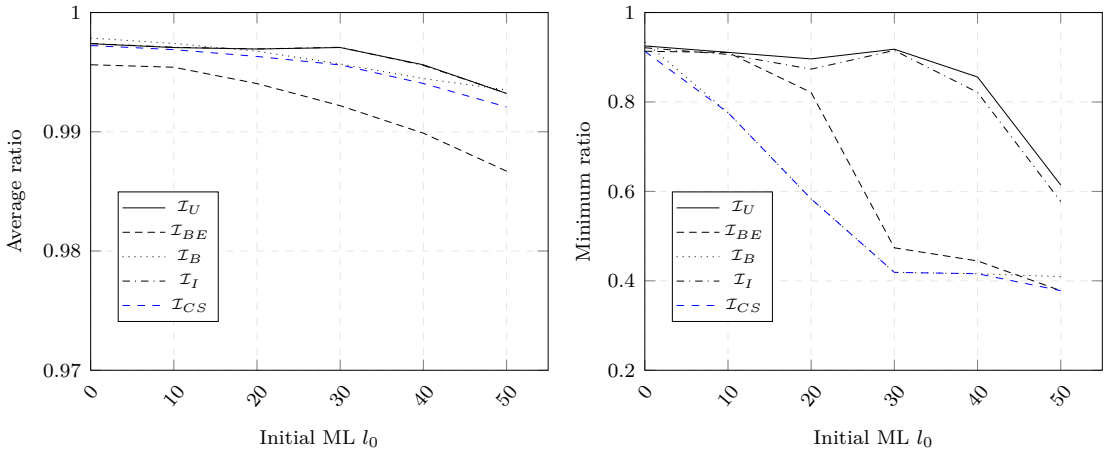


Figure 7. Average and minimum ratios depending on the initial maintenance level l_0.

- **Initial ML** (l_0): The curves in Figure 7 show the development of the minimum ratios and of the average ratios with increasing value of l_0 which resemble each other. For high initial MLs, lower ratios are achieved on average. It can be hypothesized that the higher the initial ML, the longer necessary PMAs can be delayed on the transparent machine compared to the non-transparent machine. If the ML is already low at the beginning of the observation period, both machine types may be forced to schedule PMAs promptly and thus have similar contributions in the objective function. Figure 7 shows that the decrease of the ratio with rising l_0 is most significant for $\mathcal{I}_B$ and $\mathcal{I}_{BE}$.
- **Duration of an EMA** (d^{ema}): The influence of the parameter d^{ema} is clearly visible in Figure 8 and can also be easily understood. As the duration of an EMA increases, both, the average and the minimum ratios, decrease for all instance class sets. This

effect can be explained by the fact that the advantage of the transparent machine lies precisely in scheduling PMAs only when they are really beneficial depending on the current ML. This is more often the case when an EMA could otherwise occur and the duration to be expected thereby exceeds the duration of a PMA. However, if an EMA does not take much more time than a PMA, then the time saving advantage of the transparent machine diminishes.

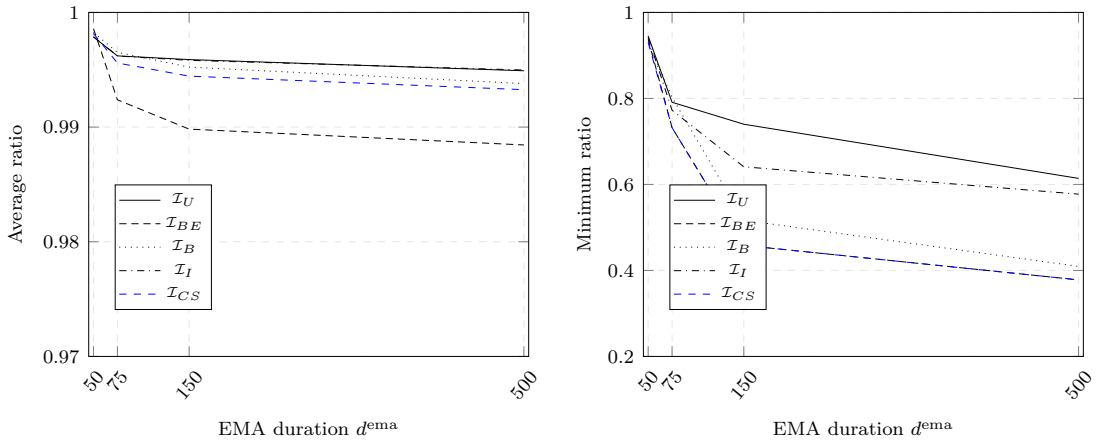


Figure 8. Average and minimum ratios depending on the duration d^{ema} of an EMA.

Summarizing, although we can see and explain developments of the ratios in varying numbers of jobs, initial MLs, and durations of EMAs the influence of each of these parameters individually is very small. In the following section we investigate their influence in combination with parameters of deterioration distributions.

5.2.4. Distribution-dependent parameters

In the last section, it has become clear that the different underlying deterioration distributions may have an impact on the ratio. As discussed before we see potential for higher ratios in particular for long durations of an EMA and deterioration distributions with a low probability of high deteriorations. Hence, in the following we investigate the influence of the distribution parameter q_B of $B(K-1, q_B)$ and the distribution parameter q_{BE} of $Bern(q_{BE})$ in combination with the duration d^{ema} of an EMA.

- **Binomial distribution parameter** (q_B): The parameter of the binomial distribution specifies the slope and position of the bell-shape. As q_B tends to 0 or 1 the distribution becomes more and more skewed to the left or right. In terms of deterioration, this means that if q_B tends to 0 very low deteriorations have high probabilities and the probabilities decrease as deteriorations increase. We consider the two-dimensional relationship between the parameters q_B and d^{ema} and the aver-

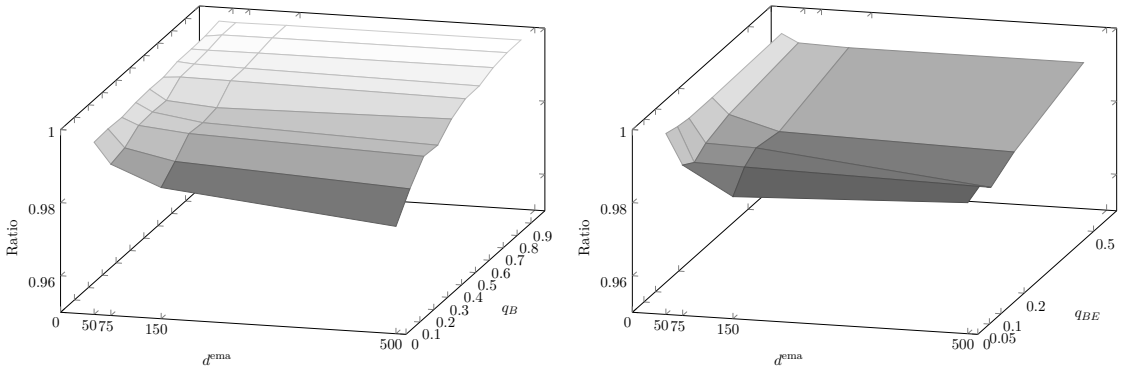


Figure 9. Average ratio for $\mathcal{I}_B$ depending on d^{ema} and q_B and for $\mathcal{I}_{BE}$ depending on d^{ema} and q_{BE}.

age ratio on the left side of Figure 9.

The clear picture emerges that the average ratio of instance classes in the set $\mathcal{I}_B$ strongly depends on a coupling from the duration of an EMA and the distribution parameter q_B. For high d^{ema} and low values of q_B a clear decrease in the ratio can be seen. This is consistent with previous observations. For a low q_B, the lower deterioration realizations in the interval are very likely to occur. Thus, with high probability, the ML of the machine decreases as slightly as possible for this deterioration distribution. However, EMAs are associated with such a high duration that on both, the transparent and the non-transparent machine, it is beneficial to prevent EMAs as much as possible. In the case of the non-transparent machine, this corresponds to the decision-making policy of always carrying out preventive PMAs in order to safeguard against the unlikely event of very high deterioration and a possible subsequent EMA. For the transparent machine, on the other hand, we can decide more flexibly and therefore perform fewer PMAs (or conduct them later), which has a positive effect on the expected sum of completion times.

- **Bernoulli distribution parameter** (q_{BE}): The results for instance classes in $\mathcal{I}_{BE}$ follow a similar pattern. The distribution parameter q_{BE} corresponds to the probability that a job causes the deterioration $\Pi + \frac{K-1}{2}$. In particular, for increasing q_{BE}, the probability of occurrence of the higher deterioration increases. Therefore, the savings potential decreases, as well, when using the transparent machine, which is why the average ratio increases, as can be seen on the right side of Figure 9.

Summarizing, while the ratio is close to 1 (and the advantage of the transparent machine is small) if the duration d^{ema} of an EMA is short or high deteriorations are either likely or impossible the transparent machine's potential over the non-transparent one increases if d^{ema} is long and high deteriorations are possible but unlikely. For these settings we have seen average ratios of instances from one instance class as low as 0.378.

6. Conclusion and outlook

In this paper, we treat a new version of a maintenance scheduling problem on a single machine. Unlike existing models, the job sequence is fixed, but we consider job-dependent and stochastic machine deteriorations caused by processing jobs. The machine is either transparent or non-transparent, i.e. we can observe its current state after each job is completed or not. In addition, we allow complete flexibility about when to schedule MAs as long as no job has to be interrupted. EMAs, which arise as soon as the ML would drop below zero, and PMAs increase the ML to the maximum ML delaying following jobs. After introducing the optimization problems with objective to minimize the expected total completion time we provide complexity results stating that both optimization problems are NP-hard. Furthermore, we develop solution approaches that solve the optimization problems exactly in pseudo-polynomial time. The computational study reveals that both approaches perform well as solving an instance in any case took less than 3.015s (less than 0.04s and 0.15s for up to 50 jobs and up to 100 jobs, respectively).

Our solutions approaches can be easily adapted to account for stochastic processing times of jobs, stochastic durations of MAs, setup times of MAs, or restarts of jobs interrupted by EMAs. While the Bellman function or transitions costs might differ in structure, the solution frameworks used here can handle these extensions. However, there is a conceivable possibility that the advantage of the transparent machine is more significant for alternative maintenance duration definitions. It is also interesting to involve an alternative type of non-transparent machine that can adjust decision policies online as soon as EMAs occur. This seems to be a practical variation, as an EMA is observable by the user of the machine, providing information about the machine's state and resulting possibly in different maintenance decisions.

Future research might allow continuous probability distributions reflecting deteriorations. While ideas used in this paper might be used we do not see how the same frameworks could be used. Furthermore, it would be interesting to resolve the fixed job sequence and consider the scheduling problem to sequence jobs and to schedule PMAs. Finally, in particular in real world applications one might benefit from simple decision rules for under which circumstances to conduct a PMA. It might be possible to derive those from structural insights into optimum solutions to the problem considered in this paper.

References

Ahmad, R., Kamaruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial application. *Computers & Industrial Engineering*, *63*(1), 135–149.

Bock, S., Briskorn, D., Horbach, A. (2012). Scheduling flexible maintenance activities subject to job-dependent machine deterioration. *Journal of Scheduling*, *15*(5), 565–578.

Brucker, P. (2004). Scheduling Algorithms. Springer. 4th ed.

Chen, J.S. (2008). Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. *European Journal of Operational Research*, *190*, 90–102.

Garey, M.R., Johnson, D.S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company.

Gawiejnowicz, S. (2020). A review of four decades of time-dependent scheduling: main results, new topics, and open problems. *Journal of Scheduling*, *23*(1), 3–47.

Gawiejnowicz, S., Kononov, A. (2010). Complexity and approximability of scheduling resumable proportionally deteriorating jobs. *European Journal of Operational Research*, *200*(1), 305–308.

Geurtsen, M., Didden, J., B.H.C., Adan, J., Atan, Z., Adan, I. (2023). Production, maintenance and resource scheduling: A review. *European Journal of Operational Research*, *305*(2), 501–529.

Gordon, V.S., Potts, C.N., Strusevich, V.A., Whitehead, J.D. (2008). Single machine scheduling models with deterioration and learning: handling precedence constraints via priority generation. *Journal of Scheduling*, *11*, 357–370.

Grigoriu, L., Briskorn, D. (2017). Scheduling jobs and maintenance activities subject to job-dependent machine deteriorations. *Journal of Scheduling*, *20*(2), 183–197.

Kubzin, M.A., Strusevich, V.A. (2006). Planning machine maintenance in two-machine shop scheduling. *Operations Research*, *54*(4), 789–800.

Lee, C.Y. (2004). Machine scheduling with availability constraints, in: Leung, J.Y.T. (Ed.), Handbook of scheduling: Algorithms, models and performance anaylsis. CRC Press. chapter 22, pp. pp. 22–1–22–13.

Lee, C.Y., Lei, L., Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research*, *70*, 1–41.

Lee, C.Y., Leon, V.J. (2001). Machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, *128*(1), 119–128.

Lee, C.Y., Lin, C.S. (2001). Single-machine scheduling with maintenance and repair rate-modifying activities. *European Journal of Operational Research*, *135*(3), 493–513.

Liao, C.J., Chen, W.J. (2003). Single-machine scheduling with periodic maintenance and nonresumable jobs. *Computers & Operations Research*, *30*(9), 1335–1347.

Lodree Jr, E.J., Geiger, C.D. (2010). A note on the optimal sequence position for a rate-modifying activity under simple linear deterioration. *European Journal of Operational Research*, *201*(2), 644–648.

Luo, W., Cheng, T.C.E., Ji, M. (2015). Single-machine scheduling with a variable maintenance activity. *Computers & Industrial Engineering*, *79*, 168–174.

Luo, W., Xu, Y., Tong, W., Lin, G. (2019). Single-machine scheduling with job-dependent machine deterioration. *Journal of Scheduling*, *22*(6), 691–707.

Ma, Y., Chu, C., Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, *58*(2), 199–211.

Mellouli, R., Sadfi, C., Chu, C., Kacem, I. (2009). Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times. *European Journal of Operational Research*, *197*(3), 1150–1165.

Mobley, R. (2002). An introduction to predictive maintenance. Elsevier.

Mor, B., Mosheiov, G. (2012). Scheduling a maintenance activity and due-window assignment based on common flow allowance. *International Journal of Production Economics*, *135*(1), 222–230.

Mosheiov, G., Sarig, A. (2009). Scheduling a maintenance activity and due-window assignment on a single machine. *Computers & Operations Research*, *36*(9), 2541–2545.

Mosheiov, G., Sidney, J.B. (2010). Scheduling a deteriorating maintenance activity on a single machine. *Journal of the Operational Research Society*, *61*(5), 882–887.

Pinedo, M. (2009). Planning and scheduling in manufacturing and services. Springer. 2nd ed.

Pinsky, M., Karlin, S. (2010). An introduction to stochastic modeling. Academic press.

Powell, W. (2011). Approximate Dynamic Programming: Solving the Curses of Dimensionality. John Wiley and Sons, Inc. 2nd ed.

Sanlaville, E., Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, *35*(9), 795–811.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, *121*(1), 1–15.

Shabtay, D., Steiner, G. (2007). A survey of scheduling with controllable processing times. *Discrete Applied Mathematics*, *155*(13), 1643–1666.

Strusevich, V.A., Rustogi, K., et al. (2017). Scheduling with time-changing effects and rate-modifying activities. Springer.

Sun, K., Li, H. (2010). Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines. *International Journal of Production Economics*, *124*(1), 151–158.

Vickson, R. (1980). Two single machine sequencing problems involving controllable job processing times. *AIIE transactions*, *12*(3), 258–262.

Xu, D., Yin, Y., Li, H. (2009). A note on "scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan". *European Journal of Operational Research*, *197*(2), 825–827.

Xu, S., Dong, W., Jin, M., Wang, L. (2020). Single-machine scheduling with fixed or flexible maintenance. *Computers & Industrial Engineering*, *139*, Article 106203.

Yang, S.J. (2010). Single-machine scheduling problems with both start-time dependent learning and position dependent aging effects under deteriorating maintenance consideration. *Applied Mathematics and Computation*, *217*(7), 3321–3329.

Yang, S.J. (2013). Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. *Applied Mathematical Modelling*, *37*(5), 2995–3005.

Yang, S.J., Yang, D.L. (2010). Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. *Omega*, *38*(6), 528–533.

Zhang, X., Yin, Y., Wu, C.C. (2017). Scheduling with non-decreasing deterioration jobs and variable maintenance activities on a single machine. *Engineering Optimization*, *49*(1), 84–97.

Zhao, C.l., Tang, H.Y. (2010). Single machine scheduling with general job-dependent aging effect and maintenance activities to minimize makespan. *Applied Mathematical Modelling*, *34*(3), 837–841.

C. Article III – Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration

Title: Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration
Authors: Arter, A., Briskorn, D., and Gönsch, J.
Journal: European Journal of Operational Research
Status: Published
Citation: Arter, A., Briskorn, D., Gönsch, J. (2026a). Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *335*(1), 67-82. https://doi.org/10.1016/j.ejor.2026.04.013

The following pages present a reprinted version of the above article, reformatted to match the layout of this dissertation.

Copyright:
© 2026 The Authors. Published by Elsevier B.V. This is an open access article distributed under the terms of the Creative Commons Attribution (CC BY) License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration

Antonia Arter[1], Dirk Briskorn[1], Jochen Gönsch[2]

[1]: Bergische Universität Wuppertal, Schumpeter School of Business and Economics,

[2]: Universität Duisburg-Essen, Mercator School of Management

This research has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 452470135.

Abstract

In this paper, we consider a single machine scheduling problem with a fixed job sequence involving job-dependent deterioration and maintenance activities. The deteriorations underlie continuous probability distributions and degrade the machine's state. Deteriorations may lead to a machine breakdown, which demands a time-consuming maintenance activity. To prevent the machine from breaking down, planned maintenance activities can be scheduled between jobs in order to restore the state of the machine. Thereby, the current state of the machine is observable at the completion of any job and is decisive for the decision about planning a maintenance activity. The goal is to minimize the expected total completion time of the jobs in the sequence. We formalize the optimization problem, formulate a dynamic programming formulation representing the problem, and present different heuristic solution approaches. In a computational study we analyze the results for the different proposed solution approaches and show that multiple approaches provide good solutions for the scheduling of planned maintenance activities.

Keywords: Scheduling · Approximate dynamic programming · Stochastic job-dependent machine deterioration · Maintenance activities · Single machine

1. Introduction

In the majority of scheduling literature, it is assumed that machines are available without restrictions over the entire planning horizon (see Brucker (2004) and Pinedo (2009)). In

reality, however, machines are constantly exposed to stress due to ongoing production, which is reflected in the deterioration of the machine. The machines must therefore be maintained in order to keep an acceptable production quality and preserve the longevity of the machine. *Maintenance activities* (MAs) usually lead to an interruption of production and therefore have a direct impact on the processing of jobs. For this reason, problem settings that take limited machine availability into account have increasingly been considered in research in the recent years.

In this paper we consider a variant of the problem introduced by Briskorn et al. (2024) which in turn is based on the problem introduced by Bock et al. (2012). A single machine has a *maintenance level* (ML) over time which indicates the condition of the machine. Thereby the ML at any point in time over the planning horizon is real-valued as opposed to integer valued as considered in Briskorn et al. (2024). The ML can be intuitively interpreted as the machine's state by reflecting that a higher ML represents a better condition the machine is in. Consequently, a ML of zero means that the condition of the machine is so poor that further use of the machine will result in the machine breaking down. If the processing of a job has not yet been completed when the machine's ML becomes zero, a breakdown occurs. In that case, immediately time-consuming repair work becomes necessary in order to repair the machine.

There are two actions that have a direct impact on the machine's ML. On the one hand, processing jobs causes stochastic deteriorations (real-valued) of the machine which corresponds to a drop in the ML. On the other hand, time-consuming MAs can be conducted in order to improve the condition of the machine. An MA always leaves the machine in perfect shape, thus, the ML rises to the maximum possible ML $l^{\max}$. There are two different types of MAs. *Planned maintenance activities* (PMAs) can be scheduled between any two consecutive jobs, at the beginning of the planning horizon or after completing the last job in the sequence. Thereby, the decision if a PMA takes place depends, among others, on the actual ML of the machine. Note that we do not allow PMAs to interrupt jobs. The reason is that it would require strong assumptions about occurrence of deterioration during the processing of jobs in order to time such PMAs and we are not convinced that such information is reliably available. There are no restrictions on the total number of PMAs apart from the natural restriction that no more PMAs can be scheduled than possible PMA slots are indicated from the number of jobs. In particular, a maximum of one PMA may be scheduled in each permitted PMA slot, whereby the execution of several PMAs would not bring any advantages anyway. *Emergency maintenance activities* (EMAs) are due in case the machine breaks down. Thus, EMAs are not scheduled explicitly but are triggered automatically as soon as a job's deterioration exceeds the ML at the start of the job. The duration of a PMA depends on the current condition of the machine. The smaller the ML is at the PMA's start, the longer the PMA takes since the ML is always increased to $l^{\max}$. EMAs necessarily start from ML zero. However, the duration of an EMA

exceeds the duration of a PMA started at ML zero since the EMA needs to be conducted spontaneously causing an unprepared need of materials and personnel. In any case, the processing of jobs needs to be stopped to conduct MAs. EMAs interrupt jobs, which can be completed after the EMA is completed while PMAs only start when no job is currently processed. Therefore, any kind of MA delays the production flow and, thus, the completion times of all following jobs. Obviously the need to consider this scheduling problem lies in the potential to save time and costs by scheduling less time-consuming PMAs at the right time to avoid larger delays due to EMAs caused by machine breakdowns in the further processing.

In this paper, we consider the problem of scheduling PMAs on a single machine with a fixed job sequence so that the expected total completion time is minimized. Thereby, when processing jobs the machine is exposed to stochastic job-dependent deterioration which follows a continuous job-dependent probability distribution which we assume to know. We assume that the ML can be observed by the decision maker at any time over the planning horizon. As opposed to the setting in Briskorn et al. (2024) the domain of the ML is not discrete in the paper at hand, that is the ML can take any real value from the closed interval $[0, l^{\max}]$. As a consequence, while continuous distributions may be more realistic, the combinatorial approaches proposed in Briskorn et al. (2024) cannot be directly applied.

We consider fixed job sequences, which at first glance seems to be a rather narrow setting. However, we find direct applications in practice. For example, a major pharmaceutical company in Germany has its production organized in large-batch campaigns with the sequence of production lots being fixed over long periods of time, sometimes months or years. It is important, then, to have sufficient buffer times for possible delays in production due to, for example, machine unavailability periods due to maintenance activities, so that the follow-up campaigns can start on time. Furthermore, the argument also applies to the completion of the individual jobs in the campaign for two reasons. For one thing, the company should be able to guarantee their customers achievable delivery dates to ensure customer satisfaction. From a more economic point of view on the producer's side it is also highly relevant to control the completion times of the individual jobs in the campaign in order to estimate the capital commitments during the campaign and to ensure liquidity. Taken together, buffer times are important but simultaneously they shall not exceed the actually needed time in an unnecessarily high manner. To summarize, these are precisely the reasons why we consider the problem at hand. Furthermore, the approaches developed in this paper can build a basis for application in production facilities which are more flexible in the order of their processed jobs. If the job sequence is no longer fixed, a metaheuristic approach could be used to handle the problem. By decomposing the decisions about the job sequence and the scheduling of PMAs, the metaheuristic might focus on deriving an optimum or good job sequence. In order to evaluate each candidate sequence we rely on having a fast approach determining the optimum PMA schedule for a given job sequence.

Finding such an approach is one of the main aims of this work.

Overall, the main contributions of this paper are threefold. First, we develop a dynamic programming (DP) model capturing the problem definition. Second, we propose multiple heuristic approaches tackling it. Finally, we compare the different approaches in an extensive computational study.

The paper is organized as follows: In Section 2 we will give an overview of the existing literature in the research field of scheduling including machine availability constraints. Section 3 is dedicated to formally define the optimization problem and provides a DP model formulation of the problem. In Section 4 we will present different heuristic solution approaches, namely an approximate dynamic programming (ADP) approach, a discretization approach and approaches using threshold policies, for the problem at hand. Section 5 evaluates and compares the presented solution approaches. We close the work with a conclusion, some managerial implications and an outlook on future research in Section 6.

2. Literature review

This section is intended to provide an overview of previous research in the field of scheduling under consideration of limited machine availability. Especially in the last two decades, the research community has become increasingly interested in scheduling problems with machine availability constraints. The earliest surveys are provided by Lee et al. (1997), Sanlaville and Schmidt (1998), and Schmidt (2000). They cover scheduling problems dealing with unavailability periods for single and multi machine settings. Thereby, the authors differentiate between unavailability due to machine breakdowns (stochastic), preventive maintenance activities (deterministic) or other constraints, with focus on the deterministic case and distinguish between non-preemptive and preemptive scheduling. More recently Ma et al. (2010) provided an updated survey on scheduling problems with deterministic machine availability constraints for different machine environments. The work of Dehnar Saidy and Taghavi-Fard (2008) includes the stochastic case. In Strusevich and Rustogi (2017), scheduling literature concerned with maintenance is reviewed and categorized according to the type of MA. The authors use the term *compulsory maintenance* for a setting where the machine is assigned MAs in prefixed time intervals, *flexible maintenance* for maintenance that is mandatory to be scheduled within given time windows or before reaching a deadline and *rate-modifying activities* (RMA) for production interruptions due to MAs that are not needed to keep processing going but improve the machine's speed. In general, the reviewed literature focuses on problem settings where MAs are not directly affected by job properties and the main driver for scheduling MAs is the elapsed time since the beginning of the planning horizon or the completion of the last MA. To the best of

our knowledge, the latest published survey is from Geurtsen et al. (2023). It addresses all kinds of scheduling problems involving maintenance scheduling, resource scheduling or a combination of both.

Ahmad and Kamaruddin (2012) provide a literature survey on scheduling problems with machine unavailability constraints from an application-oriented perspective. In most settings the machine becomes unavailable due to observable high failure rates of the equipment or breakdowns. Different types of MAs can be conducted in order to prevent both, failure costs and production loss, due to machine downtime. If the maintenance is *condition-based* it is assumed that different, mostly technical, parameters of the machine can be monitored and in the case of conspicuous deviations of, e.g. vibration or noise level, MAs are conducted. We find strong parallels to our machine state model since they also consider a measurable state of the machine. On the other hand, the authors consider *time-based* maintenance. MA decisions are determined based on the age and the expected lifetime of the equipment, which can be derived from observed failure time data. The main difference of these maintenance models lies in the independence between the processed jobs (except processing times) and the maintenance decisions. In contrast, we assume job-dependent deteriorations that are not necessarily correlated with the jobs' processing times.

We would like to shortly mention a research field, which, however, is only indirectly related to scheduling topics. In the area of *maintenance optimization*, MAs need to be scheduled mostly independent of job sequences to keep the system running. Thereby, machine deterioration is modeled as a stochastic process over time, see Sharma et al. (2011), De Jonge and Scarf (2020) and Arts et al. (2025) for review articles. In particular, the surveys provide interesting approaches for modeling the condition of a machine as a discrete or continuous state space. Even in the field of artificial intelligence there is a non-negligible body of literature dealing with maintenance optimization, see Ogunfowora and Najjaran (2023).

In the following paragraphs we will structure the literature according to various problem characteristics observed in the field of maintenance scheduling. Please note that we hereby do not have a strict partitioning since a problem can be specified according to different characteristics.

Many authors analyze maintenance scheduling problems with strict time constraints, typically with less flexibility than the approach in this paper offers. Often, time windows for scheduling MAs are fixed in advance, either restricting the MAs to be scheduled within a fixed interval or defining the unavailability periods, see Lee (1999), Mellouli et al. (2009), Gawiejnowicz and Kononov (2010), and Zhong et al. (2014). Meanwhile, the problems discussed in Lee et al. (2015) and Lei and Liu (2020) consider MAs that need to be scheduled within fixed time intervals in multi-machine settings. Mosheiov et al. (2018) do not force the scheduled MA to be scheduled fully within the interval boundaries, but have a fixed time window when to start the MA. In Yang et al. (2002) the authors deal with a pretty

plain setting of scheduling jobs and one single MA within a fixed interval to minimize the makespan on a single machine. Another time constraint can be found in the requirement to perform MAs periodically. This constraint can be reconciled with the fact that some machines must be regularly maintained over the entire planning horizon to avoid breakdowns, see, Chen (2008), Ji et al. (2007), and Xu et al. (2020) for single machine problems and Perez-Gonzalez et al. (2020) for a flow-shop problem. Close to that, there might be an upper bound for the time to elapse between two MAs, which is considered by Cui and Lu (2017) and Xu et al. (2020). Sun and Li (2010) extend the problem to a setting with two parallel identical machines to minimize the total completion time of jobs and Lee et al. (2018) deal with the problem of minimizing total tardiness. In Drozdowski et al. (2017) the authors consider a maintenance scheduling with similar restrictions, whereby the deadline by which maintenance must be carried out is not necessarily time-related but depends on the number of jobs performed. All these problems have in common that the state of the machine does not need to be monitored explicitly because it is assumed that no machine breakdown can occur if the MAs are conducted on time. In our problem setting machine breakdowns are not triggered by time passed and, hence, we consider the machine's state explicitly. Furthermore, it is feasible in our setting to accept machine breakdowns. The main reason for scheduling PMAs is to minimize the completion times by reducing machine downtime.

Another characteristic to classify the reviewed literature is given by the allowed number of MAs. This may be limited due to budget restrictions or the availability of maintenance personnel. Some authors allow at most one MA, see, e.g., Lee and Leon (2001), Mosheiov and Sarig (2009), and Sun et al. (2023), or enforce exactly one MA, see Kubzin and Strusevich (2006). The problem introduced by Lee and Lin (2001) also allows at most one MA. However, in the event of a machine failure prior to the scheduled MA, a repair work sets in and the MA is no longer necessary for the completion of jobs in the absence of further breakdowns. In principle, this setting is close to the ones in our problem formulation, but significantly less complex. In our problem setting, if EMAs occur, the ML of the machine is increased and, at least in the short term, MAs become less likely, depending of course on the deterioration of jobs. Other authors assume a given number of MAs to be scheduled. Yang and Yang (2013) consider a setting with unrelated machines and individual constraints for the number of MAs on each machine while in Yang (2013) the total number of MAs over all machines is limited. In our setting we do not have limitations for the number of triggered EMAs or the number of PMAs to be scheduled, except the natural limitation of PMAs due the number of jobs in the sequence.

In any MA scheduling problem with an objective function related to time, the duration of MAs has a high impact on the derived optimal maintenance schedule. Either the duration of an MA is constant, see, e.g., Lee and Lin (2001) and Mor and Mosheiov (2012), or the duration of an MA increases with the elapsed time since the beginning of the planning

horizon or the completion of the last MA. This goes along with the assumption that the state of the machine deteriorates over time and the worse the state is the longer it takes to maintain the machine. Thereby, Mosheiov and Sidney (2010) assume a linear increase in the MA's duration depending on its start time. On the other hand, Yang and Yang (2010) introduce a setting where MAs' durations get shorter the more MAs have been conducted so far due to learning effects. Detti et al. (2019) propose a more general approach by modeling the duration of the MA as a random variable which takes different values for different scenarios. In all these models, the state of the machine again is not needed to be monitored explicitly as we only have dependencies of the position of the MA or its start time and a scenario consideration. Problems where the durations of MAs depend on the current state of the machine have been considered in Bock et al. (2012), Briskorn et al. (2024) and Grigoriu and Briskorn (2017) and are closely related to the one in this paper. Thereby the state of the machine is not necessarily correlated to the elapsed time. In general, defining and monitoring the machine state in real world applications is a big issue in the field of predictive maintenance. Thereby, researchers try to use sensor techniques in order to be able to read out the machine's state at any time over the processing horizon and decide based on this about MAs. Especially from an operational point of view this delivers a pretty realistic setting. In our problem formulation we stay close to the model proposed by Briskorn et al. (2024) where the duration of a PMA depends on the state of the machine, which in turn depends on the jobs processed, PMAs scheduled and EMAs conducted so far, and we have a constant ML improvement rate. Other authors consider objective functions where costs are attributed to maintenance such that maintenance scheduling can be completely incorporated into the operations scheduling, see e.g. Randall and Basciftci (2025).

In a large field of literature it is assumed that MAs and the machine's speed are interdependent. Typically, machine performance declines with continuous use, known as the *ageing effect*. MAs are then used to improve or restore the condition of the machine. This creates a trade-off between delaying jobs for a MA and decreasing processing times after the MA. Lee and Leon (2001) introduce a rather simple setting with the option to schedule a single MA which changes the machine's speed. The processing times of jobs, then, depend on the fact if the job is scheduled before or after the potential MA. Mor and Mosheiov (2015) integrate due windows and job dependent modifying rates, thus, the extent by which the processing times are modified depends on the job. Mosheiov and Oron (2021) provide a dynamic programming algorithm for the single machine setting to schedule one MA with rate-modifying property. Zhao et al. (2009) propose a pseudo-polynomial time dynamic programming algorithm to minimize the total completion time for a two-parallel machine setting with a single rate-modifying MA per machine. In Phosavanh and Oron (2024) the authors deal with a two agent version of the problem. They provide complexity results for various objective functions. Ji and Cheng (2010) and Yang and Yang (2013) extend the problem to identical parallel machines, whereby each machine can be maintained several

times and each MA has its own modifying rate. Further, the processing times do not only depend on the yes or no category if the job is scheduled before or after a certain MA, but also on its position in the schedule related to the last MA. The latter assumption is frequently found in the literature and goes along with the understanding of a deteriorating system. As further examples we can name the works of Gordon et al. (2008) and Rustogi and Strusevich (2014). Thereby, both works incorporate time-dependent processing times. In total, the processing time of a scheduled job, thus, depends on the scheduled MAs, the position of the job in the sequence and the run time of the machine (both in relation to the last scheduled MA). Moreover, resource allocation may have an impact on the processing times, see Edis et al. (2013). By contrast, in the model at hand we have fixed processing times that are given as input.

The problem presented in the paper at hand goes back to the problem introduced by Bock et al. (2012). They consider a single machine with a measurable machine state and deterministic machine deterioration. PMAs need to be scheduled to prevent machine breakdowns categorically. A full MA restores the perfect machine state while a partial MA improves the machine state by an arbitrary, feasible increment. The duration of MAs increases linear with the proportion of the state improvement. The authors provide a complexity analysis for different objective functions, including total completion time. They show that the full MA case is always NP-hard, unless all jobs deteriorate the machine equally. If so, the problem can be solved in polynomial time. For the partial MA case, almost all problem variants can be solved in polynomial time. Luo et al. (2019) continue complexity analysis for the partial MA case and show that the problem is NP-hard for the total completion time as the objective function. Grigoriu and Briskorn (2017) consider makespan minimization problems for the full MA case and provide a 5/4 linear time approximation algorithm. Moreover, they extend the problem to a parallel machine setting, perform approximability analysis, and propose heuristics. In Briskorn et al. (2024), the authors take the problem introduced by Bock et al. (2012) limit themselves to fixed job sequences and extend the job-dependent deterioration to stochastic deterioration and minimize the total expected completion time. Machine breakdowns cause a time intense repair work. The authors introduce two machine settings based on the available information about the machine's state. The first setting is one in which the machine state cannot be directly read during processing. In this case, a pre-determined PMA schedule is required. The second setting is one in which the current state is continuously displayed. In this case, situational PMA decisions can be made. The authors prove NP-hardness for both settings. They also develop exact solution approaches using dynamic programming with pseudo-polynomial running times. In the paper at hand we take the problem setting from Briskorn et al. (2024) and extend the job-dependent stochastic deterioration to continuity.

With respect to its structure, the problem considered is a sequential or dynamic decision problem (see, e.g., Gromicho et al. (2012), Renke et al. (2021) and Shen et al. (2023))

characterized by regularly required PMA decisions. These decisions are conducted after completing any job in the given job sequence and are based on the available information holding the current ML of the machine. Beside the PMAs, the second relevant problem component is the random deterioration of the machine during the processing of jobs. The problem at hand, thus, can be classified as a stochastic, dynamic decision problem with binary decisions (for/against PMAs), a continuous random variable (deterioration) resulting in a continuous state variable (the machine's ML; see Section 3) over a finite horizon. This type of problems has been tackled numerically by three distinct communities: optimal control (e.g. Lewis et al. (2012)), machine learning/RL (e.g. Wiering and Van Otterlo (2012)), and DP (e.g. Powell (2007)). They worked mostly independently and in some cases even developed the same method under different names. Only recently, researchers started to unify this body of knowledge, see, e. g. Powell (2022).

A natural choice to model such problems are stochastic DP models or Markov decision processes. In literature, we find both concepts to model stochastic, dynamic decision problems in a quite similar manner using alternative terminologies. We will use a DP model in our work and, hence, use the corresponding terminology. DP models are often used if the considered decision problem has both a discrete state space and a discrete set of possible actions to decide on. The core idea of solving a DP model is by evaluating states using a value function and making optimal decisions based on these evaluations. For a manageable size of the discrete space and the decision options a DP, then, can be solved to optimality via backwards induction, see e.g., Powell (2007, Chapter 3.2). However, a continuous version of such a DP, as considered in this work, can only be solved exactly in very special cases, see, e.g., Powell (2022, Chapter 14.4). Fundamentally, today, there are two approaches to tackle these analytically intractable problems. The traditional way is forward dynamic programming, which basically mimics the underlying system. This system is repeatedly simulated over the time horizon, and these evaluations are used to find a good decision policy. This approach is probably so popular because simulating appears relatively easy and hides any curse of dimensionality very well, see Powell (2022, Chapter 16).

In this work we use a relatively new approach called backwards ADP, e.g. Powell (2022, Chapter 15). This approach goes backwards in time like classical backwards induction, but in each stage evaluates only a subset of states and/or transitions to approximate the value function. It is also often much faster than forward ADP, but requires careful sampling of the states evaluated to obtain a good approximation of the value function in the relevant area of the state space. Of course, there are also other approaches. For example, in the RL community the actor-critique framework is quite popular. This may include actor-critic algorithms that use separate approximations for the policy (the actor) and the value function (the critic) which were introduced by Barto et al. (1983) and Sutton (1984). These algorithms provably converge to a local optimum under certain conditions,

see Konda and Tsitsiklis (2000) for details. The approach we will use itself can be classified in this framework as a critic-only algorithm as it solely uses an approximation of the value function but not a separate approximation of the policy. Other approaches – more from the DP community – are a direct policy search, see, e.g., Nascimento and Powell (2009) and Scott and Powell (2012), or an approximate value iteration (see, e.g., Jiang and Powell (2015).

3. Optimization problem and model formulation

This section is dedicated to give a definition of the optimization problem studied in this paper. To do so, at first we provide the formal problem definition in Section 3.1 before we model the optimization problem as a stochastic DP in Section 3.2.

3.1. Problem definition

Table 1 summarizes the notation used in the problem definition introduced below. We assume a single machine provided with a numeric ML corresponding to the condition of the machine. The ML can take any real value between 0 and a maximum ML $l^{\max} \in \mathbb{R}^+$. At the beginning of the planning horizon the machine has a given initial ML $l_0 \in [0, l^{\max}]$. A set $J = \{1, \ldots, n\}$ of jobs needs to be processed on the machine in ascending order of the job indices. Thus, the sequence of jobs to be processed is fixed. The ML is affected over time by jobs being processed and non-preemptive maintenance activities being conducted. The ML is decreased over time by processing a job. Maintenance takes place in form of PMAs or automatically triggered EMAs to counter out the deterioration of the operating system or as a consequence of a machine breakdown.

The processing time of each job does not depend on the ML of the machine and is given by $p_j \geq 0$ for $j \in J$. Processing job $j \in J$ causes the ML to decrease by $\tilde{D}_j$. Thereby, $\tilde{D}_j$ is a random variable underlying a continuous probability density function f_j with positive support, thus $f_j(\delta) = 0$ for $\delta \in \mathbb{R}$, $\delta < 0$. The realization $\tilde{D}_j = \delta$ indicates that job j causes a reduction in the ML by a non-negative amount δ with an unknown course of deterioration over the processing time of j. The cumulative distribution function of $\tilde{D}_j$ is denoted as F_j. Further, we denote the dependent random variable representing the ML at the completion of job $j \in J$ as $\tilde{L}_j$. The corresponding probability distribution arises fully from the stochastic deterioration of all jobs processed so far and the PMA decisions. Here, for simplicity of notation $\tilde{L}_0$ reflects the certain initial ML of l_0.

A solution M is a function $M : J \times [0, l^{\max}] \to \{0, 1\}$, $(j, l) \mapsto M(j, l)$. The image of a

J	Set of jobs $\{1, \ldots, n\}$
$\tilde{C}_j$	Completion time of job $j \in J$
$\tilde{D}_j$	Deterioration of job $j \in J$
f_j	Probability density function of random variable $\tilde{D}_j$
F_j	Cumulative distribution function of random variable $\tilde{D}_j$
p_j	Processing time of job $j \in J$
d^{ema}	Duration of an EMA
$E_j^{\text{ema}}(L)$	Expected number of EMAs during job j started with ML $0 \leq L \leq l^{\max}$
$\tilde{L}_j$	ML at the completion of job j
$l^{\max}$	Maximum ML
l_0	Initial ML
M	Decision policy for MPP-C
n	Number of jobs
$M(j,l)$	Decision before job $j \in J$ at ML $l \in [0, l^{\max}]$
Z	Total expected completion time

Table 1. Notation for MPP-C.

given tuple $(j,l) \in J \times [0, l^{\max}]$ under M is either equal to zero or equal to one, indicating whether $(M(j,l) = 1)$ or not $(M(j,l) = 0)$ a PMA is planned immediately before job j, if the ML $\tilde{L}_{j-1}$ after completing job $j-1$ equals l. From a user's perspective, M thus provides a comprehensive decision policy regarding the deployment of PMAs across the entire planning horizon for all possible occurring machine states. We assume that the ML can be observed by the user at any time and, thus, in particular after finishing any job on the machine.

We assume that a PMA always raises the ML to the maximum possible ML $l^{\max}$ and, wlog., we assume a constant improvement rate of 1. In consequence, the duration of a PMA planned immediately before job $j \in J$ takes $l^{\max} - \tilde{L}_{j-1}$ time units and leaves the machine with ML $l^{\max}$.

If the ML drops to zero during the processing of a job an EMA becomes due. An EMA raises the ML to $l^{\max}$ and has fixed duration $d^{\text{ema}} \geq l^{\max}$. A plausible premise is that the duration of an EMA is at least as long as the duration of a PMA started at ML zero due to equipment or personnel being required at the machine unexpectedly or damaged tools which need to be exchanged or repaired. Note that if a deterioration $\tilde{D}_j = \delta$ with $\delta > l^{max}$ materializes, then EMAs cannot be avoided entirely and even multiple may be necessary while processing job j. It arises the question whether one could schedule job suspending PMAs during the processing of individual jobs to avoid EMAs. Since we do not know when during the processing of a job the deterioration occurs and to which extent, the potential point in time when the machine breaks down hardly can be anticipated and, therefore, the advantage of suspending PMAs over EMAs is negligible. Furthermore, any interruption of processing on the machine involves additional effort, as the machine would

have to be switched from processing mode to maintenance mode which is easier to switch on when no job is currently being processed.

The completion times of jobs $j \in J$ are represented by random variables $\tilde{C}_j$. For ease of notation $\tilde{C}_0 = 0$. Note that the completion times $\tilde{C}_j$ as well as the present MLs $\tilde{L}_j$ depend on the solution M at hand, but for reasons of readability we omit an additional M in the index. We can express $\tilde{C}_j$ as the sum of four components: $\tilde{C}_{j-1}$ (the completion time of the previous job), a random variable representing the duration of a PMA between job $j-1$ and j, the deterministic processing time of job j, and a random variable representing the total duration of EMAs during job j's processing. Our goal is to find a solution that minimizes the expected total completion time, given by $E\left[\sum_{j=1}^{n} \tilde{C}_j\right] = \sum_{j=1}^{n} E[\tilde{C}_j]$. Each $E[\tilde{C}_j]$ can be broken down into the expected values of the individual components mentioned. To understand how a solution impacts the expected total completion time, we first need to determine the expected number of EMAs based on the ML at the start of a job.

A job $j \in J$ starts either immediately after the completion of the previous job or after the completion of the PMA, if one is scheduled between these two jobs. In the first case, the ML at the start time of j is $\tilde{L}_{j-1}$. If a PMA is scheduled, the ML at j's start is known to be $l^{\max}$. The current ML has a direct influence on how the processing of j proceeds. Depending on the realized deterioration during the processing of job j, the processing time can be extended by triggering EMAs. We denote the expected number of EMAs during job j started with ML $L \in [0, l^{\max}]$ as $E_j^{\text{ema}}(L)$. It holds

$$E_j^{\text{ema}}(L) = \int_{\mathbb{R}^+} f_j(\delta) \cdot \max\{0, \lceil (\delta - L)/l^{\max} \rceil\}\, d\delta. \tag{1}$$

With the expression for $E_j^{\text{ema}}(L)$ we are now equipped to derive $E[\tilde{C}_j]$. For a solution M we get

$$\begin{aligned} E[\tilde{C}_j] = E[\tilde{C}_{j-1}] + \underbrace{E[M(j, \tilde{L}_{j-1}) \cdot (l^{\max} - \tilde{L}_{j-1})]}_{\text{expected duration of PMA before } j} + p_j \\ + \int_0^{l^{\max}} q_{j-1}(l) \cdot \underbrace{E_j^{\text{ema}}(l + M(j,l) \cdot (l^{\max} - l))}_{\text{expected number of EMAs during } j \text{ at ML } l} \cdot d^{\text{ema}} dl \end{aligned} \tag{2}$$

for the probability density function q_{j-1} of the random variable $\tilde{L}_{j-1}$ for $j > 1$ if all PMAs are conducted according to the decision policy M. Recall that $\tilde{L}_0$ can be seen as a random variable taking the value l_0 for certain. Note that density function q_{j-1}, and in turn also $E[\tilde{C}_j]$, are dependent on M but we, again, decided not to add an additional index for readability reasons. The remaining summands in (2) correspond to the expected completion time of the preceding job plus the processing time of job j. Note, that all the terms do not depend on assumptions on how the deterioration is distributed over time during the processing of an individual job.

To conclude, the maintenance planning problem with continuous deterioration (MPP-C) is to find a solution with minimum total expected completion time of jobs, which is given by $Z = E[\sum_{j=1}^{n} \tilde{C}_j] = \sum_{j=1}^{n} E[\tilde{C}_j]$.

Note that conducting a PMA after the last job to restore the state of the machine at the end of the planning horizon does not impact the objective value, as it does not delay any job in the sequence. It is easy to see that the processing times of jobs contribute a constant, namely, $\sum_{j \in J} (n+1-j) \cdot p_j$, to the total expected completion time of each solution. Consequently, we will assume, for the remainder of this paper, that $p_j = 0$ for each $j \in J$. It should be noted that this assumption still allows us to find optimal solutions for general problem instances with non-zero processing times, but we do not have to consider the constant in every mathematical expression which improves readability.

Briskorn et al. (2024) have shown that the problem with deteriorations following discrete distributions is NP-hard. This result carries over to MPP-C as it concerns a more general setting.

3.2. DP model

For the formal model formulation of the maintenance scheduling problem as a DP, we use the standard notation introduced by Powell (2007). Any DP can be defined by the following components: states, decision variables, exogenous information, a transition function, and a contribution function. States correspond to certain points in time over the planning horizon and encompass all information necessary to go forward. A fundamental distinction is made whether this point in time is right before or after a decision is made. In the second case all direct effects of the decision on the system are already taken into account and reflected in the state. Depending on the perspective the states are referred to as pre-decision states or post-decision states. In each case, the goal is to evaluate the states numerically with a suitable value function which will be used to determine the optimum decision policy and simultaneously calculate the value of the objective function. We will define the DP via post-decision states, as a preparation for the later on proposed ADP approach. The post-decision state's decisive advantage is that in our ADP approach, it inherently encompasses a Monte Carlo approximation of the DP's expectation calculated at every stage.

Before the formal definition of states we would like to summarize the events. Figure 1 illustrates the sequence of events between two points in time which will be represented by post-decision states. The ML right before the start of job j is denoted by $L_j \in [0, l^{\max}]$. The exogenous information δ_j, corresponding to the realized deterioration, becomes known while job j is processed. After its completion a new ML is observed and it is decided

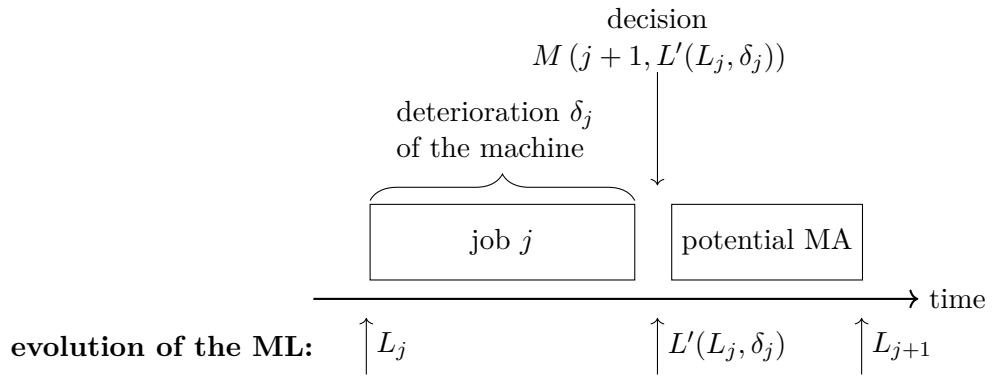


Figure 1. Sequence of events between two post-decision states for $j \in \{1, \ldots, n\}$.

whether to conduct a PMA. The observed ML $L'(L_j, \delta_j)$ is the realization of the random variable $\tilde{L}_j$ conditional on L_j and δ_j and is, therefore, deterministic. Binary indicator $M(j+1, L'(L_j, \delta_j))$ refers to the decision regarding a PMA immediately before $j+1$ when this ML is reached. The ML after the PMA (if conducted) and immediately before the next job, $j+1$, is denoted by L_{j+1}. Further, we introduce a dummy job $j = 0$, whose deterioration $\tilde{D}_0$ is described by a Dirac delta distribution at $\delta = 0$. Therefore, $f_0(\delta) = 0$ for $\delta \neq 0$, as the entire probability is concentrated at $\delta = 0$. In the problem context this means that job $j = 0$ causes no deterioration and, thus, cannot trigger any EMA and no time elapses between its start and completion. Note that the only relevant value for the ML L_0 right before the dummy job coincides with the ML at the beginning of the planning horizon l_0. As a result, the introduction of the job has no impact on the objective function but will have technical advantages to uniformly define post-decision states.

The stochasticity of the problem arises fully from the deterioration caused by jobs. Each job's deterioration is independent of the realized deteriorations of all foregoing jobs. Therefore, all we need for future decisions is the index which indicates the progress along the job sequence and the current ML of the machine. Accordingly, we define post-decision states as (j, L_j) for each $j = 1, \ldots, n$ and $L_j \in [0, l^{\max}]$ and, additionally, state $(0, L_0) = (0, l_0)$ corresponding to dummy job 0. Each state (j, L_j) corresponds to the situation where the machine has an ML of exactly L_j right before job j starts. Note that the future evolution of the system, which basically is the evolution of the ML of the machine, does not depend on any other historical information not captured by the state definition.

The transition function S describes the next reached post-decision state if the machine is in state (j, L_j), deterioration δ_j realizes for job j, and we take the PMA decision m.

$$S((j, L_j), \delta_j, m) := \begin{cases} (j+1, l^{\max}) & \text{if } m = 1, \\ (j+1, L'(L_j, \delta_j)) & \text{if } m = 0. \end{cases} \tag{3}$$

The transition function, therefore, keeps track of the machine's ML until the next post-

decision state is reached which always equals $l^{\max}$ if a PMA is scheduled.

The contribution function $\mathcal{C}((j, L_j), \delta_j, m)$ provides the contribution to the objective function related to the transition from state (j, L_j) for deterioration δ_j and decision m. Since we consider the objective to minimize the expected total completion time, the contribution corresponds to the time elapsing between the start of job j and the start of job $j+1$ for each $j < n$ and the start of job n and its completion scaled by the number of jobs whose completion is delayed. For a given post-decision state (j, L_j) for $j = 0, \ldots, n-1$, deterioration δ_j, and decision m we have the contribution

$$\mathcal{C}((j, L_j), \delta_j, m) := \underbrace{(n-j+1) \cdot \max\{\lceil (\delta_j - L_j)/l^{\max} \rceil, 0\} \cdot d^{\text{ema}}}_{:=\mathcal{C}_{\text{p}}((j,L_j),\delta_j)} + \underbrace{m \cdot (n-j) \cdot (l^{\max} - L'(L_j, \delta_j))}_{:=\mathcal{C}_{\text{PMA}}((j,L_j),\delta_j,m)}. \tag{4}$$

contribution associated with processing job j

contribution associated with a PMA right after job j

Since a PMA after completing job n does not impact the objective value we have the contribution associated with processing job $j = n$ as $\mathcal{C}_{\text{p}}((n, L_n), \delta_n)$.

Figure 2 depicts the entire sequence of events over the whole planning horizon. It can be seen that the value of the objective function corresponds to the sum of the expected contributions associated with the processing of all jobs $j = 0, \ldots, n$ and all scheduled PMAs, i.e. including the dummy job whose contribution originates only from the PMA right at the beginning. This observation can be used to define the Bellman function.

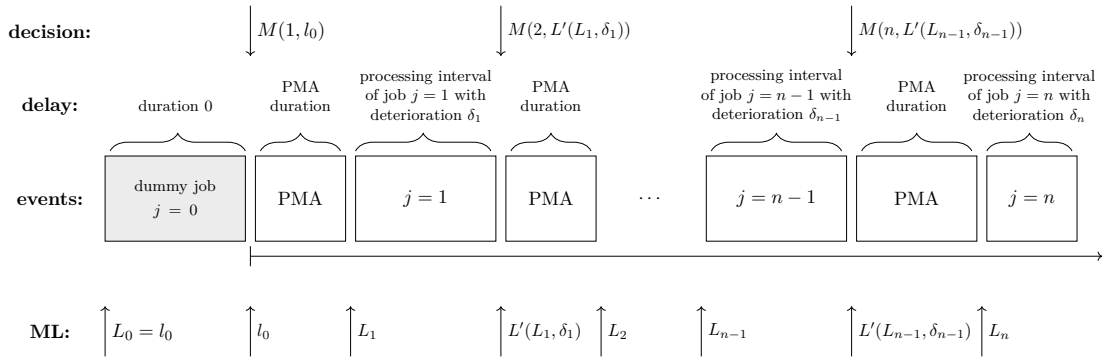


Figure 2. Visualization of all events over the planning horizon with realized deteriorations $\delta_1, \ldots, \delta_n$.

To conclude the design of the DP and to follow the target of finding the decision policy to minimize the expected sum of completion times we still need to define the post-decision value function V. The value function should correspond to the expected contribution to the objective function of future decisions. Hence, $V((j, L_j))$ reflects the expected total delay of completions of jobs $j, \ldots, n$ caused by MAs after the start of job j given that the ML at the start of j equals L_j. In other words, the value function V corresponds to

the solution of the subproblems consisting of the jobs $j' = j, j+1, \ldots, n$. We initialize the value function for the post-decision states with $j = n$ by evaluating the expression $V((n, L_n)) = E[\mathcal{C}_{\mathrm{p}}((n, L_n), \tilde{D}_n)]$. The Bellman function for the post-decision states for the remaining jobs $j = 0, \ldots, n-1$ is recursively defined as

$$\begin{aligned} V((j, L_j)) :&= E\left[\min_m \{\mathcal{C}((j, L_j), \tilde{D}_j, m) + V\left(S((j, L_j), \tilde{D}_j, m)\right)\} \mid (j, L_j)\right] \\ &= E\Big[\mathcal{C}_{\mathrm{p}}((j, L_j), \tilde{D}_j) \\ &\qquad + \min_m \left\{\mathcal{C}_{\mathrm{PMA}}((j, L_j), \tilde{D}_j, m) + V\left(S((j, L_j), \tilde{D}_j, m)\right)\right\} \mid (j, L_j)\Big] \end{aligned} \tag{5}$$

A complete evaluation of all post-decision states in the DP makes it possible to derive the optimal decision policy for any state (j, L) we are in. More precisely, the optimal decision $M^*(j, L)$ about a PMA before job j, $j = 1, \ldots, n$, at ML $L \in [0, l^{\max}]$ can be derived as

$$M^*(j, L) := \underset{m \in \{0,1\}}{\arg\min} \left\{m \cdot (n - j + 1) \cdot (l^{\max} - L) + V((j, m \cdot l^{\max} + (1 - m) \cdot L))\right\}. \tag{6}$$

It should be noted that in equation (6) in order to make an optimal decision for a given machine state only two values corresponding to the two decision options need to be compared with each other. The advantage here is that a decision can be computed quickly in any machine state, since we know the evaluations of the post-decision states. The objective value of the optimum decision policy corresponds to $V((0, l_0))$ since this is by definition the expected contribution to the objective value from the starting point at the beginning of the planning horizon.

4. Solution approaches

The exact solution of the DP formulation presented in Section 3.2 is computationally intractable for most instances. Therefore, we propose several heuristic solution approaches to determine decision policies for MPP-C, starting with an ADP approach in Section 4.1 and a discretization approach in Section 4.2. While these have a profound theoretical base, we lay the emphasis on relatively simple policies in Section 4.3 where we present static and dynamic threshold policies.

4.1. The approximate dynamic programming approach

As already anticipated in the foregoing sections, it is usually impossible to analytically evaluate the Bellman equation (5) of the DP model presented in Section 3.2. A common

way of dealing with this situation is to use an ADP approach to approximate the Bellman values $V((j,L))$ for $j = 0,\dots,n$ and $L \in [0, l^{\max}]$. The purpose now is to find an approximation $\hat{V} : \{0\} \cup J \times [0, l^{\max}] \to \mathbb{R}^+$ of V. To derive $\hat{V}$ we use a job-wise (also called stage-wise in general DP terms) approximation approach via $r \in \mathbb{N}$ univariate basis functions $\phi_i : [0, l^{\max}] \to \mathbb{R}^+$ for $i = 1,\dots,r$. For each job $j \in J$ the evaluation of the post-decision states is approximated by a linear combination of the basis functions ϕ_i, $i = 1,\dots,r$. Formally, $\hat{V}$ can be defined as follows

$$\hat{V} : \{0\} \cup J \times [0, l^{\max}] \to \mathbb{R}^+,\ (j,L) \mapsto \sum_{i=1}^{r} \beta_{j,i}\phi_i(L) \tag{7}$$

for real valued coefficients $\beta_{j,i}$ with $j = 1,\dots,n$ and $i = 1,\dots,r$. The process of determining the coefficients is referred to as learning.

To give a general overview of the approximation method, we will outline how $\hat{V}$ is derived step by step. Basically, we use a sampling approach combined with an ordinary least squares (OLS) method. To do so, a set of equidistant sampling points $l_1,\dots,l_k \in [0, l^{\max}]$, $k \in \mathbb{N}$, including 0 and $l^{\max}$ is defined. For each job deteriorations $\delta_{j,1},\dots,\delta_{j,k^{\text{samp}}} \in \mathit{supp}(f_j)$, $k^{\text{samp}} \in \mathbb{N}$, are drawn according to the underlying distribution. Thus, with increasing number of samples the drawn deteriorations give a good indication of the shape of the distribution which is essential for good approximations.

In general, we again follow a recursive evaluation scheme, but now with embedded approximations. We start with the initialization of the values of the post-decision states for $j = n$. For each sampling point l_i, $i = 1,\dots,k$, and each randomly drawn deterioration $\delta_{n,i'}$, $i' = 1,\dots,k^{\text{samp}}$ the state (n, l_i) is evaluated under the assumption that $\delta_{n,i'}$ materializes. Thus, we set $\hat{V}_{\text{samp}}(n, l_i, \delta_{j,i'}) := \mathcal{C}_{\text{p}}((n, l_i), \delta_{n,i'})$ and obtain the set Data_n of data points as follows

$$\text{Data}_n := \{\Big(l_i, \hat{V}_{\text{samp}}(n, l_i, \delta_{n,i'})\Big)\, | i = 1,\dots,k,\ i' = 1,\dots,k^{\text{samp}}\}. \tag{8}$$

We now obtain an expression for the evaluation of the post-decision states for $j = n$ using the OLS on the set Data_n to derive the coefficients $\beta_{n,i}$ for $i = 1,\dots,k$ as in (7). The optimum decisions about PMAs before job n can then be derived by using equation (6) replacing V by the approximation $\hat{V}$.

Using again a backwards evaluation for each $j = n-1,\dots,1$ and each $i' = 1,\dots,k^{\text{samp}}$ the state (j, l_i), $i = 1,\dots,k$ is evaluated under the assumption that $\tilde{D}_j = \delta_{j,i'}$ realizes with

$$\hat{V}_{samp}(j, l_i, \delta_{j,i'}) := \mathcal{C}((j, l_i), \delta_{j,i'}, m^*) + \hat{V}(S((j, l_i), \delta_{j,i'}, m^*))$$

for the optimum decision m^* derived with (6) using $\hat{V}$ for the evaluation of the post-decision states for $j+1$. We obtain data set Data_j and derive coefficients $\beta_{j,1},\dots,\beta_{j,r}$ using the method of OLS on the data sets Data_j. A complete backwards evaluation

and $n+1$-fold application of the OLS method provides a complete approximation of the valuations of the post-decision states for all $j = 0, \ldots, n$.

So far, we have defined the ADP framework, but it remains to choose the parameters. In order to find suitable basis functions, we conducted some preliminary tests to see which function types are suitable for describing the course of the Bellman values. Based on these preliminary observations, it was, e.g., concluded that the function of the Bellman values may have discontinuities. Therefore, it seems reasonable to have the opportunity of including indicator functions $\chi_{[0,s]}$ for $s \in (0, l^{\max}]$ to replicate potential steps in the Bellman values. Thereby, the choice of s should depend for each stage on the underlying probability distribution of deterioration since this impacts the ML where we would expect a discontinuity in the Bellman values. For a standardized use of the considered indicator functions, we define $\hat{\chi}_{j,\beta} := \chi_{[0,s_j(\beta)]}$ for the $(1-\beta)$-quantile $s_j(\beta)$ of the probability density function f_j. Here, β can be interpreted as the risk the decision-maker is willing to take that an EMA is triggered. Then, the full set of basis functions may vary among different stages of the ADP. We consider the full set Φ_j of basis functions on the jth stage of the ADP as $\Phi_j := \Phi_j^I \cup \Phi^P \cup \{e^x, \sqrt{x}\}$ with

$$\Phi_j^I := \{\hat{\chi}_{j,\beta} \,|\, \beta \in \{0, 0.05, 0.1, 0.5, 1\}\} \text{ and } \Phi^P := \{x^i \,|\, i = 0, 1, \ldots, 10\}.$$

Based on Φ_j, we consider three configurations equipping the ADP with different sets of basis functions. More precisely, we use the following subsets of Φ_j on the jth stage of the ADP:

- The *basic* configuration uses all introduced basis functions except the indicator functions, which has the advantage that the set of basis functions is not stage-dependent.
- The *stepwise* configuration uses only indicator functions.
- The *full* configuration includes all basis functions of the set Φ_j on the jth stage.

It remains to specify the number of samples and the choice of sampling points. Here, we take into account that on any stage an accurate approximation of the Bellman value for $l^{\max}$ is essential. This is because any decision about PMAs is made based on this Bellman value, as is clear from (6). Therefore, we decided to have more samples at $l^{\max}$ than at other sampling points. We chose $k = 100$ sampling points which are distributed equidistantly and 100 samples for each point but $l^{\max}$ for which we have 750 samples. These numbers have been determined in preliminary tests and turned out to be as small as possible without significantly losing approximation quality.

4.2. Discretization approach

Instead of using an ADP approach as presented in Section 4.1, another common approach is to discretize the problem at hand, solve the discretized version and generate a solution for the continuous problem from the discrete solution. In the following, we want to give a brief introduction how a discretization can be conducted for solving MPP-C. The prospect of success of the discretization approach is based on the well performing exact method for the discrete version of MPP-C using dynamic programming provided by Briskorn et al. (2024).

At first we want to elaborate on how we derive a discrete instance from a continuous instance for our optimization problem. We observe that the continuity of the problem results exclusively from the continuous deterioration of the machine during the processing of jobs. Thus, at first, we want to discretize the probability density functions f_j for $j = 1, \dots, n$. To do so, we partition the support interval $[a, b]$ equidistantly into $\zeta \in \mathbb{N}$ subintervals and select the midpoint of each subinterval as a discrete realization. The probability assigned to these discrete realizations, then, is set to the probability mass of the respective subinterval in the continuous distribution. By doing so, we receive a discrete probability distribution which traces the shape of the continuous distribution. Naturally, the finer the partitioning of the interval is, the more accurately the discrete probability distribution approximates the continuous distribution. For discrete deteriorations and since the initial ML l_0 is known, at each stage of the constructed DP, only a finite number of states, i.e. MLs, can be reached. Thus, the DP can, in theory, be reformulated to be entirely discrete in all components. However, determining the MLs that can be reached demands high computational effort increasing with the number of deteriorations per job. In particular, it is even necessary to determine all reachable states before solving the DP since it is evaluated backwards. Therefore, we decide on a slightly adapted procedure. The discrete deteriorations and l_0 are rounded to the nearest multiple of $10^{-\theta}$ for a degree of accuracy $\theta \in \mathbb{N}$. Thus, by subdividing the interval of the ML $[0, l^{\max}]$ into $l^{\max} \cdot 10^{\theta}$ discrete MLs being multiples of 10^{θ} we can construct a DP where we can define all reachable MLs as states. Although we might generate states that are never reached the overhead effort is less than the one for carefully generating only reachable states upfront.

Using the exact method provided by Briskorn et al. (2024), we obtain a decision policy for the discretized version of the instance at hand. It gives the decision about PMAs before any job $j = 1, \dots, n$ for the defined finite set of MLs. We derive a continuous decision policy by rounding any ML to the nearest discrete ML considered and applying the optimal decision resulting for the discrete problem. In the computational study we will use a discretization approach with $\zeta = 500$ discrete deteriorations per job and a discretizaton of the ML interval $[0, 25]$ into $25 \cdot 10^{\theta}$ equidistantly distributed MLs with $\theta = 3$. For sure, spoken in general terms, the precision of the discretization approach depends highly on

the chosen values of ζ and θ. The more deteriorations are considered and the finer the interval $[0, l^{\max}]$ is discretized the better results one can expect for the continuous problem converging to the optimal solution with increasing parameters. Obviously, however, this comes at a price of larger running times.

Preliminary tests show that already a rough discretization yields good results for MPP-C. At this point, we speak of good results if the observed objective values when applying the determined decision policy do not greatly improve using a decision policy provided by an even finer discretization. E.g., preliminary tests have shown that increasing $\theta = 4$ and keeping $\zeta = 500$ leads to objective values deviating in average only 0.0013% from the objective values originating from our proposed discretization configuration but with an average running time increase of 937%. Vice versa, when increasing $\zeta = 750$ and keeping $\theta = 3$ the average running time increase is around 35%, but the observed objective values are not significantly different either with an average change of 0.0044%. We like to emphasize that we still cannot guarantee to find an optimum solution although we can expect the discretization approach to converge towards one and observe exactly this effect in our preliminary tests. The proposed configuration, therefore, provides promising results within manageable running times.

4.3. Threshold policies

A solution M is a threshold policy if $M(j, l) \geq M(j, l')$ for each $j \in J$ and $0 \leq l < l' \leq l^{\max}$, that is, a PMA is not triggered for a higher ML if it is not triggered for a lower one. Such a policy is defined by a threshold value for all $j \in J$ which mandates a PMA to be carried out before job j is started if the ML is below the defined threshold value. From a user's perspective threshold policies are intuitive to grasp and rather easy to implement. We distinguish between *static* and *dynamic* threshold policies.

4.3.1. Static threshold policies

Static threshold policies are characterized by the fact that the thresholds depend exclusively on the properties of the machine and are not impacted by any properties of the jobs to be processed. Thereby, a parameter $\alpha \in [0, 1]$ specifies the level of the threshold in relation to the maximum ML $l^{\max}$. E.g., a static threshold policy with parameter $\alpha = 0.25$ corresponds to a policy with PMAs conducted if the ML drops below $0.25 \cdot l^{\max}$. In other words, PMAs are conducted if the maintenance level is below 25% of the full range. We denote the static threshold policy with parameter $\alpha \in [0, 1]$ as $\text{sTH}(\alpha)$. More formally,

sTH(α) implements the decision policy

$$M_{\text{sTH}(\alpha)} : J \times [0, l^{\max}] \to \{0, 1\}, (j, l) \mapsto \begin{cases} 1, & l < \alpha \cdot l^{\max}, \\ 0, & \text{otherwise.} \end{cases}$$

Preliminary tests reveal as one might expect that for different choices of α the performance of the static threshold policy varies. In Section 5 we will provide some insights into these preliminary tests including the chosen configuration of the static threshold policy used in the computational study.

4.3.2. Dynamic threshold policies

Dynamic threshold policies are more adapted to the problem instance. The decisions about PMAs take into account the distribution of the deterioration caused by the next job. Thus, an obvious major difference to the static threshold policies is that we do not have a unique threshold throughout the planning horizon but the thresholds differ for different decision points. The dynamic threshold policy mandates a PMA whenever the ML is such that the probability that the next job triggers an EMA if no PMA is performed is larger than $\eta \in [0, 1]$. In other words, the user can define the risk she is willing to take for an EMA and, thus, can incorporate her risk appetite. We denote the dynamic threshold policy with parameter $\eta \in [0, 1]$ as dTH(η). More formally, we obtain the decision policy

$$M_{\text{dTH}(\eta)} : J \times [0, l^{\max}] \to \{0, 1\}, (j, l) \mapsto \begin{cases} 1, & 1 - F_j(l) > \eta, \\ 0, & \text{otherwise.} \end{cases}$$

5. Computational study

This section is dedicated to evaluate and compare the different heuristic approaches presented in Section 4. Figure 3 provides a flowchart sketching how we proceed in the computational study. We see that it is conducted in three steps. First, the individual instance sets $\mathcal{I}_N$, $\mathcal{I}_e$ and $\mathcal{I}_U$ are created, whose union defines the data basis of our study, namely $\mathcal{I}_{CS}$, see Section 5.1. Decision policies are then derived in various ways, either by means of involved procedures (ADP or discretization) or by setting the steering parameter (sTH or dTH). Once solutions are determined they are evaluated by simulation. The results are presented in Section 5.2.

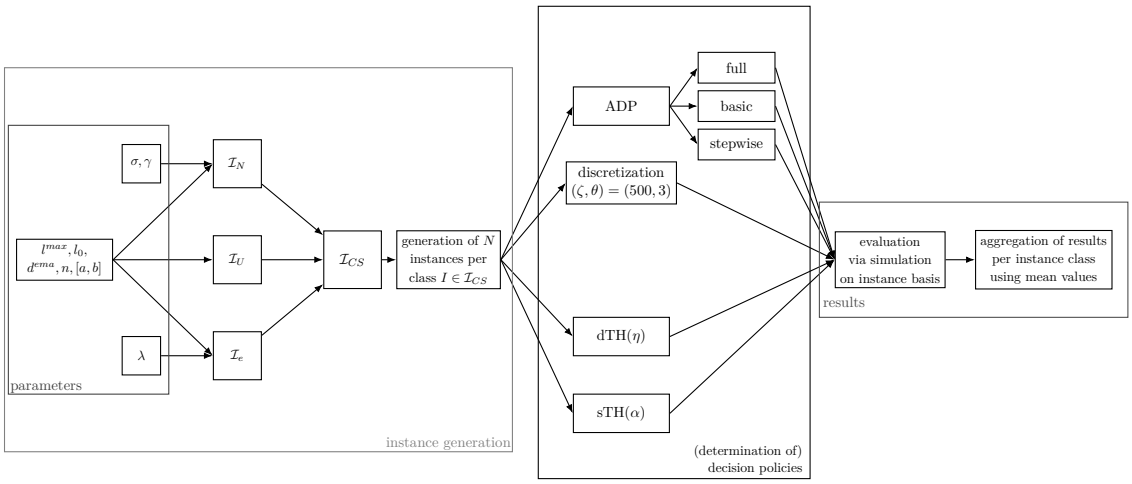


Figure 3. Flowchart depicting the layout of the computational study.

5.1. Instances

To the best of our knowledge, there is no established instance set for MPP-C. Therefore, we propose one in the section at hand.

5.1.1. Continuous probability distributions for deterioration

Before we can define the instances considered, we want to give a short overview of the continuous probability distributions used to model the deteriorations. Note that some common continuous probability distributions are supported on the whole real line. In our problem setting we do not allow negative deterioration, however. Furthermore, disproportionate high deteriorations do not seem realistic. Hence, we consider truncated distributions that are derived from the original distribution but are supported on an interval $[a, b]$, $0 \leq a < b < \infty$, only. For this purpose, we use a common approach of truncation, see Cramér (1999), which is stated as follows. Let X be a random variable underlying the parent distribution with probability density function f and cumulative distribution function F. The truncated density function f_t is then derived as follows

$$f_t(x) = f(x|a \leq X \leq b) = \begin{cases} \frac{f(x)}{F(b)-F(a)}, & a \leq x \leq b, \\ 0, & \text{else.} \end{cases} \tag{9}$$

It should be noted that $f_t(x)$ is a probability density function that preserves the shape of the parent density function on $[a, b]$. In the following we will briefly describe the considered distribution types in the computational study.

- **Truncated normal distribution:** The density function of a random variable underlying the normal distribution is characterized by its symmetric, bell-shaped curve.

It is defined by the two parameters μ and σ corresponding to mean and standard deviation. Most of the realizations cluster around the mean and the probabilities of realizations further from μ decrease exponentially. Originally, any normal distribution with mean μ and standard deviation σ, noted as $\mathcal{N}(\mu, \sigma^2)$, is supported on whole $\mathbb{R}$. For truncation, we restrict the parent density function to an interval $[a, b]$ for real-valued parameters a and b with $0 \leq a < b < \infty$. We denote a truncated normal distribution on the interval $[a, b]$ with distribution parameters μ and σ as $\mathcal{N}(\mu, \sigma^2, a, b)$.

Note that truncation only preserves the distribution's shape on the interval $[a, b]$. Regarding the similarity of the truncated and the parent density function on whole $\mathbb{R}$, the choice of the interval bounds in relation to the parameters μ and σ is crucial. Figure 4 shows the probability density functions of different truncated distributions of the parent distribution $\mathcal{N}(5, 1)$. For example, for $\mathcal{N}(5, 1, 0, 12)$ the interval $[a, b]$ encompasses 3σ and, thus, more than 99.7% of the probability mass. Accordingly, $\mathcal{N}(5, 1, 0, 12)$ is close to the density function of the parent distribution $\mathcal{N}(5, 1)$, but with smaller (and bounded), strictly positive support. In the figure, there is no visible difference, the graphs fully overlap.

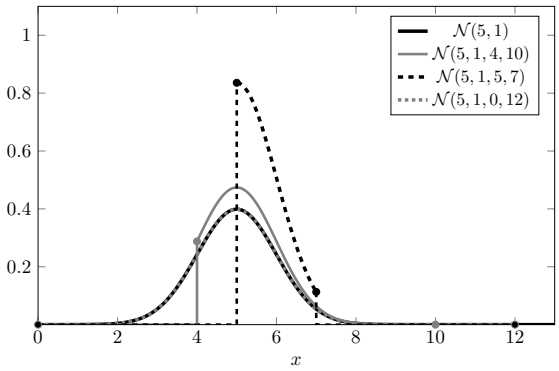


Figure 4. Density functions for truncated normal distributions of the parent distribution $\mathcal{N}(5, 1)$.

- **Continuous uniform distribution:** The continuous uniform distribution is a probability distribution where the outcome is an arbitrary value from the interval $[a, b]$ and all outcomes are equally likely. Thus, it represents a randomness, where the maximum and minimum possible outcomes are known. A continuous uniform distribution on the interval $[a, b]$ is denoted as $U(a, b)$ and, obviously, no truncation is necessary.
- **Truncated exponential distribution:** The exponential probability density function's shape is characterized by an exponential decline with increasing absolute value of the realization. The support of the exponential distribution is the real non-negative line and the value of the probability density function tends to zero if x goes to infinity. It has a long tail towards infinity, reflecting that immensely high

realizations are possible but do have low probabilities. The distribution is fully defined by its distribution parameter λ corresponding to the the inverse of its mean. Note, that the closer λ is to zero (i.e. the higher the mean), the flatter the curve gets and it spreads over a wider range of realizations with substantial probability. Here we do the truncation in two steps. First, we truncate the distribution from above to $[0, b-a]$. Second, we shift it to support $[a, b]$ by adding a to each randomly drawn realization. We will do so in the computational study to model deterioration scenarios where a lower bound a for the occurring deterioration is known and the probability decreases for larger deteriorations. We denote a truncated exponential distribution on the interval $[a, b]$ with distribution parameter λ as $Exp(\lambda, a, b)$.

5.1.2. Instance generation

We are now equipped to present the design of the instances considered in the computational study. The parameters used for generating the instances are listed in Table 2.

Thereby, the maximum ML $l^{\max}$ is constant for all instances considered to avoid scale effects. Briskorn et al. (2024) have shown for a related setting that varying the initial ML l_0 does not have any considerable impact on the solution structure. Thus, we keep l_0 equal to $l^{\max}$. This is also conceivable from a user's perspective since setting $l_0 = l^{\max}$ goes along with the assumption that the machine is handed over in perfect shape at the beginning of the planning horizon. We consider instances where jobs are homogeneous with regard to their types of deterioration. This means that the underlying probability distribution of the individual jobs in the sequence shall be similar but not necessarily equal. To achieve this, we assume that for a given instance the deteriorations of all jobs in the sequence underlie the same distribution type. Moreover, the deteriorations of jobs shall be of a similar magnitude. In instances where deteriorations are normally or exponentially distributed, this is achieved by letting density functions of all jobs share the same support set $[a, b]$.

Note, that for the uniformly distributed deteriorations having the same support means that jobs are completely identical. To avoid this, for this instance type the support of the probability density function for each job is defined as a random subset of a specified interval $[a, b]$. As the uniform distribution does not require any further parameters to be defined, the description of the instance generation of this instance type is therefore complete.

Considering the instance generation for instances with exponentially distributed deterioration, the distribution specific parameters λ_j need to be defined for every job $j = 1, \ldots, n$ in the sequence. Each λ_j, $j = 1, \ldots, n$, is randomly drawn from one of the domains attributed to the exponential parameter λ listed in Table 2. Thereby the same domain is

chosen for all jobs within one sequence. Then, the deterioration's distribution of job j is set to $Exp(\lambda_j, a, b)$.

The construction of instances with normally distributed deteriorations is a little more complex. A non-truncated normal distribution is defined by the two parameters μ and σ corresponding to the mean and standard deviation. As already anticipated, we want to construct a normal distribution truncated on $[a, b]$. To assure that sufficient probability mass lies within $[a, b]$, we demand the mean of the parent non-truncated distribution μ to lie within $[a, b]$. Its relative position can be defined by a position parameter $\gamma \in [0, 1]$ holding $\mu = a + \gamma \cdot (b - a)$. For each job j a position parameter γ_j and a standard deviation σ_j are randomly drawn implying that the distribution of deterioration of job j is $\mathcal{N}(a + \gamma_j \cdot (b - a), \sigma_j^2, a, b)$.

$l^{\max}$	25	Support $[a, b]$	$\{[0, 12.5], [0, 18.75], [0, 25], [6.25, 18.75], [6.25, 25]\}$
l_0	25	γ domain	$\{[0.25, 0.5], [0.5, 0.75], [0, 1]\}$
d^{ema}	$\{30, 50, 100, 250\}$	σ domain	$\{[0, 1.25], [1.25, 2.5]\}$
n	$\{5, 10, 20, 50\}$	λ domain	$\{[0, 0.1], [0.1, 0.5], [0.5, 1]^{*1}, [1, 2]^{*2}\}$

^{*1}: not used if $a = 0$ and $n < 20$ ^{*2}: not used if $a = 0$ and $n < 50$

Table 2. Parameter values and domains used for instance generation.

For each parameter assignment we randomly generate $N = 10$ instances. For $[a, b] = [0, \cdot]$ we do not use $\lambda \in [0.5, 1]$ or $\lambda \in [1, 2]$ depending on n in order to avoid instances with an overall deterioration which is likely to be less than the ML range of 25. Note that instances with maximum total deterioration of less than 25 are trivial since having no MA at all is optimum. In total, we generate $8,200$ instances. We aggregate the results for the 820 instance classes (defined by their parameter assignments and domains) using mean values. We denote the sets of instance classes depending on the underlying deterioration distribution type as $\mathcal{I}_N$ (normal), $\mathcal{I}_U$ (uniform) and $\mathcal{I}_e$ (exponential). Thereby the cardinality of the sets differs due to the different number of parameters and parameter domains to define an instance class. We have $|\mathcal{I}_N| = 480$, $|\mathcal{I}_e| = 260$ and $|\mathcal{I}_U| = 80$ and denote the union of these sets as $\mathcal{I}_{CS}$ which builds the complete data for the computational study.

5.2. Numerical results

This section is dedicated to evaluate the different heuristic methods proposed in Section 4. We want to investigate which method delivers the best results among all competitors and whether we can make a recommendation for the use of threshold policies. Since the different methods have strongly different characteristics, we also consider performance indicators that are only relevant for some methods.

First, in Section 5.2.1 we give some insights into the preliminary studies parameterize the

static and dynamic threshold policies introduced in Section 4.3. After that, in Section 5.2.2 we briefly elaborate whether the proposed ADP method yields a robust approach with regard to the stochasticity of the sampling, which is a central component of the learning process. This step is necessary to verify the fundamental reliability of this approach. Then we take a look at running times for determining decision policies in Section 5.2.3 before we analyze the performance of the derived decision policies in Section 5.2.4. In Section 5.2.5, the results are summarized and discussed with respect to practical recommendations.

In order to assess the quality of decision policies, they will be evaluated by simulation. The entire planning horizon is simulated several times by drawing deteriorations for each job according to the underlying distribution and carrying out PMAs according to the respective decision policy. For each simulation run and decision policy we then have an individual observed total completion time. We refer to the mean value over the total completion times of all simulation runs as the objective value associated with the policy in the following. Note that we use identical deterioration simulations (i.e. common random numbers) for evaluating different policies for comparability reasons.

All computational tests have been executed on a computer with 64 GB of RAM, an i7-12700K CPU @5 GHz running Ubuntu 22.04.

5.2.1. Configuration of threshold policies

As part of the computational study, we seek to assess the applicability of static and dynamic threshold policies. In particular, the question arises as to whether good results can already be achieved using comparably simple policies instead of costly solution approaches. In order to accurately investigate the performance of threshold policies, it remains to find appropriate values for the steering parameters $\alpha, \eta \in [0, 1]$ defining the threshold policies sTH(α) and dTH(η). In both cases we expect the value of parameter d^{ema} to have an impact on a good choice for α and η.

Starting with the static threshold policy we evaluate values for α in $\{i \cdot 10^{-1} \mid i = 0, \ldots, 10\}$. Figure 5 summarizes our results examining the average relative gap between the objective values obtained using sTH(α) and the discretization approach for the instance data in $\mathcal{I}_{CS}$. We observe significant differences in the gaps for the considered values of α depending on d^{ema}. For example, for $d^{\text{ema}} = 30$ the lowest gap of 11.72% is achieved for $\alpha = 0.4$. In contrast, for instances with $d^{\text{ema}} = 250$, maintaining $\alpha = 0.4$ results in an average gap of 182.87%, whereas for $\alpha = 0.9$ the gap is reduced to 17.42%. We observe similar effects for all different values of d^{ema}. Therefore, we decide to define a static threshold policy with parameter α depending on d^{ema} such that the observed average relative gap is minimized for each realization of d^{ema}. We refer to this policy as sTH($\alpha(d^{\text{ema}})$). Specifically, α is set to 0.4, 0.6, 0.7, and 0.9 for the considered d^{ema} values of 30, 50, 100, and 250, respectively.

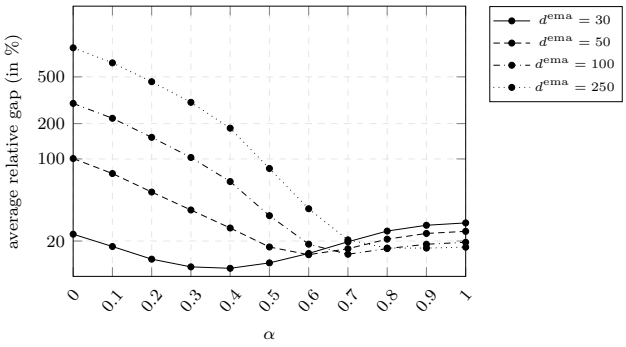


Figure 5. Average relative gap between sTH(α) and the discretization approach.

When turning our attention to the configuration of the dynamic threshold policy defined by the parameter η we observe similar effects and proceed analogously but evaluate the results for values of η in $\{i \cdot 10^{-1} \mid i = 0, \ldots, 10\} \cup \{i \cdot 10^{-2} \mid i = 1, \ldots, 9\}$. Thus, we additionally consider values of η from the subinterval $[0, 0.1]$, since for large values of d^{ema}, policies that involve taking only low risks for EMAs already appear promising beforehand, and we aim to examine this in more detail. Figure 6 shows the average relative gaps between the objective values obtained using dTH(η) and the discretization approach. Again there is a high dependency between the parameter d^{ema} and the gaps obtained. We follow the same approach as in the configuration of the static threshold policy and choose η depending on d^{ema} such that the gap is minimized for each value of d^{ema} and receive the dynamic threshold policy dTH($\eta(d^{\text{ema}})$). We set η to 0.3, 0.08, 0.03, and 0.01 for the considered d^{ema} values of 30, 50, 100, and 250, respectively.

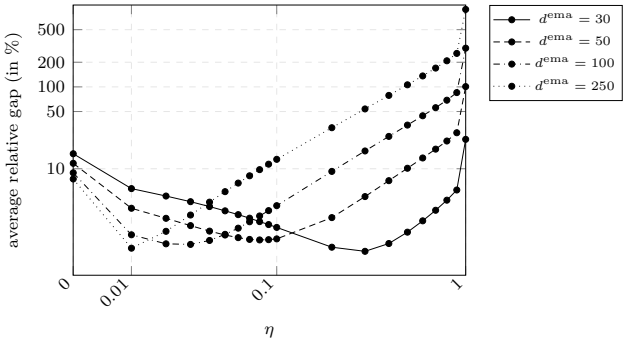


Figure 6. Average relative gap between dTH(η) and the discretization approach.

5.2.2. Robustness of the ADP approach

We want to investigate whether the ADP approach is robust against stochasticity during the learning process as outlined in Section 4.1. As the coefficients of the approximations are determined using random samples it is a legitimate question, whether it suffices to

derive the decision policy from a single learning process. To investigate this issue, we perform multiple applications of all ADP configurations (learning and applying the derived policy) to individual instances using different seeds for the learning processes. The policies obtained are evaluated by identical simulation runs and we consider the objective value for each instance and each application (i.e. learning process) of the ADP. The intention is to examine how strongly the received objective values per instance vary if the learning process is ceteris paribus repeated with different samples.

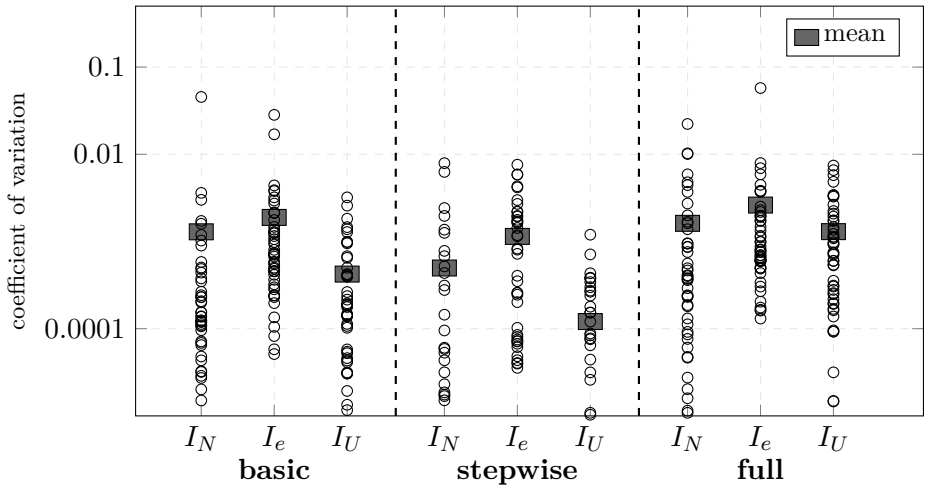


Figure 7. Coefficient of variation of the objective value for 10 independent ADP learning runs.

Figure 7 summarizes the results for all ADP configurations. It shows the coefficient of variation of the objective value for 10 independent ADP learning runs. We consider the coefficient of variation instance-wise for a random sample of 150 instances with 50 instances per instance type.

It can be seen that for all ADP configurations the received objective values do not differ substantially across various learning processes. Over all ADP configurations and instance classes we observe coefficients of variation of about 0.06 in maximum. Thereby, it can be seen that these maximum values result from individual outliers. Further, the average coefficient of variation per instance set and ADP configuration is below 0.003 with the smallest average coefficient of variation resulting from the stepwise ADP for $\mathcal{I}_U$ with 0.0001 which might be attributed to the rather narrow configuration of the stepwise ADP. Thus, the independent ADP applications may provide different decision policies which, however, provide similar objective values in average over the simulation runs. Due to these observations it is justifiable to apply the ADP (learn the coefficients) only once in our solution procedure for any chosen ADP configuration as we do not face a high risk of obtaining a particularly good or bad policy by chance. Based on these considerations, it is easy to argue that we can consider the ADP as a valid solution method in the form proposed in Section 4.1.

5.2.3. Running times

The running time analysis examines the time it takes to derive a decision policy. Due to the fact that the threshold policies have (almost) zero running time the analysis is only done for the ADP approaches, comprising all configurations, and the discretization approach.

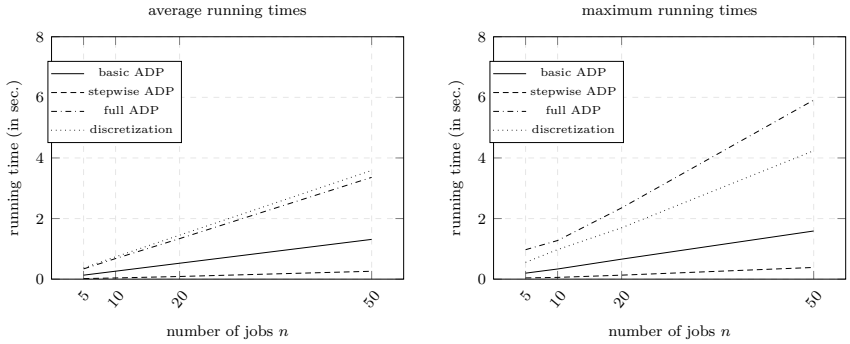


Figure 8. Running times for ADP approaches and discretization with $\zeta = 500$ and $\theta = 3$.

Figure 8 shows the average and maximum running times using the different ADP configurations depending on the number of jobs n. At this point, we do not show the running time results for the instance sets $\mathcal{I}_N$, $\mathcal{I}_e$ and $\mathcal{I}_U$ individually, as the running times are very similar for all instance sets. The figures clearly reveal the linear increase of running times with increasing n. Further, as one would expect the running times using the ADP increase with the number of basis functions used for the approximation of Bellman values. With 18 basis functions in total the full ADP uses the most basis functions and consequently has the longest running times.

The running times of the discretization approach depend highly on the chosen discretization parameters ζ and θ as outlined in Section 4.2. For the parameter assignment $\zeta = 500$ and $\theta = 3$ we observe average running times in the range of the full ADP running times, e.g. it takes around 3.5 sec in average to solve instances of the size $n = 50$. The maximum running times using this discretization, however, are a little lower than these of the full ADP approach. Further, we observe as one would expect a linear dependency between the discretization parameters and the running times which is not displayed in Figure 8.

5.2.4. Policy performance

In the following, we analyze the results with a focus on the quality of the derived decision policies. The primary analysis is based on the objective value. Additionally, we consider other stochastic aspects of the simulation results, such as the variation of the total completion time, to investigate whether the objective value significantly deviates from individual

observed values. This is particularly important from the user's perspective, as the decision policy is ultimately determined just in time for the upcoming project and is only applied once or a few times (in the case of identical production processes connected in sequence). Thereby, the stability in the observed objective values does, for sure, highly depend on the stochasticity in the instance at hand. However, it seems interesting to investigate how the different policies behave in terms of stability. Still, it should be remembered that the underlying optimization model does not aim for low variances or even require them.

The strongest result can be observed when looking at the results for the decision policies resulting from the discretization approach. We observe that, for more than 95% of all instance classes considered, the total completion time is lowest when using the policies derived by discretization. For the remaining instance classes, the difference in completion time compared to the lowest observed time is negligible. This goes along with our expectation of the discretization approach converging to the optimal solution. For this reason, we focus on the average relative gap between the objective values using the other competitors and the objective value resulting from discretization. Table 3 summarizes the results for the individual instance sets.

	$\mathcal{I}_N$	$\mathcal{I}_e$	$\mathcal{I}_U$	$\mathcal{I}_{CS}$
full ADP	0.45%	0.58%	0.54%	0.5%
basic ADP	1.07%	0.85%	0.95%	0.99%
stepwise ADP	2.81%	8.93%	0.43%	4.52%
sTH($\alpha(d^{\text{ema}})$)	13.29%	19.41%	10.61%	14.97%
dTH($\eta(d^{\text{ema}})$)	0.72%	2.15%	0.47%	1.15%

Table 3. Average relative gaps to the objective value resulting from discretization.

We can see that the full ADP yields a gap of 0.45%, 0.58%, and 0.54% for $\mathcal{I}_N$, $\mathcal{I}_e$, and $\mathcal{I}_U$, respectively, on average. Thus, the performance of the full ADP is good for all considered instance sets. In total, the full ADP even achieves a lower objective value in 5.77%, 3.46%, and 2.25% of the instances with normal, exponential, and uniform deterioration, respectively. In contrast, the highest average gap using the basic ADP is observed for $\mathcal{I}_N$ with an average gap of 1.07% and using the stepwise ADP for $\mathcal{I}_e$ with 8.93%. Thus, it can be concluded that indicator functions alone are effective to explain the course of the Bellman values for some instance sets and the same holds for the combination of polynomial, exponential, and square root basis functions. However, in order to achieve good results for all instance classes we need both types of basis functions as implemented in the full ADP. In fact, the full ADP yields the lowest gaps over all solution approaches among the instance classes $\mathcal{I}_N$ and $\mathcal{I}_e$ individually and only negligibly larger gaps than the stepwise ADP and dTH($\alpha(d^{\text{ema}})$) for the instance set $\mathcal{I}_U$. Considering the entire instance data set, the full ADP has the lowest average gap of 0.5%. In addition, in around 35% of the instance classes considered in the computational study we observe gaps smaller than 0.1%. It can therefore be concluded that the full ADP closely approximates the best

known solution.

As already anticipated at the beginning of this section sTH($\alpha(d^{\text{ema}})$) does not provide convincing results. We observe an average gap of 14.97% for the instance data $\mathcal{I}_{CS}$. Even though the duration of the EMAs is already incorporated in the sTH($\alpha(d^{\text{ema}})$) policy it is obvious to see that there is almost no chance of finding a competitive policy if the job properties and, thus, the deterioration to be expected is not taken into account in the decisions concerning PMAs. In particular, even sTH($\alpha(d^{\text{ema}})$) and, thus, any static threshold policy as defined in Section 4.3 may only serve as a satisfying benchmark policy if there are no information about job deteriorations available.

Meanwhile, we can see that dTH($\eta(d^{\text{ema}})$) can handle the instances much better as it has significantly smaller average gaps for all instance sets than sTH($\alpha(d^{\text{ema}})$). This result is not really surprising, since dTH($\eta(d^{\text{ema}})$) takes into account specific information about the next job which are not incorporated by sTH($\alpha(d^{\text{ema}})$). However, we want to highlight the strong dominance of dTH($\eta(d^{\text{ema}})$) over sTH($\alpha(d^{\text{ema}})$) that can be inferred from the values in Table 3. On the other hand, dTH($\eta(d^{\text{ema}})$) still does not yield lower average gaps for the instance sets $\mathcal{I}_N$ and $\mathcal{I}_e$ than the full ADP configuration and the minor superiority in $\mathcal{I}_U$ does not compensate for this. Especially for instance set $\mathcal{I}_e$ the average gaps differ by more than 1.5% while the average gap using the full ADP is just 0.58%.

Another interesting aspect with regard to the suitable configuration of the considered solution approaches is the approximation error for the state representing the initial situation. It is the difference between the true value of a heuristic's policy (usually obtained via simulation) and the approximation of the Bellman value calculated by the heuristic on the first stage corresponding to the initial ML. For the best performing ADP approach, the full ADP, we observe average absolute gaps of less than 1.82% for all instance sets. Meanwhile, for the discretization the average gaps range between 0.07% and 0.57% for the different instance sets. While the results for the discretization method are as one would expect given its construction, the great results for the full ADP approach are noteworthy and once more confirm the successful configuration of the ADP.

We now take a closer look at additional statistical parameters in order to gain a better impression of the results starting with the standard deviation of the total completion time over the simulation runs. To make the data comparable we use the coefficient of variation (standard deviation/mean).

Figure 9 shows the development of the average coefficient of variation for each instance set depending on the number of jobs n. The courses in the graphs show that the coefficients of variation tend to zero with increasing n for the instance sets $\mathcal{I}_N$ and $\mathcal{I}_U$ for all considered decision policies. Thereby, the results tend with an almost similar slope towards zero for all approaches considered. The lowest coefficients of variation are observed for the instances from $\mathcal{I}_N$ with $n = 50$ with around 0.03 in average over all solution approaches

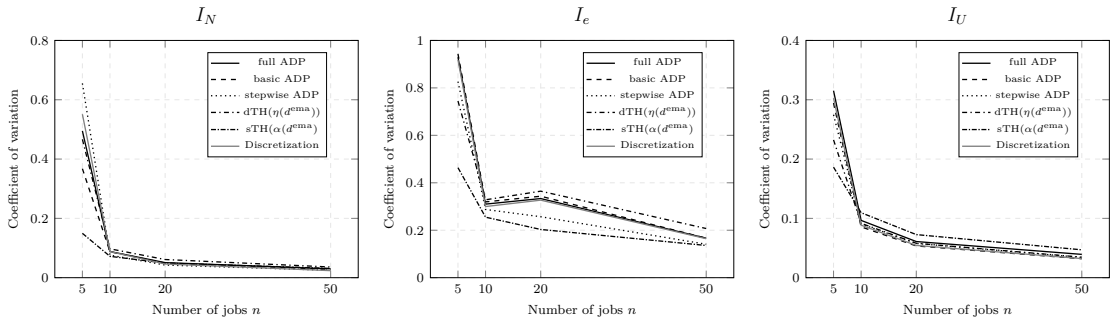


Figure 9. Average coefficient of variation of the total completion times resulting from simulation.

whereby it is minimal for the discretization approach with 0.023. The ADP approaches provide decision policies with small coefficients of variations for all instance classes and especially for these from the instance sets $\mathcal{I}_N$ and $\mathcal{I}_U$ and sufficiently high values of n.

For the instance set $\mathcal{I}_e$ we see slightly higher coefficients of variation. Except for dTH($\eta(d^{\text{ema}})$) and the stepwise ADP we observe a slight increase in the average coefficients of variation for $n = 20$ (resulting from outlier instances) that deviate from the monotonically decreasing trend observed for $\mathcal{I}_N$ and $\mathcal{I}_U$ as n increases. Concerning the higher coefficients of variation in general, we were able to see that the distribution properties of the exponential distribution cause this behavior. Due to high deteriorations with low probability the overall probability of any job causing a high deterioration is low but not equal to zero and at the same time it increases with the number of jobs. As a consequence, unpredictable EMAs become more likely which leads to higher total completion times in some individual simulation runs and, thus, to a higher variability of the observed values. Note that sTH($\alpha(d^{\text{ema}})$) remains significantly less affected by this.

Thus, the discretization approach and the full ADP approach in particular deliver good policies with respect to the objective value and at the same time the variance in the total completion times of the individual simulation runs is reasonably low. These two heuristics based on the DP formulation provide objective values close or equal to the best objective value found and comparable variances.

Finally, we take a look at the value at risk (VaR) and the conditional value at risk (cVaR). As we have a minimization objective, the value at risk to level τ corresponds to the $(1-\tau)$ quantile, i.e. with a probability of $(1-\tau)$, the expected total completion time of the simulation runs does not exceed the VaR. The cVaR at confidence level τ_c corresponds to the expected total completion time in the worst $\tau_c \cdot 100\%$ of the simulation runs.

Since all following considerations are based on the results for the individual simulation runs we consider the results instance-wise to not loose accuracy due to calculating average values repeatedly. To ensure comparability, we consider the VaR and the cVaR always in relation to the minimum observed VaR and cVaR, respectively. Table 4 shows the average

VaR to the level of confidence $\tau = 0.1$						
	$\mathcal{I}_N$		$\mathcal{I}_e$		$\mathcal{I}_U$	
	mean	median	mean	median	mean	median
full ADP	0.70%	0.11%	0.92%	0.24%	0.82%	0.20%
basic ADP	1.22%	0.15%	1.10%	0.29%	1.17%	0.29%
stepwise ADP	2.92%	0.85%	6.83%	2.10%	0.51%	0.03%
dTH($\eta(d^{\text{ema}})$)	0.74%	0.04%	3.11%	1.10%	0.74%	0.09%
sTH($\alpha(d^{\text{ema}})$)	11.57%	4.42%	15.25%	6.98%	13.92%	7.68%
Discretization	0.29%	0.00%	1.39%	0.00%	0.27%	0.00%

cVaR to the level of confidence $\tau_c = 0.1$						
	$\mathcal{I}_N$		$\mathcal{I}_e$		$\mathcal{I}_U$	
	mean	median	mean	median	mean	median
full ADP	2.67%	0.36%	4.53%	0.89%	2.77%	0.62%
basic ADP	2.88%	0.38%	6.16%	0.89%	2.08%	0.32%
stepwise ADP	5.48%	0.76%	8.83%	2.47%	0.97%	0.02%
dTH($\eta(d^{\text{ema}})$)	3.28%	1.27%	7.59%	4.34%	1.13%	0.42%
sTH($\alpha(d^{\text{ema}})$)	12.28%	3.80%	12.10%	5.25%	15.61%	9.03%
Discretization	3.76%	0.01%	3.45%	0.22%	1.28%	0.00%

Table 4. Average and median gap to the minimum VaR and cVaR for the different solution approaches.

and median gap of the VaR and cVaR with levels of confidence $\tau = 0.1$ and $\tau_c = 0.1$.

For the objective values achieved by the discretization policy we see that the VaR in average only slightly deviates from the smallest observed VaR and the median is zero for all instance sets considered. Thus, for at least 50% of the considered instances the lowest VaR is obtained when using the decision policy derived from discretization. We observe comparable results for the median gaps of the cVaRs but simultaneously the average gap to the smallest reached cVaR is with around 3.8% for the instance set $\mathcal{I}_N$ higher than for the VaR. From this, we conclude that for individual instances the expected objective value in the worst 10% of scenarios exceeds the cVaR obtained when using the decision policy of at least one of the other policies. Thus, it can be concluded that individual simulation runs have high total completion times. For $\mathcal{I}_N$, the full ADP shows the smallest average gap (2.67%) to the lowest observed cVaR among all approaches. For the instance set $\mathcal{I}_U$, the stepwise ADP, in turn, attains the lowest average gap of 0.97%. Consequently, we can see that the different ADP policies perform satisfactorily in the worst case simulation runs. However, we can not identify one of the presented ADP configurations or the discretization as performing outstandingly well for all instance sets. Therefore, it might be reasonable for the user to further asses which kind of deterioration one may expect and which approach promises the best results. Table 4 confirms once more the clear inferiority of sTH($\alpha(d^{\text{ema}})$). For the instance set $\mathcal{I}_U$ the expected total completion time in the worst 10%, the cVaR, is in average 15.61% higher than the lowest reached cVaR. Further, the median, exceeds the observed median values for all other solution approaches. We observe comparable results for the VaR. The average gaps of the (c)VaR using dTH($\eta(d^{\text{ema}})$) are significantly lower than those resulting from sTH($\alpha(d^{\text{ema}})$). For the instance set $\mathcal{I}_U$ we even observe lower average gaps to the minimum VaR and the minimum cVaR than when using the full ADP. In all other cases, the gaps are higher than for the full ADP approach. Overall, the risk of expecting high objective values when using dTH($\eta(d^{\text{ema}})$) is manageable, but higher than for the full ADP when looking at the instance data $\mathcal{I}_{CS}$ as a whole.

5.2.5. Summary of results and practical implications

Finally, we briefly summarize the results for the different approaches and provide recommendations for practitioners. The full ADP and the discretization approach achieve the best results with respect to almost all performance indicators considered, with the discretization being slightly better. In practice, the two approaches mainly differ in the availability of the PMA decision policy. When using the discretization, PMA decisions prior to any job are specified for all discrete MLs in $[0, l^{\max}]$, resulting – with our parameters – in a memory requirement of $25{,}000 \cdot n$ bits per instance. Storing state evaluations instead of decisions increases this to $25{,}000 \cdot 32 \cdot n$ bits. In contrast, the ADP approach approximates the Bellman functions using 18 coefficients, requiring $18 \cdot 32 \cdot n$ bits of memory, with decisions derived from a just-in-time evaluation for two MLs, $l^{\max}$ and the current ML. Overall, the ADP method requires only 2.3% and 0.07%, respectively, of the memory needed for the discretization approach, making it an attractive option, especially for local execution on smart devices.
In contrast, the proposed threshold policies require no high computational effort. Compared to the static threshold policy, the dynamic threshold policy with parameters adjusted to d^{ema} achieves remarkably good results and, given an average optimality gap not exceeding 2.15% and its simplicity, can be highly recommended for practical use.

6. Conclusion and outlook

In this paper, we present different approaches to solve a maintenance scheduling problem on a single machine with stochastic, continuous deterioration caused by job processing. In contrast to most of the literature, the job sequence is fixed. The central decision problem is when to schedule PMAs, allowing full flexibility as long as PMAs do not interrupt jobs, which is rarely considered in existing work. EMAs occur when the ML drops below zero, delay the completion of the current and all subsequent jobs, and take longer than PMAs. After formally defining the NP-hard optimization problem of minimizing the expected total completion time, we propose a DP formulation. Since an exact evaluation is infeasible, we introduce several heuristic approaches, namely a discretization approach, an ADP approach with three configurations, and threshold policies. A simulation-based computational study assesses the quality of the resulting decision policies. The results show that the discretization approach approximates an optimal solution, the full ADP policy performs well with manageable running times. Further, the dynamic threshold policy is worth highlighting as it performs reasonably well and can be recommended for practitioners due to its simplicity.

The presented methods based on the DP formulation can be easily adopted to different

problem variants. It is conceivable, e.g. that the duration of the PMAs is stochastic, that delays arise caused by setup times for PMAs or that jobs must either be restarted as soon as an EMA occurs or that they may be interrupted for PMAs. By adding these features, the valuation basis of the procedure changes, but the proposed framework can be adopted. Further, it might also be interesting to consider a problem variant where deteriorations do not follow static distributions. This scenario might be reasonable, e. g., if the distribution depends on the experiences in the production process so far. If the actual deterioration distribution depends only on the job and the current ML of the machine this setting can be easily incorporated into the model presented in this paper.

So far, we do not exploit the continuous structure of the problem. As the decision in each pre-decision node is binary, there is not much to gain from a closed-form solution, which often considerably accelerates the solution of problems with continuous decisions. More promising extensions include the consideration of problem variants in which analytically tractable components arise from alternative distribution functions, or the restriction to a given policy class. As, for example, the dynamic threshold performs very well, future work could try to analytically determine optimal parameter values for similar policies. This opens up a broad field for future research in which the continuous structure of the ADP developed in this paper may prove beneficial.

As further future research, it is close at hand that the problem shall be extended so that the job sequence needs to be decided in addition to the scheduling of the PMAs. The approach presented in this paper serves as a basis for finding and evaluating suitable candidates for the job sequence using a meta heuristic approach. In addition, structural characteristics of decision policies can be derived from possible further computational studies, which can be helpful in real-world applications.

References

Ahmad, R., Kamaruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial application. *Computers & Industrial Engineering*, *63*(1), 135–149.

Arts, J., Boute, R.N., Loeys, S., van Staden, H.E. (2025). Fifty years of maintenance optimization: Reflections and perspectives. *European Journal of Operational Research*, *322*(3), 725–739.

Barto, A.G., Sutton, R.S., Anderson, C.W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-13*(5), 834–846.

Bock, S., Briskorn, D., Horbach, A. (2012). Scheduling flexible maintenance activities subject to job-dependent machine deterioration. *Journal of Scheduling*, *15*(5), 565–578.

Briskorn, D., Gönsch, J., Thiemeyer, A. (2024). Scheduling maintenance activities subject to stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *319*(1), 62–78.

Brucker, P. (2004). Scheduling Algorithms. Springer. 4th ed.

Chen, J.S. (2008). Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan. *European Journal of Operational Research*, *190*(1), 90–102.

Cramér, H. (1999). Mathematical methods of statistics. volume 26. Princeton university press.

Cui, W.W., Lu, Z. (2017). Minimizing the makespan on a single machine with flexible maintenances and jobs' release dates. *Computers & Operations Research*, *80*, 11–22.

De Jonge, B., Scarf, P.A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*(3), 805–824.

Dehnar Saidy, H.R., Taghavi-Fard, M.T. (2008). Study of scheduling problems with machine availability constraint. *Journal of Industrial and Systems Engineering*, *1*(4), 360–383.

Detti, P., Nicosia, G., Pacifici, A., de Lara, G.Z.M. (2019). Robust single machine scheduling with a flexible maintenance activity. *Computers & Operations Research*, *107*, 19–31.

Drozdowski, M., Jaehn, F., Paszkowski, R. (2017). Scheduling position-dependent maintenance operations. *Operations Research*, *65*(6), 1657–1677.

Edis, E.B., Oguz, C., Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, *230*(3), 449–463.

Gawiejnowicz, S., Kononov, A. (2010). Complexity and approximability of scheduling resumable proportionally deteriorating jobs. *European Journal of Operational Research*, *200*(1), 305–308.

Geurtsen, M., Didden, J., B.H.C., Adan, J., Atan, Z., Adan, I. (2023). Production, maintenance and resource scheduling: A review. *European Journal of Operational Research*, *305*(2), 501–529.

Gordon, V.S., Potts, C.N., Strusevich, V.A., Whitehead, J.D. (2008). Single machine scheduling models with deterioration and learning: handling precedence constraints via priority generation. *Journal of Scheduling*, *11*, 357–370.

Grigoriu, L., Briskorn, D. (2017). Scheduling jobs and maintenance activities subject to job-dependent machine deteriorations. *Journal of Scheduling*, *20*(2), 183–197.

Gromicho, J.A.S., Van Hoorn, J.J., Saldanha-da Gama, F., Timmer, G.T. (2012). Solving the job-shop scheduling problem optimally by dynamic programming. *Computers & Operations Research*, *39*(12), 2968–2977.

Ji, M., Cheng, T.E. (2010). Scheduling with job-dependent learning effects and multiple rate-modifying activities. *Information Processing Letters*, *110*(11), 460–463.

Ji, M., He, Y., Cheng, T.E. (2007). Single-machine scheduling with periodic maintenance to minimize makespan. *Computers & Operations Research*, *34*(6), 1764–1770.

Jiang, D.R., Powell, W.B. (2015). Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming. *INFORMS Journal on Computing*, *27*(3), 525–543.

Konda, V.R., Tsitsiklis, J.N. (2000). Actor-critic algorithms, in: Solla, S.A., Leen, T.K., Müller, K. (Eds.), Advances in neural information processing systems, MIT Press, Cambridge, MA. pp. 1008–1014.

Kubzin, M.A., Strusevich, V.A. (2006). Planning machine maintenance in two-machine shop scheduling. *Operations Research*, *54*(4), 789–800.

Lee, C.Y. (1999). Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research*, *114*(2), 420–429.

Lee, C.Y., Lei, L., Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research*, *70*, 1–41.

Lee, C.Y., Leon, V.J. (2001). Machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, *128*(1), 119–128.

Lee, C.Y., Lin, C.S. (2001). Single-machine scheduling with maintenance and repair rate-modifying activities. *European Journal of Operational Research*, *135*(3), 493–513.

Lee, J.Y., Kim, Y.D., Lee, T.E. (2018). Minimizing total tardiness on parallel machines subject to flexible maintenance. *International Journal of Industrial Engineering*, *25*(4), 472–489.

Lee, W.C., Wang, J.Y., Lee, L.Y. (2015). A hybrid genetic algorithm for an identical parallel-machine problem with maintenance activity. *Journal of the Operational Research Society*, *66*(11), 1906–1918.

Lei, D., Liu, M. (2020). An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. *Computers & Industrial Engineering*, *141*, Article 106320.

Lewis, F.L., Vrabie, D., Syrmos, V.L. (2012). Optimal control. John Wiley & Sons.

Luo, W., Xu, Y., Tong, W., Lin, G. (2019). Single-machine scheduling with job-dependent machine deterioration. *Journal of Scheduling*, *22*(6), 691–707.

Ma, Y., Chu, C., Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, *58*(2), 199–211.

Mellouli, R., Sadfi, C., Chu, C., Kacem, I. (2009). Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times. *European Journal of Operational Research*, *197*(3), 1150–1165.

Mor, B., Mosheiov, G. (2012). Scheduling a maintenance activity and due-window assignment based on common flow allowance. *International Journal of Production Economics*, *135*(1), 222–230.

Mor, B., Mosheiov, G. (2015). Scheduling a deteriorating maintenance activity and due-window assignment. *Computers & Operations Research*, *57*, 33–40.

Mosheiov, G., Oron, D. (2021). A note on scheduling a rate modifying activity to minimize total late work. *Computers & Industrial Engineering*, *154*, Article 107138.

Mosheiov, G., Sarig, A. (2009). Scheduling a maintenance activity and due-window assignment on a single machine. *Computers & Operations Research*, *36*(9), 2541–2545.

Mosheiov, G., Sarig, A., Strusevich, V.A., Mosheiff, J. (2018). Two-machine flow shop and open shop scheduling problems with a single maintenance window. *European Journal of Operational Research*, *271*(2), 388–400.

Mosheiov, G., Sidney, J.B. (2010). Scheduling a deteriorating maintenance activity on a single machine. *Journal of the Operational Research Society*, *61*(5), 882–887.

Nascimento, J.M., Powell, W.B. (2009). An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Mathematics of Operations Research*, *34*(1), 210–237.

Ogunfowora, O., Najjaran, H. (2023). Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. *Journal of Manufacturing Systems*, *70*, 244–263.

Perez-Gonzalez, P., Fernandez-Viagas, V., Framinan, J.M. (2020). Permutation flowshop scheduling with periodic maintenance and makespan objective. *Computers & Industrial Engineering*, *143*, Article 106369.

Phosavanh, J., Oron, D. (2024). Two-agent single-machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, *312*(3), 866–876.

Pinedo, M. (2009). Planning and scheduling in manufacturing and services. Springer. 2nd ed.

Powell, W.B. (2007). Approximate Dynamic Programming: Solving the curses of dimensionality. volume 703. John Wiley & Sons.

Powell, W.B. (2022). Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions. John Wiley & Sons.

Randall, N., Basciftci, B. (2025). Risk-averse contextual predictive maintenance and operations scheduling with flexible generation under wind energy uncertainty. *European Journal of Operational Research*, *327*(1), 174–190.

Renke, L., Piplani, R., Toro, C. (2021). A review of dynamic scheduling: Context, techniques and prospects, in: Implementing Industry 4.0: The Model Factory as the Key Enabler for the Future of Manufacturing. Springer, pp. 229–258.

Rustogi, K., Strusevich, V.A. (2014). Combining time and position dependent effects on a single machine subject to rate-modifying activities. *Omega*, *42*(1), 166–178.

Sanlaville, E., Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, *35*(9), 795–811.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, *121*(1), 1–15.

Scott, W.R., Powell, W.B. (2012). Approximate Dynamic Programming for Energy Storage with New Results on Instrumental Variables and Projected Bellman Errors. Technical Report. Princeton University. Princeton, NJ. Working Paper.

Sharma, A., Yadava, G.S., Deshmukh, S.G. (2011). A literature review and future perspectives on maintenance optimization. *Journal of Quality in Maintenance Engineering*, *17*(1), 5–25.

Shen, Z.J.M., Xie, J., Zheng, Z., Zhou, H. (2023). Dynamic scheduling with uncertain job types. *European Journal of Operational Research*, *309*(3), 1047–1060.

Strusevich, V.A., Rustogi, K. (2017). Scheduling with time-changing effects and rate-modifying activities. Springer.

Sun, K., Li, H. (2010). Scheduling problems with multiple maintenance activities and non-preemptive jobs on two identical parallel machines. *International Journal of Production Economics*, *124*(1), 151–158.

Sun, X., Liu, T., Geng, X.N., Hu, Y., Xu, J.X. (2023). Optimization of scheduling problems with deterioration effects and an optional maintenance activity. *Journal of Scheduling*, *26*(3), 251–266.

Sutton, R.S. (1984). Temporal Credit Assignment in Reinforcement Learning. Ph.D. thesis. University of Massachusetts Amherst. Amherst, MA.

Wiering, M.A., Van Otterlo, M. (2012). Reinforcement learning. *Adaptation, learning, and optimization*, *12*(3), 729.

Xu, S., Dong, W., Jin, M., Wang, L. (2020). Single-machine scheduling with fixed or flexible maintenance. *Computers & Industrial Engineering*, *139*, Article 106203.

Yang, D.L., Hung, C.L., Hsu, C.J., Chern, M.S. (2002). Minimizing the makespan in a single machine scheduling problem with a flexible maintenance. *Journal of the Chinese Institute of Industrial Engineers*, *19*(1), 63–66.

Yang, D.L., Yang, S.J. (2013). Unrelated parallel-machine scheduling problems with multiple rate-modifying activities. *Information Sciences*, *235*, 280–286.

Yang, S.J. (2013). Unrelated parallel-machine scheduling with deterioration effects and deteriorating multi-maintenance activities for minimizing the total completion time. *Applied Mathematical Modelling*, *37*(5), 2995–3005.

Yang, S.J., Yang, D.L. (2010). Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. *Omega*, *38*(6), 528–533.

Zhao, C.L., Tang, H.Y., Cheng, C.D. (2009). Two-parallel machines scheduling with rate-modifying activities to minimize total completion time. *European Journal of Operational Research*, *198*(1), 354–357.

Zhong, X., Ou, J., Wang, G. (2014). Order acceptance and scheduling with machine availability constraints. *European Journal of Operational Research*, *232*(3), 435–441.

D. Article IV – Integrated scheduling of jobs and maintenance activities on a single machine exposed to stochastic job-dependent machine deterioration

Title: Integrated scheduling of jobs and maintenance activities on a single machine exposed to stochastic job-dependent machine deterioration
Authors: Arter, A., and Briskorn, D.
Status: Working Paper
Citation: Arter, A., Briskorn, D. (2026). Integrated scheduling of jobs and maintenance activities on a single machine exposed to stochastic job-dependent machine deterioration. Working Paper.

Integrated scheduling of jobs and maintenance activities on a single machine exposed to stochastic job-dependent machine deterioration

Antonia Arter[1], Dirk Briskorn[1]

[1]: Bergische Universität Wuppertal, Schumpeter School of Business and Economics

This research has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 452470135.

Abstract

In this paper, we consider a single machine scheduling problem with job-dependent deterioration and maintenance activities to compensate for the machine's deterioration. The deterioration is stochastic and degrades the machine state, measured by a numerical maintenance level. Excessive deterioration causes a machine breakdown demanding a time-consuming repair work resetting the machine state. To prevent a machine breakdown, planned maintenance activities can be scheduled. Thereby, decisions on conducting planned maintenance activities depend on the observable current maintenance level of the machine. The goal is to determine a job sequence and a decision policy for scheduling planned maintenance activities such that the expected weighted total completion time is minimized. We formalize the optimization problem and present different solution approaches. We develop an exact Branch and Bound framework based on a problem specific lower bound and complement it with several heuristic approaches, including beam search variants, tabu search and priority rules. The computational study shows that the exact approaches solve instances of manageable size to optimality, while the heuristic approaches provide promising solutions for instances with up to 50 jobs.

Keywords: Scheduling · Stochastic job-dependent machine deterioration · Maintenance activities · Branch and Bound · Single machine

1. Introduction

A common assumption in scheduling literature is unlimited machine availability over the entire planning horizon (cf. Brucker (2004) and Pinedo (2009)). In practice, however, continuous processing causes progressive deterioration of the machine, which may lead to either machine breakdowns or products of insufficient quality and in consequence to processing interruptions required for repair work. With regard to the longevity of the machine it is reasonable to conduct *maintenance activities* (MAs) at suitable points in time to improve the machine's state in advance. These, however, lead to interruptions of processing and, therefore, have an impact on the overall production efficiency. For this reason, the efficient integration of MAs into scheduling problems has received increasing attention in recent years.

We consider a single machine setting on which a given set of jobs has to be scheduled. Thereby, the machine is equipped with a numerical *maintenance level* (ML) measuring the state of the machine. The interpretation of the ML is quite intuitive with a higher ML inducing a better machine state. Additionally, an ML of zero indicates that the machine is not capable of further processing jobs due to an insufficient machine condition. In consequence, if a machine with ML zero continues to be used, e.g. if the current job has not been completed yet, the machine breaks down and a time-consuming repair work becomes necessary to restore the functionality of the machine.

Consequently, there are two factors that directly impact the ML. First, the processing of jobs causes direct load on the machine resulting in a reduction of the ML. In the problem at hand we consider stochastic deterioration. Thereby, the realized deterioration caused by a job can only be determined once the processing of the job is completed and the current ML is observed. This is because no information is available on how deterioration evolves during a job's deterministic processing time, and therefore it is not possible to infer the realized deterioration from interim MLs before the job is completed. This assumption is grounded in practical observations where deterioration results from material properties that can arise abruptly and unpredictably, such as irregularities in natural materials. Secondly, MAs can be conducted to restore the perfect machine condition by increasing the ML to $l^{\max}$. Thus, the observable ML at any time over the planning horizon ranges between zero and $l^{\max}$ and its development depends on the realized deteriorations and the conducted MAs.

We distinguish between two types of MAs that differ in how they are triggered. *Planned maintenance activities* (PMAs) can be voluntarily started at any time the machine is not currently processing a job, thus, at the beginning of the planning horizon, in between two consecutive jobs or after completing the last job. Note that no job can be interrupted to schedule a PMA. The decision if a PMA is conducted is based, among others, on the current ML of the machine. The number of possible PMAs is naturally bounded by the

number of permitted slots since scheduling multiple PMAs within one MA slot does not have any advantage. The duration of a PMA depends on the ML when starting the PMA and the duration is proportional to the conducted ML improvement. The more the ML is increased the longer the PMA takes. *Emergency maintenance activities* (EMAs) are automatically triggered as soon as the ML has reached zero and the current job is not completed yet or in other words the job's deterioration exceeds the available ML. As soon as the machine breaks down the EMA takes place and improves the ML to $l^{\max}$ with fixed duration exceeding the duration of a PMA improving the ML to the same extent. This extra duration can be interpreted as extra delay due to unforeseeable need of personnel and materials. Further, it is assumed that the progress of the interrupted job can be completely saved and the job can be completed after the EMA is finished. These assumptions in turn justify the extra duration of an EMA over a PMA. The option to schedule PMAs during the processing of jobs with the purpose to avoid EMAs is excluded. In particular, as we do not know how the deterioration is distributed over the deterministic processing time of a job, it can not even be determined with certainty at which time during the job's processing a PMA would have to be scheduled to avoid an EMA and, therefore, the desired relieving effect can not be achieved.

In this paper, we consider the problem of integrated scheduling of jobs and PMAs on a single machine such that the expected total weighted completion time is minimized. The expected total weighted completion time is chosen for two reasons. On the one hand, it is important for a company to be able to provide its customers with achievable delivery dates, which can only be estimated if the delay due to MAs is taken into account. On the other hand, from a more economic point of view, it is important for the producer to ensure liquidity by keeping track of the cash flows associated with the completion of individual jobs. The problem under consideration is an extension of the problems introduced by Briskorn et al. (2024) and Arter et al. (2026a), who consider PMA scheduling for fixed job sequences. The main contributions of this paper are threefold. First, we formalize the integrated sequencing and PMA planning problem under stochastic job-dependent deterioration. Second, we develop an exact *Branch and Bound* (BaB) algorithm based on a problem specific *lower bound* (LB). Third, we evaluate exact and heuristic solution methods in an extensive computational study.

The paper is organized as follows. In Section 2 we provide an overview of the literature related to job scheduling problems including machine unavailability. In Section 3 we formally define the considered optimization problem. In Section 4 we present different solution approaches with a focus on a suitable LB for an exact BaB approach. Section 5 evaluates and compares all presented solution approaches in a computational study before we close the work with a conclusion in Section 6.

2. Literature review

This section provides an overview of previous work on job scheduling with machine availability constraints. Especially over the last two decades, interest in problems involving machine unavailability, regardless of the cause, has increased, as such settings arise in various real-world applications. Further, we show how the problem considered in this work fits into the existing literature.

Given its relevance in both practical and theoretical terms, several surveys reviewing and categorizing the relevant literature are available. Lee et al. (1997), Sanlaville and Schmidt (1998), and Schmidt (2000) provide the earliest surveys in this field. Therein, the authors review literature dealing with different machine environments, such as single machine, parallel machines, or flow shop. From these surveys a general classification with regard to the machine's unavailability can be derived. It is distinguished between unavailability due to machine breakdowns (stochastic) or preventive maintenance (deterministic) including also foreseeable downtimes specified externally or run time limitations of the machine. Ma et al. (2010) provide a survey on literature dealing with deterministic machine availability constraints. In their work, machines are assumed to be unavailable within given time windows for different machine environments. Further, the authors classify problems with regard to the option of preemption. In machine environments with multiple machines a further distinction is made whether the processing of preempted jobs may be continued on another machine or not. Dehnar Saidy and Taghavi-Fard (2008) provide a survey covering settings in which unavailability periods may occur at random, and Kaabi and Harrath (2014) focus on parallel machine environments. In Strusevich and Rustogi (2017) a comprehensive introduction into scheduling with rate-modifying activities is given. These types of activities are close to the concept of maintenance, however, they are not necessarily needed to keep production running but have an impact on the machine's speed. The basic idea is to equate machine deterioration with a decline in the production rate, thereby creating incentives to maintain the machine in order to restore the machine performance.

To the best of our knowledge, the latest published surveys are those by Geurtsen et al. (2023) and Arter et al. (2026b). Both provide a comprehensive introduction into scheduling problems involving machine unavailability and maintenance. Thereby, Geurtsen et al. (2023) provide the first survey to jointly address the three scheduling dimensions of production, maintenance and resource scheduling. They introduce a classification scheme based on, among others, the type of deterioration, distinguishing between time-dependent, position-dependent, and job-dependent deterioration. In Arter et al. (2026b), scheduling problems from the literature considering state-dependent machine availability are examined. The authors present a systematic survey for classifying problems where machine availability is not primarily determined by the elapsed working time of the machine but rather by the current state of the machine. Furthermore, they develop a classification

scheme that can be integrated into the commonly used Graham notation (see Graham et al. (1979)) and captures some relevant maintenance properties in a four-field tuple. A fundamental distinction in their work is made between whether the deterministic processing times of jobs depend on the machine state and whether the deterioration per job is equal, linear in the job's deterministic processing time, or individual.

Given some undeniable conceptual overlaps we would like to draw attention to the surveys by Alaswad and Xiang (2017), De Jonge and Scarf (2020) and Arts et al. (2025). These are published in the field of maintenance optimization and, in principle, deal with similar problems as proposed in the surveys referred to above published in the field of operations research. The fundamental difference lies in the assumption that machine deterioration occurs independently of the jobs processed but follows a stochastic process over time modeling failure behavior of the machine (see Van Horenbeek et al. (2010)). The machine's failure behavior is, thereby approached by understanding the physics of failure and how the worsening of the machine can be predicted by monitoring technical machine properties like vibration or noise level or estimating the machine state over time. It is close at hand that the researchers in the field of maintenance optimization approach the problem from a more application oriented perspective. In recent publications the integration of artificial intelligence when predicting the machine state development over time has become more relevant (see, e.g., Ogunfowora and Najjaran (2023), Zhao et al. (2025)).

The single machine scheduling problem of minimizing the total weighted completion time without MAs is a well investigated problem and the optimal sequence is known to be the sequence of jobs ordered in *weighted shortest processing time* (WSPT) proven in Smith (1956). There are numerous extensions of this problem, such as including release dates, precedence constraints, sequence-dependent setup times, batch structures, or machine unavailability periods, most of which render the problem NP-hard (see Lenstra et al. (1977) and Allahverdi et al. (2008) for overviews).

For the remainder of this section we now concentrate on job scheduling problems involving MAs. The relevant literature investigates such problems across a variety of machine environments and the way in which maintenance is incorporated differs considerably. MA features can be identified that characterize how maintenance is taken into account. In the following, we structure the literature review along some reasonable features and discuss, for each of them some variants that have been considered in the literature.

A key feature is the flexibility in planning and conducting MAs. A general distinction can be made whether unavailability periods due to MAs are given as external constraints, see, e.g., Lee (1999), Gawiejnowicz and Kononov (2010), Zhong et al. (2014), Shabtay and Zofi (2018) and Choi and Park (2022), or whether the positioning of MAs is part of the scheduling problem like it is in the problem at hand. The related works of Mor and Mosheiov (2015) and Zhang et al. (2018) consider problem environments with MAs to be scheduled.

The next feature to be considered is the number of MAs over the planning horizon. A large stream of literature, including the works of Mor and Mosheiov (2015) and Hsu et al. (2011), restricts to at most one MA, while Kubzin and Strusevich (2006) consider a setting demanding exactly one MA. A special version of scheduling at most one MA is considered in Lee and Lin (2001), where a repair activity sets in automatically if a machine breakdown occurs prior to the scheduled MA. If so, the originally scheduled MA becomes obsolete. This assumption is comparable to the concept of EMAs considered in our problem setting, but less complex. In the setting considered in this work, the occurrence of an EMA does not imply that no further MA is required. After an EMA has been triggered and the machine has been restored, the machine state deteriorates anew as subsequent jobs are processed, so that additional MAs may become necessary. In other works there are limitations on the total number of MAs, see, e.g., Rustogi and Strusevich (2015) and Yang and Yang (2013). A special case with regard to the number of MAs is given by machine environments demanding periodic maintenance. Thereby, the periodic MAs may be flexibly scheduled such that the run time of the machine between two MAs does not exceed a given time limit as in Cui and Lu (2017), Mosheiov and Sarig (2009) and Wang et al. (2018). In Lee et al. (2015) and Lei and Liu (2020) MAs need to be conducted within a given time window for multi machine settings. Low et al. (2010) extend the consideration of periodic MAs by limiting simultaneously the number of processed jobs and the run time before the next MA has to be conducted. Zhou et al. (2021) investigate a single machine problem with fixed periodic unavailability periods due to preventive MAs.

The durations of MAs may be either fixed (see, e.g., Lee and Leon (2001), Hsu et al. (2011) and Ji et al. (2013)) or depend on the machine workload so far. Many authors assume a linear dependency between the MA duration and the elapsed time since the beginning of planning horizon or the completion of the last MA. This is a conceivable assumption for machine settings where the machine state deteriorates equally over time and the time it takes to restore the machine increases with a worse machine condition. Problems with time dependent MA duration are for example considered in Kubzin and Strusevich (2006), Mosheiov and Sidney (2010) and Zhang et al. (2018). Rustogi and Strusevich (2014) expand on these considerations by investigating MA durations depending on the elapsed time, the position of the MA within the job schedule and a further MA dependent factor. The latter is defined for a set of MAs that can be scheduled. Yang and Yang (2010) consider the reverse case, in which MA durations shorten as more MAs are carried out, due to learning effects.

The improvement type of the MAs is a further feature that can be used to categorize the reviewed literature. The first common assumption is, that an MA improves the machine state to its perfect state as in Xue et al. (2014), Sun et al. (2023), Sun and Geng (2019) and Liu et al. (2017). Alternatively, settings are considered in which the improvement of the MAs is individually, often referred to as a partial maintenance (see, e.g. Bock et al.

(2012) and Rustogi and Strusevich (2015)). In the problem setting investigated in this work, the PMA duration is linear in the ML improvement and the duration af an EMA is fixed.

The concept of rate-modifying activities, introduced above, has given rise to a large stream of literature that models the machine performance as declining with a worsening machine state, an effect often referred to as the aging effect. In such settings the processing times of jobs, therefore, depend on the machine state and MAs can be used to compensate for the occurred deterioration restoring the machine's performance. It arises an obvious trade-off between delaying jobs due to MAs and reducing processing times. As examples we can name the works of Lee and Lin (2001), Lee and Leon (2001), and Zhu et al. (2017) for single machine settings and the works of Zhao et al. (2009), Yang et al. (2014) and Woo et al. (2017) for multi machine settings. In this work, the processing times of jobs are deterministic and independent of the machine state. The actual time required to complete a job may nevertheless be prolonged whenever an EMA is triggered.

The optimization problem considered in this work is based on the problem introduced by Bock et al. (2012). The authors investigate a single machine scheduling problem integrating machine deterioration. Thereby, they assume that the current machine state is measured by a numeric ML that decreases during the processing of jobs due to deterioration. The extent of decrease is deterministic and possibly job-dependent. The authors introduce two settings that differ in the improvement achieved by an MA. In the partial maintenance case, an MA may raise the ML by an arbitrary feasible increment, whereas in the full maintenance case it restores the ML to its maximum level. In both cases, the duration of the MA is linear in the level improvement. Among others, the authors consider optimization problems defined using different objective functions, including the total completion time, and show that the full maintenance case is always NP-hard, unless the discrete deterioration is equal among all jobs. If so, the problem is solvable in polynomial time. Luo et al. (2019) extend the complexity analysis to the partial MA case and establish NP-hardness when total completion time is minimized. For the full MA case, Grigoriu and Briskorn (2017) address makespan minimization, give a linear-time 5/4-approximation, and generalize the setting to parallel machines, for which they analyze approximability and propose heuristics. Building on the model of Bock et al. (2012), Briskorn et al. (2024) and Arter et al. (2026a) limit themselves to fixed job sequences and minimize expected total completion time. The deterministic job-dependent deterioration is replaced by stochastic job-dependent deterioration underlying a discrete or continuous probability distribution, respectively. The paper at hand provides the direct extension of the just referred previous works by resolving the fixed job sequence assumption and solving the integrated problem of scheduling jobs and PMAs.

3. Problem definition

We consider a single machine setting with an observable ML as introduced in Section 1. An instance is defined by three numbers $l^{\max} \in \mathbb{N}^+$, $l_0 \in [0, l^{\max}]$, and $d^{\text{ema}} \geq l^{\max}$ reflecting the maximum ML, the initial ML, and the duration of EMAs, respectively, and a set $J = \{1, \ldots, n\}$ of jobs that need to be processed on the machine. Each job $j \in J$ has a deterministic processing time $p_j \geq 0$ and a weight $w_j \geq 0$ that are independent of the machine's condition. Further, each job deteriorates the machine to an individual extent. The deteriorations are stochastic and represented by random variables $\tilde{\Delta}_j$ for each job $j \in J$.

A solution to our problem consists of two components, namely a job sequence and a PMA policy. The job sequence σ is a permutation of the job set J and, thus, contains each job in J exactly once. We denote by $\sigma(k) \in J$ and $\sigma^{-1}(j) \in \{1, \ldots, n\}$ the job scheduled in the k-th position and the position of job $j \in J$, respectively. We denote by $\mathcal{S}_J$ the set of permutations of job set J.

To compensate for the machine's deterioration there are two types of MAs, PMAs and EMAs, both increasing the ML to $l^{\max}$. The two types of MAs further differ in their durations. The duration of EMAs equals the instance parameter d^{ema} while the duration of a PMA is linear in the ML improvement with improvement rate 1. While EMAs are triggered automatically, we decide about when to conduct PMAs. More specifically, for each job $j \in J$ we decide whether a PMA is conducted immediately before j starts or not. A PMA policy, thus, is a function $P : J \times [0, l^{\max}] \to \{0, 1\}$, $(j, l) \mapsto P(j, l)$. The image of an arbitrary tuple (j, l) under P indicates whether a PMA is conducted ($P(j, l) = 1$) or not ($P(j, l) = 0$) if the ML at j's start is l without a PMA. The set of all PMA policies for an instance with job set J is denoted as $\mathcal{P}(J)$.

A solution consisting of job sequence σ and PMA policy P affects start times and completion times of jobs and the ML over time. We introduce the observable ML immediately after completing job $j = 1, \ldots, n$ under job sequence σ and PMA policy P as a dependent random variable $\tilde{L}_j(\sigma, P)$. For simplicity, we define $\sigma(0) = 0$ and consider the random variable $\tilde{L}_0(\sigma, P)$ reflecting the initial ML which is known to be l_0 and independent of σ and P. Random variables $\tilde{L}_1(\sigma, P), \ldots, \tilde{L}_n(\sigma, P)$ together with σ and P imply random variables $\tilde{M}_j^{\text{PMA}}(\sigma, P)$ and $\tilde{M}_j^{\text{EMA}}(\sigma, P)$ for each job $j \in J$ reflecting the delay of all following jobs caused by a PMA immediately before job j and the delay of the completion of job j and all following jobs caused by EMAs during the processing of job j, respectively. Note that $\tilde{M}_j^{\text{PMA}}(\sigma, P) = 0$ and $\tilde{M}_j^{\text{EMA}}(\sigma, P) = 0$ if no PMA and EMA is conducted.

Similarly, we consider $\tilde{C}_j(\sigma, P)$ be the random variable representing the completion time of job j in sequence σ when applying PMA policy P. For ease of notation we introduce $\tilde{C}_0(\sigma, P) = 0$ to anchor the beginning of the planning horizon. The completion time of

any job j can be broken down into three components, namely the completion time of the preceding job, the deterministic processing time of the job itself and the delay due to MAs immediately before and during processing job j. For each $j \in J$ we derive $\tilde{C}_j(\sigma, P)$ as

$$\tilde{C}_j(\sigma, P) = \tilde{C}_{\sigma(\sigma^{-1}(j)-1)}(\sigma, P) + \underbrace{p_j + \tilde{M}_j^{\mathrm{PMA}}(\sigma, P) + \tilde{M}_j^{\mathrm{EMA}}(\sigma, P)}_{=\tilde{T}_j(\sigma,P)}. \tag{1}$$

Here, $\tilde{T}_j(\sigma, P)$ is a random variable capturing the time span between completion of the previous job and completion of job j. From now on we refer to $\tilde{T}_j(\sigma, P)$ as the *effective processing time* of job j in solution (σ, P), although it includes both the deterministic processing time and delays resulting from MAs.

A solution (σ, P) is evaluated by the expected sum of weighted completion times as

$$\begin{aligned} Z(\sigma, P) &= E\left[\sum_{j=1}^{n} w_j \tilde{C}_j(\sigma, P)\right] = \sum_{j=1}^{n} E[w_j \tilde{C}_j(\sigma, P)] = \sum_{j=1}^{n} w_j E[\tilde{C}_j(\sigma, P)] \\ &= \sum_{i=1}^{n} \left(\sum_{k=i}^{n} w_{\sigma(k)}\right) E[\tilde{T}_{\sigma(i)}(\sigma, P)] \end{aligned} \tag{2}$$

where we obtain the expression in (2) by reordering the summands according to the order of jobs in σ. We refer to $Z(\sigma, P)$ as the *expected total weighted completion time* (ETWC) of solution (σ, P). The integrated *job sequencing and maintenance planning problem* (JSMPP) is to determine an optimum solution, that is a solution with minimum ETWC. The introduced notation is summarized in Table 1.

J	Set of jobs $\{1, \ldots, n\}$
$\mathcal{S}_J$	Set of permutations of J
σ	Job sequence $\sigma \in \mathcal{S}_J$
$\mathcal{P}(J)$	Set of PMA policies $P : J \times [0, l^{\max}] \to \{0, 1\}$
P	PMA policy $P \in \mathcal{P}(J)$
$P(j, l)$	PMA decision before job j at ML $l \in [0, l^{\max}]$
$\tilde{C}_j(\sigma, P)$	Completion time of job j in sequence σ applying P
$\tilde{\Delta}_j$	Random variable of the deterioration of job j
d^{ema}	Duration of an EMA
$\tilde{L}_j(\sigma, P)$	ML at the completion of job j in sequence σ applying P
$l^{\max}$	Maximum ML
l_0	Initial ML
$\tilde{M}_j^{\mathrm{EMA}}(\sigma, P)$	Duration of EMAs during job j in sequence σ applying P
$\tilde{M}_j^{\mathrm{PMA}}(\sigma, P)$	Duration of a PMA scheduled before job j in sequence σ applying P
n	Number of jobs
p_j	Deterministic processing time of job j
$\tilde{T}_j(\sigma, P)$	Effective processing time of job j in sequence σ applying P
w_j	Weight of job j
$Z(\sigma, P)$	ETWC for solution (σ, P)

Table 1. Notation for JSMPP.

With regard to the computational complexity we fall back on the results provided in Arter et al. (2026a), Bock et al. (2012), and Briskorn et al. (2024). First, deciding the PMA policy itself is strongly NP-hard for given σ, see Arter et al. (2026a) and Briskorn et al. (2024). Note that JSMPP generalizes this setting since by having deterministic processing times being overwhelmingly large (still significantly different) we can essentially fix any sequence. Second, the deterministic version of the problem where each job causes a deterministic deterioration is proven to be NP-hard in Bock et al. (2012).

4. Solution approaches for JSMPP

In this section we present several solution approaches for solving JSMPP. All of them rely on separate mechanisms for determining job sequence σ and PMA policy P. The decisions are taken hierarchically: PMA policies are determined for given job sequences (or subsequences). The approaches for this subproblem are adopted from Arter et al. (2026a) and Briskorn et al. (2024) and summarized in Section 4.1. Section 4.2 to Section 4.4, then, detail different approaches to determine job sequences. In Section 4.2 and Section 4.3 we develop a BaB approach and a tabu search heuristic, respectively, as more involved solution approaches. Finally, a simple priority rule for determining the job sequence is introduced in Section 4.4. An overview of the different solution approaches is given in Section 4.5.

4.1. PMA policies for given job sequences

Using dynamic programming in order to determine a PMA policy for a given job sequence is a natural choice. For the case where jobs' deteriorations are discretely distributed Briskorn et al. (2024) developed an exact and computationally efficient *dynamic program* (DP). States reflect having processed a certain number of jobs and having reached a certain ML while transitions reflect the two options we have, namely to conduct a PMA or not. We refer to this DP as the *policy-DP* (P-DP) in the following.

In Arter et al. (2026a) this DP has been adapted to an approximate DP in order to solve the case where jobs' deteriorations are continuously distributed. While this approach provides reliably good solutions among the competitors, it suffered from significant run times for larger instances. The authors show that a discretization approach enabling the P-DP to be employed achieves lower run times while preserving the solution quality. Since the discretization can in principle be refined arbitrarily, the optimum can be approximated to any desired accuracy, so that for a sufficiently fine discretization the P-DP policy can

be treated as optimal in what follows. As an additional alternative, a *dynamic threshold* (dTH) policy has been developed, where a PMA is conducted whenever the ML is such that the probability that the next job triggers an EMA, if no PMA is performed, is larger than $\eta \in [0, 1]$, where η, as a steering parameter, has to be calibrated. More formally, the dTH policy is defined as

$$P_{\mathrm{dTH}(\eta)} : J \times [0, l^{\max}] \to \{0, 1\}, (j, l) \mapsto \begin{cases} 1, & 1 - F_j(l) > \eta, \\ 0, & \text{otherwise.} \end{cases}$$

for the distribution function F_j of the deterioration $\tilde{\Delta}_j$ caused by job $j \in J$. We refer to the parameterization provided in Arter et al. (2026a) with η depending on the ratio between d^{ema} and $l^{\max}$. While run times of determining the dTH policy are negligible and optimality gaps are very small (1.15% on average as reported in Arter et al. (2026a)), applying the dTH policy to a fixed job sequence is not necessarily optimum. Hence, even a potentially optimum framework will become heuristic if dTH policies are consulted for sequence evaluation. On the upside, run times will decrease significantly.

4.2. BaB approach for solving JSMPP

This section introduces a BaB approach for solving JSMPP. Section 4.2.1 describes the search tree structure and the node selection strategies. Section 4.2.2 develops a LB.
Note that for all BaB approaches we derive an initial upper bound by evaluating the WSPT order of jobs and refrain from computing upper bounds for other nodes to reduce run times.

4.2.1. Search tree and node selection strategies

We follow a common approach in scheduling literature and build job sequences sequentially. Thereby, jobs are assigned in reverse order, beginning with the job in position n. We do so to exploit the structure of the P-DP which is evaluated using backwards evaluation (see Briskorn et al. (2024) for details). For an instance I of JSMPP the search tree consists of $|J| = n$ levels including an initial level 0 holding a single root node. A node on level k, $k = 1, \ldots, n-1$ corresponds to a partial sequence in which jobs have been scheduled in the last k positions $n - k + 1, \ldots, n$. Note that a full sequence is implied as soon as level $n - 1$ is reached.

Formally, let $\mathcal{S}_{J,k} := \{\sigma = (\sigma(1), \ldots, \sigma(k)) \,|\, \sigma(i) \in J, \sigma(i) \neq \sigma(i'), i \neq i', i = 1, \ldots, k\}$ be the set of subsequences of length $k < n$ for the underlying job set $J = \{1, \ldots, n\}$. Then, a node N on level k is fully defined by a subsequence $\hat{\sigma}_N \in \mathcal{S}_{J,k}$. Let J^N denote the set

of sequenced jobs in node N. The set of unsequenced jobs, then, is $\bar{J}^N := J \setminus J^N$. The root node at level 0 corresponds to an empty subsequence with no jobs sequenced yet. Branching at node N on level k is performed by selecting each job $j \in \bar{J}^N$ and assigning it to position $n-k$ which yields a child node N' on level $k+1$ with subsequence $\hat{\sigma}_{N'} \in \mathcal{S}_{J,k+1}$ holding $\hat{\sigma}_{N'}(1) = j$ and $\hat{\sigma}_{N'}(i) = \hat{\sigma}_N(i-1)$ for $i = 2, \ldots, k+1$. The number of N's child nodes is $n-k$ which corresponds to the number of candidates for extending subsequence $\hat{\sigma}_N$.

As in most BaB approaches, nodes are evaluated using LBs, which bound the minimum ETWC for all possible full sequences within the current branch from below. When applying a BaB approach, the next node to be branched may be selected based on those bounds. Obviously, we then favor nodes with low LBs and consider three different node selection strategies.

The first one is a *Depth-First Search* (DFS), where the node deepest in the current search tree path with the lowest LB is selected to be branched next. We consider the DFS for the problem at hand for two reasons. In preliminary experiments we experienced rather excessive memory requirements even for small instances and, hence, keeping memory requirements as low as possible, as supported by DFS, becomes relevant. Furthermore, we expect that having jobs in close to WSPT order provides good solutions and, hence, we have a good idea in which region of the search tree to delve to find good solutions fast.
The second node selection strategy to be considered is *Cyclic Best-First Search* (CBFS) as introduced in Morrison et al. (2016). In one specific variant of CBFS, nodes are partitioned into so called contours induced by their level in the search tree. The term contour, thereby, describes a partitioning of search tree levels into $g_{\text{CBFS}} \geq 1$ groups and is commonly used in the stream of literature using CBFS. For example, for n contours each level of the search tree forms its own contour, whereas for $n/2$ contours two consecutive levels are aggregated into one contour. The algorithm, then, iterates over the contours with increasing levels and selects in each iteration the node in the current contour with the lowest LB. When a node of the last contour has been branched the algorithm restarts with the first contour. This strategy hybridizes DFS and classical Best-First Search and gives the opportunity to switch between promising regions of the search space while keeping the memory usage manageably low. Since no node is permanently discarded unless the LB exceeds the lowest yet found upper bound CBFS preserves optimality.

The third node selection strategy to be considered is a heuristic *Beam Search* (BS). Here, the algorithm walks through each level of the search tree and retains only a fixed number $w_{\text{BS}} \geq 1$ of nodes, the beam width, on the current level which are again those with the lowest LBs. All remaining nodes on the level are discarded. By doing so the number of nodes to be branched is reduced which in turn reduces run times. However, it is not guaranteed that the subtree containing the optimum sequence is not discarded so that using BS constitutes a heuristic solution approach only. Note that for sufficiently high

numbers of nodes on each level retained BS reduces to a Breadth-First Search.

4.2.2. Lower bound

The efficiency of the BaB strongly depends on the ability to prune individual branches of the search tree. Effective pruning reduces the size of the search space, thereby enabling the optimal sequence to be identified more efficiently. For node N on level k the subsequence of jobs in J^N assigned to positions $n-k+1,\dots,n$ is referred to as subsequence $\hat{\sigma}_N$. The set of full sequences ending with this subsequence is denoted as $\mathcal{S}_{J|\hat{\sigma}_N} := \{\sigma \in \mathcal{S}_J \,|\, \sigma(n-k+i) = \hat{\sigma}_N(i), i = 1,\dots,k\}$. In the following, we first present the overall bounding scheme before giving details about how to design the components. The used notation is summarized in Table 2.

I	Instance of JSMPP
N	Node
$\mathcal{S}_{J,k}$	Set of subsequences of length $k < n$ of job set J
$\mathcal{S}_{J\|\hat{\sigma}_N}$	Set of full sequences ending with subsequence $\hat{\sigma}_N$
$\hat{\sigma}_N \in \mathcal{S}_{J^N}$	Fixed subsequence in node N
$\sigma'_N \in \mathcal{S}_{\bar{J}^N}$	Subsequence complementing $\hat{\sigma}_N$ to a full sequence in node N
J^N	Set of jobs in subsequence $\hat{\sigma}_N$
$\bar{J}^N$	Set of jobs not in subsequence $\hat{\sigma}_N$

Table 2. Summary of the notation used for the LB.

When considering node N, the objective function can be decomposed into the contribution of the jobs scheduled in J^N and the contribution of the not yet scheduled jobs in $\bar{J}^N$. This underlying decomposition is from now on referred to as the *instance split*. Given an instance I of JSMPP and node N on level k of the BaB tree we consider the ETWC for a full sequence $\sigma \in \mathcal{S}_{J|\hat{\sigma}_N}$ and PMA policy $P \in \mathcal{P}(J)$ and reformulate as

$$\begin{aligned} Z_I(\sigma,P) &= \sum_{i=1}^{n}\left(\sum_{l=i}^{n} w_{\sigma(l)}\right) E[\tilde{T}_{\sigma(i)}(\sigma,P)] \\ &= \sum_{i=1}^{n-k}\left(\sum_{l=i}^{n} w_{\sigma(l)}\right) E[\tilde{T}_{\sigma(i)}(\sigma,P)] + \sum_{i=n-k+1}^{n}\left(\sum_{l=i}^{n} w_{\sigma(l)}\right) E[\tilde{T}_{\sigma(i)}(\sigma,P)] \\ &= \underbrace{\sum_{i=1}^{n-k}\left(\sum_{l=i}^{n} w_{\sigma(l)}\right) E[\tilde{T}_{\sigma(i)}(\sigma,P)]}_{\substack{=:\mathcal{Z}_1(\sigma,P) \\ \text{contribution of jobs not in } \hat{\sigma}_N}} + \underbrace{\sum_{i=1}^{k}\left(\sum_{l=i}^{k} w_{\hat{\sigma}_N(l)}\right) E[\tilde{T}_{\hat{\sigma}_N(i)}(\sigma,P)]}_{\substack{=:\mathcal{Z}_2(\sigma,P) \\ \text{contribution of jobs in } \hat{\sigma}_N}}. \end{aligned} \tag{3}$$

Let $\sigma'_N \in \mathcal{S}_{\bar{J}^N}$ denote the subsequence of σ restricted to the first $n-k$ positions, i.e., $\sigma'_N(i) = \sigma(i)$ for $i = 1,\dots,n-k$. The full sequence σ can, then, be seen as a concatenation of disjoint subsequences σ'_N and $\hat{\sigma}_N$. Figure 1 provides a schematic visualization of the instance split using the introduced notation.

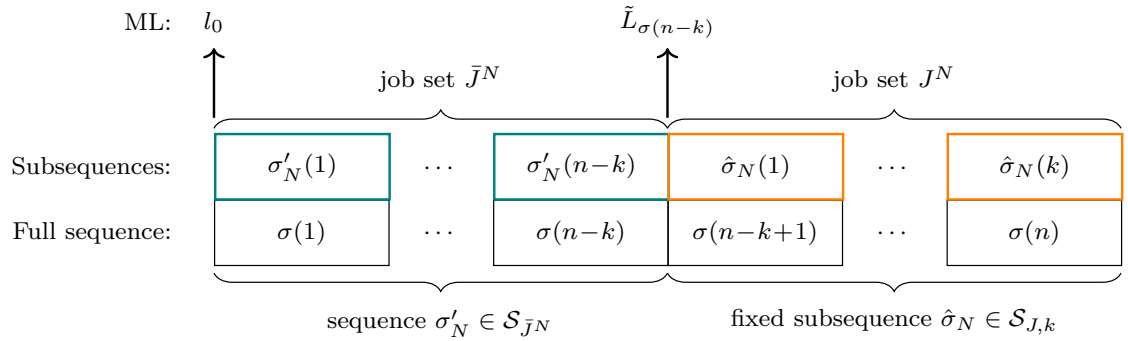


Figure 1. Visualization and notation of the instance split.

Note that $\mathcal{Z}_1(\sigma, P)$ and $\mathcal{Z}_2(\sigma, P)$ are not independent since the sequence of jobs in $\bar{J}^N$ and the PMA policy influence, among others, the ML $\tilde{L}_{\sigma(n-k)}$ upon completion of the job positioned last. This random variable is decisive for the further ML development, PMA durations and EMA occurrences during the processing of the jobs scheduled in $\hat{\sigma}_N$ and, thus, impacts the effective processing times of jobs in J^N. However, a decoupling of $\mathcal{Z}_1(\sigma, P)$ and $\mathcal{Z}_2(\sigma, P)$ can be achieved by assuming that the ML at completion of the last job in $\bar{J}^N$ has a certain value l^* irrespective of the subsequence σ'_N and PMA policy P. Note that the value of l^* does not have any impact on $\mathcal{Z}_1(\sigma, P)$. On the other hand, $\mathcal{Z}_2(\sigma, P)$ decreases with increasing value of l^* since PMAs potentially become shorter and EMAs become less likely. Hence, in order to derive an LB we choose $l^* = l^{\max} - \mu$ where μ is an LB on the deterioration caused by any job in $\bar{J}^N$. We denote the contribution of jobs in J^N under the assumption of the initial ML l^* as $\mathcal{Z}_2(\sigma, P \,|\, l^*)$. Once decoupled, minimizing the expression in (3) can be achieved by deciding (i) sequence σ'_N of jobs in $\bar{J}^N$ and the portion of the PMA policy applied to those jobs and (ii) the portion of the PMA policy applied to the fixed jobs in J^N independently. To sum up, fixing the ML to l^* removes the dependence of the effective processing times of the fixed jobs in $\hat{\sigma}_N$ on subsequence σ'_N and prior PMA decisions, so that $\mathcal{Z}_2$ depends only on the applied PMA policy and, consequently, we denote $\mathcal{Z}_2(P \,|\, l^*)$ instead of $\mathcal{Z}_2(\sigma, P \,|\, l^*)$. Let $\mathcal{Z}_1^*$ and $\mathcal{Z}_2^*(l^*)$ be the minimum value of $\mathcal{Z}_1(\sigma, P)$ and $\mathcal{Z}_2(P \,|\, l^*)$, respectively. That is

$$\mathcal{Z}_1^* := \min_{\substack{\sigma \in \mathcal{S}_{J|\hat{\sigma}_N} \\ P \in \mathcal{P}(J)}} \mathcal{Z}_1(\sigma, P) \text{ and } \mathcal{Z}_2^*(l^*) := \min_{P \in \mathcal{P}(J)} \mathcal{Z}_2(P \,|\, l^*).$$

Lemma 3. *$\mathcal{Z}_1^* + \mathcal{Z}_2^*(l^*)$ is an LB on the minimum ETWC achievable for a solution (σ, P) in node N with $\sigma \in \mathcal{S}_{J|\hat{\sigma}_N}$ and $P \in P(J)$.*

We omit a formal proof since the lemma follows directly from the fact that $\mathcal{Z}_2(P \,|\, l^*)$ is decreasing in l^*. Intuitively speaking, $\mathcal{Z}_1^* + \mathcal{Z}_2^*(l^*)$ is an LB since we set the initial ML for jobs in J^N to the best possible value and do so without accounting for any delay caused by a PMA between the last job in $\bar{J}^N$ and $\hat{\sigma}_N(1)$ to reach this ML.

Since $\mathcal{Z}_1^*$ and $\mathcal{Z}_2^*(l^*)$ can be evaluated independently we can fall back on the methods outlined in Section 4.1 to determine $\mathcal{Z}_2^*(l^*)$ using the optimal P-DP policy. We can derive an estimate for $\mathcal{Z}_2^*(l^*)$ using dTH (and we do so in our computational study) but this does not guarantee an LB. Hence, we focus on $\mathcal{Z}_1^*$ in the following and start by reformulating $\mathcal{Z}_1(\sigma, P)$. With

$$W_N := \sum_{l=n-k+1}^{n} w_{\sigma(l)} = \sum_{l=1}^{k} w_{\hat{\sigma}_N(l)} = \sum_{j \in J^N} w_j$$

reflecting the total weight of fixed jobs we reformulate $\mathcal{Z}_1(\sigma, P)$ to

$$\begin{aligned}
\mathcal{Z}_1(\sigma, P) &= \sum_{i=1}^{n-k} \left(\sum_{l=i}^{n} w_{\sigma(l)} \right) E[\tilde{T}_{\sigma(i)}(\sigma, P)] \\
&= \sum_{i=1}^{n-k} \left(\sum_{l=i}^{n-k} w_{\sigma(l)} + W_N \right) E[\tilde{T}_{\sigma(i)}(\sigma, P)] \\
&= \sum_{i=1}^{n-k} \left(\sum_{l=i}^{n-k} w_{\sigma(l)} + W_N \right) E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma, P) + p_{\sigma(i)} + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma, P)] \\
&= \sum_{i=1}^{n-k} \left(\sum_{l=i}^{n-k} w_{\sigma(l)} + W_N \right) \cdot p_{\sigma(i)} \\
&\quad + \sum_{i=1}^{n-k} \left(\sum_{l=i}^{n-k} w_{\sigma(l)} + W_N \right) E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma, P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma, P)] \\
&= \underbrace{\sum_{i=1}^{n-k} \left(\sum_{l=i}^{n-k} w_{\sigma(l)} \right) p_{\sigma(i)}}_{=:\mathcal{Z}_3(\sigma)} + \underbrace{W_N \cdot \sum_{j \in \bar{J}^N} p_j}_{=:\mathcal{C}_1(N)} \\
&\quad \underbrace{+ \sum_{i=1}^{n-k} \left(\sum_{l=i}^{n-k} w_{\sigma(l)} + W_N \right) E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma, P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma, P)]}_{=:\mathcal{Z}_4(\sigma, P)}
\end{aligned}$$

Note that $\mathcal{C}_1(N)$ is a constant and does not depend on sequence $\sigma \in \mathcal{S}_{J|\hat{\sigma}_N}$ or PMA policy $P \in \mathcal{P}(J)$. Note, furthermore, that $\mathcal{Z}_3(\sigma)$ depends on σ but not on P.

Let $\mathcal{Z}_3^*$ and $\mathcal{Z}_4^*$ be the minimum value of $\mathcal{Z}_3(\sigma)$ and $\mathcal{Z}_4(\sigma, P)$, respectively, that is

$$\mathcal{Z}_3^* := \min_{\sigma \in \mathcal{S}_{J|\hat{\sigma}_N}} \mathcal{Z}_3(\sigma) \text{ and } \mathcal{Z}_4^* := \min_{\substack{\sigma \in \mathcal{S}_{J|\hat{\sigma}_N} \\ P \in \mathcal{P}(J)}} \mathcal{Z}_4(\sigma, P).$$

Therefore, $\mathcal{Z}_3^*$ and $\mathcal{Z}_4^*$ can be achieved by choosing (i) the sequence of jobs in $\bar{J}^N$ and (ii) the sequence of jobs in $\bar{J}^N$ together with a PMA policy. Building upon Lemma 3, we conclude Lemma 4.

Lemma 4. *$\mathcal{Z}_3^* + \mathcal{C}_1(N) + \mathcal{Z}_4^* + \mathcal{Z}_2^*(l^*)$ is an LB on the minimum ETWC achievable for a solution (σ, P) in node N with $\sigma \in \mathcal{S}_{J|\hat{\sigma}_N}$ and $P \in \mathcal{P}(J)$.*

It is well known that $\mathcal{Z}_3^*$ is achieved by having jobs in WSPT order. It remains to determine an LB for $\mathcal{Z}_4^*$ comprising the impact of delays by MAs. The idea is to bound the expected delay of the l-th job, $1 \leq l \leq n-k$, due to MAs regardless of the full job sequence σ. For given job sequence σ and PMA policy P, the expected delay is bounded from below by the amount by which the total expected deterioration up to this point exceeds the initial ML. Following this idea, we obtain

$$\begin{aligned}
\sum_{i=1}^{l} E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma,P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma,P)] &= E[\sum_{i=1}^{l} \tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma,P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma,P)] \\
&\geq E\left[\max\left\{\sum_{i=1}^{l} \tilde{\Delta}_{\sigma(i)} - l_0, 0\right\}\right] \\
&\geq \max\left\{E\left[\sum_{i=1}^{l} \tilde{\Delta}_{\sigma(i)} - l_0\right], 0\right\} \\
&= \max\left\{\sum_{i=1}^{l} E[\tilde{\Delta}_{\sigma(i)}] - l_0, 0\right\}
\end{aligned}$$

for each PMA policy P, sequence $\sigma \in \mathcal{S}_{J|\hat{\sigma}_N}$ and position l. Thereby, the first inequality simply accounts for MAs being required to keep the ML non-negative. The second inequality above holds due to Jensen's inequality applicable due to the max function being convex.

We consider two sequences σ^w and σ^δ of all jobs in $\bar{J}^N$ in non-increasing weights and non-decreasing expected deterioration. Then

$$\max\left\{\sum_{i=1}^{l} E[\tilde{\Delta}_{\sigma(i)}] - l_0, 0\right\} \geq \max\left\{\sum_{i=1}^{l} E[\tilde{\Delta}_{\sigma^\delta(i)}] - l_0, 0\right\}$$

and

$$\sum_{i=l}^{n-k} w_{\sigma(i)} \geq \sum_{i=l}^{n-k} w_{\sigma^w(i)}$$

for each sequence σ and each position $1 \leq l \leq n-k$. Thus, with

$$\begin{aligned}
\mathcal{Z}_4(\sigma,P) &= \sum_{i=1}^{n-k}\left(\sum_{l=i}^{n-k} w_{\sigma(l)} + W_N\right) E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma,P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma,P)] \\
&\geq \sum_{i=1}^{n-k}\left(\sum_{l=i}^{n-k} w_{\sigma^w(l)}\right) E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma,P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma,P)] \\
&\quad + W_N \cdot \sum_{i=1}^{n-k} E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma,P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma,P)] \\
&= \sum_{l=1}^{n-k} w_{\sigma^w(l)} \sum_{i=1}^{l}\left(E[\tilde{M}_{\sigma(i)}^{\mathrm{PMA}}(\sigma,P) + \tilde{M}_{\sigma(i)}^{\mathrm{EMA}}(\sigma,P)\right)
\end{aligned}$$

$$
\begin{aligned}
&+ W_N \cdot \sum_{i=1}^{n-k} E[\tilde{M}^{\mathrm{PMA}}_{\sigma(i)}(\sigma, P) + \tilde{M}^{\mathrm{EMA}}_{\sigma(i)}(\sigma, P)] \\
\geq\ & \sum_{l=1}^{n-k} w_{\sigma^w(l)} \cdot \max\left\{\sum_{i=1}^{l} E[\tilde{\Delta}_{\sigma(i)}] - l_0, 0\right\} \\
&+ W_N \cdot \max\left\{\sum_{i=1}^{n-k} E[\tilde{\Delta}_{\sigma(i)}] - l_0, 0\right\} \\
\geq\ & \sum_{l=1}^{n-k} w_{\sigma^w(l)} \cdot \max\left\{\sum_{i=1}^{l} E[\tilde{\Delta}_{\sigma^\delta(i)}] - l_0, 0\right\} \\
&+ W_N \cdot \max\left\{\sum_{i=1}^{n-k} E[\tilde{\Delta}_{\sigma(i)}] - l_0, 0\right\} =: \mathcal{C}_2(N)
\end{aligned}
$$

we receive a constant bound $\mathcal{C}_2(N)$ on $\mathcal{Z}_4^*$ independent of σ and P. We conclude Theorem 3.

Theorem 3. *$\mathcal{Z}_3^* + \mathcal{C}_1(N) + \mathcal{C}_2(N) + \mathcal{Z}_2^*(l^*)$ is an LB on the minimum ETWC achievable for a solution (σ, P) in node N with $\sigma \in \mathcal{S}_{J|\hat{\sigma}_N}$ and $P \in \mathcal{P}(J)$.*

Proof. The theorem follows directly from Lemma 4 and the fact that $\mathcal{C}_2(N) \leq \mathcal{Z}_4^*$. □

4.3. Tabu search

As an alternative solution approach we propose a *tabu search* (TS) using insertion moves to iterate over job sequences. As the initial solution we use the job sequence ordered according to the WSPT rule, which is optimal for the underlying weighted completion time problem without deterioration and, therefore, provides a strong initial solution. Different job sequences are considered by the TS. Each job sequence under consideration is then evaluated by means of applying either a PMA policy derived from the P-DP or the dTH policy as proposed in Section 4.1. When the P-DP is used, it is ensured that all sequences are evaluated accurately, that is the best policy is derived for the sequence at hand. This is not necessarily the dTH policy which as a heuristic evaluation carries the risk that sequences are evaluated inaccurately and, as a consequence, potentially discarded although they would be promising under an optimal PMA policy.

In each iteration, the algorithm explores the full insertion neighborhood of the current sequence, that is, all sequences that can be obtained by removing a job from its current position and reinserting it at another position (keeping the relative order of all other jobs). An insertion move is considered admissible if the moved job is not tabu according to a criterion specified in the following. Within one iteration of the TS, the sequence with the lowest derived ETWC is selected even if it does not improve upon the current sequence which enables the search to escape local optima. The job that has been moved is then

declared tabu for a fixed number of iterations τ, the tabu tenure, during which it may not be moved again. This job-based tabu attribute prevents the search from repeatedly moving the same job and thereby cycling through a small set of sequences. Following the standard aspiration criterion, a tabu move is nevertheless admitted if it leads to a sequence better than the best solution found so far. The search terminates once a maximum number $it^{\max}$ of iterations is reached, or once a given number it^{ni} of consecutive iterations have passed without improving the best-known solution. The fully configured solution approach is denoted as $\text{TS}(\tau, it^{\max}, it^{\text{ni}})$.

4.4. Heuristic priority rules

Due to the close relationship to the classic single machine scheduling problem without deterioration, it seems natural to adapt the WSPT rule to JSMPP. Incorporating deterioration into this ratio is, however, not straightforward. A higher deterioration value does not necessarily induce an MA during the processing of the corresponding job since the occurrence of MAs depends on the job sequence and the development of the ML over time rather than on a single job's deterioration alone. Nevertheless, it is reasonable to hypothesize that a job with a high expected deterioration is more likely to require a PMA prior to its processing to reduce the risk of EMAs, implying a longer effective processing time of the job itself. We therefore propose to treat the expected deterioration incurred during the processing of job j as a proxy for this delay, where the delay magnitude is determined by the PMA duration required to restore the ML. We introduce the weighted processing and deterioration ratio as a criterion for defining the job order. For job j the considered ratio is computed as

$$r_j(\alpha) := \frac{\alpha \cdot p_j + (1-\alpha) \cdot E[\tilde{\Delta}_j]}{w_j},$$

where $\alpha \in [0,1]$ controls the weighting of both components. For $\alpha = 1$ deterioration effects are not considered at all while for $\alpha = 0$ the decision is based on the expected deteriorations only. For $\alpha = 0.5$ deterministic processing times and expected deteriorations are weighted equally. The *weighted shortest processing and deterioration rule*, denoted as $\text{WSPDR}(\alpha)$, induces a sequence of jobs in non-decreasing order of $r_j(\alpha)$ which coincides with the WSPT order for $\alpha = 1$. This sequence, then, can be complemented by the P-DP policy or the dTH policy as proposed in Section 4.1. For simplicity, we refer to this solution approach as $\text{WSPDR}(\alpha)$, as well. Note that the advantage of using dTH policies in order to reduce computational efforts is limited since there is only one PMA policy to be determined which only worsens the evaluation of the final sequence. For this reason, we will not consider $\text{WSPDR}(\alpha)$ using dTH policies any further.

4.5. Summary of solution approaches for JSMPP

At this point we want to briefly summarize the solution approaches introduced above and characterize their solution quality. All BaB approaches share the same search tree as defined in Section 4.2 and use the LB according to Theorem 3. The solution approaches differ in the node selection strategies and how (sub)sequences are evaluated. For the node selection strategy we distinguish between DFS, CBFS and BS. If sequences are evaluated using optimum PMA policies derived with the P-DP the BaB approaches using DFS and CBFS are exact solution methods. Using BS yields a heuristic approach due to its node selection strategy which does not guarantee that all parts of the search tree are considered. If the dTH policy is applied to compute LBs, all proposed BaB approaches become heuristic. Note that the final job sequence still is evaluated using the optimal PMA policy by solving a P-DP once guaranteeing that the final solution is evaluated consistently and not penalized by the heuristic policy approximation. Besides the BaB approaches, TS($\tau, it^{\max}, it^{\text{ni}}$) and WSPDR($\alpha$) are heuristic solution approaches by construction.

We refer to a solution approach by specifying the method to determine the job sequence when P-DPs are employed. Solution approaches employing the dTH policy are referred to by specifying the method to determine the job sequence marked with a tilde. Table 3 provides an overview of the approaches and their classification, with exact approaches printed in bold.

PMA policy	Job sequence				
	BaB approaches			TS	Priority rule
P-DP	**BaB(DFS)**	**BaB(CBFS,g_{CBFS})**	BaB(BS,w_{BS})	TS($\tau, it^{\max}, it^{\text{ni}}$)	WSPDR(α)
dTH	$\widetilde{\text{BaB}}$(DFS)	$\widetilde{\text{BaB}}$(CBFS,g_{CBFS})	$\widetilde{\text{BaB}}$(BS,w_{BS})	$\widetilde{\text{TS}}$($\tau, it^{\max}, it^{\text{ni}}$)	-

Table 3. Overview of all considered solution approaches for JSMPP (exact approaches in bold).

5. Computational study

This section is dedicated to evaluate the performance of all solution approaches for JSMPP summarized in Table 3. We begin with defining suitable instance sets in Section 5.1. Section 5.2 provides the parameter settings for the solution approaches introduced in Section 4. The numerical results are presented in Section 5.3.
All computational tests have been executed on a computer with 64 GB of RAM, an i7-12700K CPU @5 GHz running Ubuntu 22.04.

Machine environment	
$n \in \{4,5,6,7,8,9,10\} =: \mathcal{N}$	Number of jobs in $\mathcal{I}$
$n \in \{15,20,30,40,50\} =: \mathcal{N}^L$	Number of jobs in $\mathcal{I}^L$
$l^{\max} = l_0 = 25$	Maximum and initial ML
$d^{\text{ema}} \in \{30,50,100\}$	EMA duration
Generating parameters	
$\beta \in \{1,2,4\}$	Deteriorations and processing times intervals
$\gamma \in \{0,3,7\}$	Weight intervals
Deterioration	
$m \in \{1,2\}$	Deterioration precision
$n_{\text{det}} \in \{1,3,5\}$	Number of deteriorations per job
$[(0.4 - 0.1 \cdot \beta) \cdot l^{\max}, (0.4 + 0.1 \cdot \beta) \cdot l^{\max}]$	Deterioration interval
Deterministic processing times	
$p \ll \tilde{\Delta}: \ [0, (0.4 - 0.1 \cdot \beta) \cdot l^{\max}]$	*Lower* deterministic processing times
$p \sim \tilde{\Delta}: \ [(0.4 - 0.1 \cdot \beta) \cdot l^{\max}, (0.4 + 0.1 \cdot \beta) \cdot l^{\max}]$	*Equal* deterministic processing times
$p \gg \tilde{\Delta}: \ [(0.4 + 0.1 \cdot \beta) \cdot l^{\max}, l^{\max}]$	*Higher* deterministic processing times
Weights	
$[1 - \gamma \cdot 0.1, 1 + \gamma \cdot 0.1]$	Weight interval

Table 4. Parameter configuration for instance generation of instance sets $\mathcal{I}$ and $\mathcal{I}^L$.

5.1. Instance generation

Before defining the instance generation in detail we want to give some preliminary remarks. It has to be noted that for optimally solving JSMPP using the exact BaB approaches, continuously distributed deteriorations have to be discretized in order to apply the P-DP. Therefore, we restrict our considerations to discrete deterioration in advance. The main complexity drivers are, then, the number of jobs n, the number of deterioration realizations per job n_{det} and the precision m of the deterioration realizations. By holding m constant for all jobs within one instance, the set of relevant MLs equals the subset of $[0, l^{\max}]$ holding all multiples of 10^{-m}. Since all relevant MLs must be evaluated when deriving PMA policies via evaluating the P-DP, a substantial increase in computational effort is to be expected as m grows.

We are now equipped to present the design of the instances considered in the computational study. We consider two instance sets $\mathcal{I}$ and $\mathcal{I}^L$ using the parameter configuration as provided in Table 4 differing exclusively in the number of jobs $n \in \mathcal{N}$ and $n \in \mathcal{N}^L$ respectively. Instance set $\mathcal{I}$ holds instances of manageable size for which all proposed approaches are expected to be computationally tractable. Instance set $\mathcal{I}^L$ consists of larger instances investigated separately in Section 5.3.3. All random instances are generated using a fixed random seed.

The maximum ML $l^{\max}$ is held constant across all instances to avoid scale effects. Further, we assume a constant initial ML l_0 equaling the maximum ML $l^{\max}$. This assumption is reasonable for two reasons. First, it has already been shown in Briskorn et al. (2024) for a closely related setting that varying the initial ML does not provide decisive infor-

mation on the solution structure. Second, it reflects a plausible real-world setting, since $l_0 = l^{\max}$ corresponds to a machine handed over in perfect condition at the beginning of the planning horizon. The duration d^{ema} of an EMA varies across the instances under consideration in order to investigate different machine environments. We consider instances with jobs that are homogeneous with respect to their deterioration structure. Accordingly, the number of discrete possible deteriorations equals a fixed number n_{det} for every job to be scheduled. The underlying distribution is generated randomly using predefined deterioration intervals. These are chosen relative to $l^{\max}$. Values substantially above $0.5 \cdot l^{\max}$ would largely eliminate PMA planning, since processing two consecutive jobs without a PMA would almost always trigger an EMA. We consider deterioration intervals $[(0.4 - 0.1 \cdot \beta) \cdot l^{\max}, (0.4 + 0.1 \cdot \beta) \cdot l^{\max}]$ for $\beta \in \{1, 2, 4\}$. Deteriorations are, then, randomly drawn from the just defined interval and corresponding probabilities are randomly drawn from $[0, 1]$ such that we have a valid discrete probability distribution. We further distinguish between three classes of instances by generating deterministic processing times from intervals below, equal to, or above the deterioration interval. The resulting classes enable us to analyze how the performance of the proposed approaches changes when delays induced by MAs dominate, contribute similarly to, or are dominated by deterministic processing times. Weights are also randomly generated and are drawn from an interval centered around 1 with interval width $0.2 \cdot \gamma$ for a generating parameter γ. By allowing an interval width of $\gamma = 0$, we also include the special case of equal weights. Overall, we investigate the two instance sets $\mathcal{I}$ and $\mathcal{I}^L$ consisting of 3 402 and 2 430 instances respectively. We further define the disjoint instance subsets $\mathcal{I}_l, \mathcal{I}_e, \mathcal{I}_h$ and $\mathcal{I}_l^L, \mathcal{I}_e^L, \mathcal{I}_h^L$, comprising the instances associated with *lower*, *equal* and *higher* processing times as listed in Table 4.

5.2. Parametrization of solution approaches for JSMPP

It remains to set the open parameters for all solution approaches presented in Section 4. For simplicity, all parameter configurations are introduced in Table 5 using the notation of the P-DP based solution approaches. The same parameter settings are also used for the corresponding dTH-based approaches.

We start with the different BaB approaches. The BaB(DFS) is fully defined without any parameter settings. For the parameter g_{CBFS} for BaB(CBFS,g_{CBFS}) we consider values in $\{2, 3, 4, 5\}$ suitable for the instance sizes considered in the computational study. The BaB(BS,w_{BS}) is defined by the number of nodes to be branched on each level of the search tree. We consider $w_{\text{BS}} \in \{10, 20, 100, 200, 500\}$ to analyze the trade-off between computational effort and solution quality.
We consider a TS configuration as proposed in Section 4.3 with a tabu tenure of $\tau = 4$.

BaB approaches	
$g_{\text{CBFS}} \in \{2, 3, 4, 5\}$	Number of contours in BaB(CBFS,g_{CBFS})
$w_{\text{BS}} \in \{10, 20, 100, 200, 500\}$	Beam width in BaB(BS,w_{BS})
TS$(\tau, it^{\max}, it^{\text{ni}})$	
$\tau = 4$	Tabu tenure
$it^{\max} = 1\,000$	Maximum number of iterations
$it^{\text{ni}} = 30$	Maximum number of consecutive iterations without improvement
WSPDR(α)	
$\alpha \in \{0, 1/3, 0.5, 2/3, 1\}$	Weighting parameter

Table 5. Parameter configuration for all solution approaches for JSMPP.

The approach terminates after $it^{\max} = 1\,000$ iterations or after $it^{\text{ni}} = 30$ consecutive iterations without improvement. It has been shown in preliminary tests that this configuration is suitable for all tests in the computational study. For readability, we refer to TS$(4, 1\,000, 30)$ simply as TS from now on, as this is the only configuration considered in the computational study.
We consider $\alpha \in \{0, 1/3, 0.5, 2/3, 1\}$ for WSPDR(α), covering among others the classical WSPT order ($\alpha = 1$), the exclusion of processing times in the ratio ($\alpha = 0$) and equal weighting of processing times and expected deterioration ($\alpha = 0.5$). The two remaining values for α represent a 2:1 weighting of the two components, in favor of the expected deterioration ($\alpha = 1/3$) and in favor of the processing times ($\alpha = 2/3$), respectively.

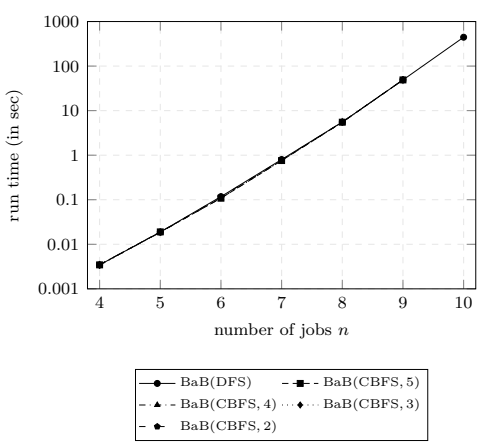
5.3. Numerical results

This section evaluates the solution approaches developed in Section 4 with respect to two dimensions, their run times and the achieved ETWC, starting with instance set $\mathcal{I}$. We first analyze the run times in Section 5.3.1. Section 5.3.2 then turns to the solution quality, reporting both the share of instances solved to optimality and the average optimality gaps. Finally, Section 5.3.3 extends the study to instance set $\mathcal{I}^L$ in order to compare the performances of the heuristic approaches with increasing number of jobs.

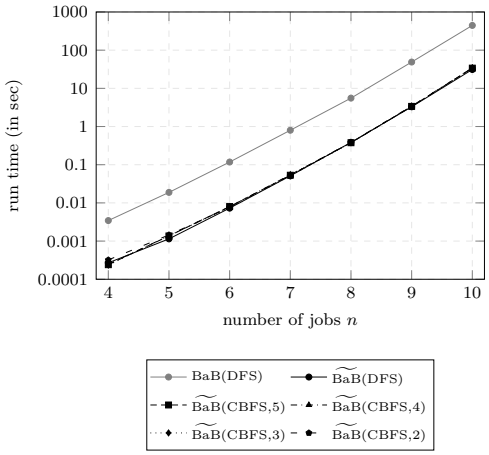
5.3.1. Run times

We first examine the run times of the proposed solution approaches. We expect the number of jobs n, the number of deteriorations per job n_{det} and the deterioration precision m to be the relevant complexity drivers.

We start our analysis with run times depending on the number of jobs n. Figure 2 presents the average run times of the two proposed BaB approaches BaB(DFS) and



(a) Sequence evaluation using P-DP policies.



(b) Sequence evaluation using dTH policies.

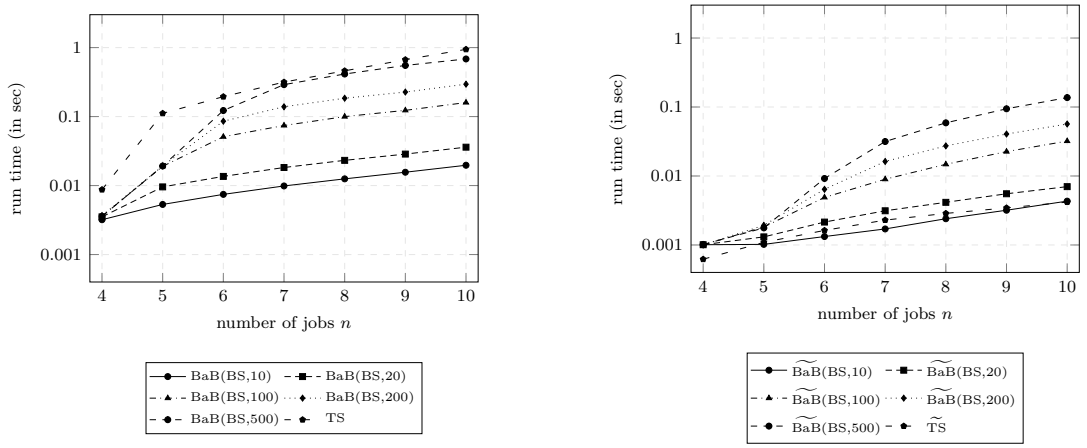
Figure 2. Average run times of DFS- and CBFS-based BaB approaches depending on the number of jobs n.

BaB(CBFS,g_{CBFS}) as a function of n, averaged over the 486 instances per value of n. Figure 2a depicts the results for the exact approaches. The curves almost completely overlap, with average run times ranging from about 0.003 sec for $n = 4$ to about 49 sec for $n = 9$. The approximately linear progression on the logarithmic scale indicates exponential growth in run times, so that solving JSMPP to optimality with the proposed exact BaB approaches demands considerable computational effort as n increases. For $n = 10$, only BaB(DFS) solves all considered instances, requiring about 444.6 sec on average, whereas BaB(CBFS,g_{CBFS}) exceeds the available memory on some of them. This is caused by the deliberate design choice of the approaches' implementation to store an evaluated P-DP stage in every node. Since sequences and P-DP stages are both constructed backwards, previously computed stages can be reused when extending a subsequence, substantially reducing computational effort at the expense of increased memory consumption. Consequently, memory becomes the limiting factor for BaB(CBFS,g_{CBFS}), which maintains many open nodes simultaneously, while BaB(DFS) stores only nodes along the current search path.

The almost identical run time of BaB(DFS) and BaB(CBFS,g_{CBFS}) are explained by the comparable numbers of evaluated nodes, which differ by less than 1% for $n \geq 5$. Hence, the computational bottleneck is the evaluation of nodes rather than the node selection strategy. The marginal run time differences among the exact approaches are not systematic. Neither BaB(DFS) nor any BaB(CBFS,g_{CBFS}) variant is consistently faster across all values of n and run time differences stem from minor overhead in managing the stored nodes from which the next one to be branched is selected rather than from the node selection strategy itself.

The run times for $\widetilde{\text{BaB}}$(DFS) and $\widetilde{\text{BaB}}$(CBFS,g_{CBFS}) are depicted in Figure 2b, together with those of BaB(DFS) for comparison. The curves of the dTH variants differing in the

node selection strategy almost coincide, which is in line with the previous conclusions. Thereby, the dTH variants take roughly 7% on average of the run times of their exact counterparts, ranging from below 0.0003 sec for $n = 4$ to about 3.4 sec for $n = 9$ (compared to about 49 sec for the exact approaches) due to the fast evaluation of sequences using dTH policies. Further, memory is no longer a limiting factor when using dTH policies. As a result, all $\widetilde{\text{BaB}}(\text{CBFS},g_{\text{CBFS}})$ variants now terminate for $n = 10$ in about 34 sec on average. The traversed search trees are slightly smaller under the dTH policies, with the number of evaluated nodes being reduced by about 1.2% on average. This reduction results from the suboptimal evaluation of sequences under dTH policies. Since the applied PMA policy is not optimal, the objective it assigns to a sequence is higher than the one obtained by applying the optimum P-DP policy. This raises the heuristic LBs computed in the nodes, so that the pruning condition may be met more often and fewer nodes are evaluated. The same suboptimality also affects the initial upper bound, which is likewise obtained via the application of dTH policies and is therefore looser than under an exact evaluation. A looser upper bound weakens pruning and would enlarge the search tree, so that the observed net reduction indicates that the effect of the raised node bounds slightly outweighs that of the looser upper bound. At the same time, since the dTH policy is not optimal, it can not be ruled out that a node's heuristic LB exceeds the true optimal objective value of the corresponding subproblem. In that case the bound is invalid and nodes may be pruned that are retained by the exact approaches. How these effects impact the solution quality is examined by means of the ETWC gaps in Section 5.3.2.



(a) Sequence evaluation using P-DP policies.

(b) Sequence evaluation using dTH policies.

Figure 3. Average run times of BaB(BS,w_{BS}) and TS approaches depending on the number of jobs n.

In Figure 3 the run times of the remaining heuristic solution approaches are reported. Thereby the results when using P-DP policies and the results when using dTH policies for sequence evaluation are depicted in Figure 3a and Figure 3b respectively. Note that there are no results reported for WSPDR(α) since deriving the job order following the defined priority rule and the ETWC is conducted in almost zero time for instance sizes

with $n \leq 10$.

In contrast to the exact approaches, the development of run times on the logarithmic scale is markedly flatter, reflecting the polynomial rather than exponential scaling of the BaB(BS,w_{BS}). For fixed w_{BS} the average run time is linear in the number of jobs n. Even the most expensive configuration, BaB(BS,500), requires merely about 0.68 sec for solving instances with $n = 10$ and thus remains remarkably below the run times of the exact BaB approaches at the same instance size. For both, Figure 3a and Figure 3b, the graphs for $w_{\text{BS}} \in \{100, 200, 500\}$ and $n \leq 5$ coincide. This holds since the total number of nodes per search tree level is less than 100 and, thus, the same nodes are branched for all values of w_{BS} listed above. Further, employing the dTH policies for sequence evaluation in BaB(BS,w_{BS}) reduces the run times by 80% on average. The TS exhibits the highest run times among the heuristics reported in Figure 3, with run times of 0.94 sec on average for $n = 10$. Thereby, it still solves all instances in $\mathcal{I}$ substantially faster than the exact solution approaches.

	BaB(DFS)	$\widetilde{\text{BaB}}$(DFS)	BaB(BS,500)	$\widetilde{\text{BaB}}$(BS,500)	TS	$\widetilde{\text{TS}}$
$n_{\text{det}} = 1$	27.67	3.19	0.17	0.04	0.23	0.01
$n_{\text{det}} = 3$	76.09	5.10	0.30	0.05	0.39	0.02
$n_{\text{det}} = 5$	110.48	6.65	0.43	0.05	0.54	0.03
$m = 1$	24.09	1.55	0.10	0.01	0.13	0.01
$m = 2$	118.74	8.41	0.50	0.08	0.63	0.03

Table 6. Average run times (in sec) depending on the number of deteriorations n_{det} and the deterioration precision m.

Finally, we briefly examine the two remaining complexity drivers, the number of deteriorations per job n_{det} and the deterioration precision m, with run times averaged over all instances in $\mathcal{I}$ with the respective parameter value reported in Table 6. For both parameters, we restrict the presentation to representative solution approaches. This is justified by the foregoing observations, since BaB(DFS) and BaB(CBFS,g_{CBFS}) differ only marginally in their run times and the BaB(BS,w_{BS}) configurations behave alike but linearly scaled. As one would expect, run times increase with both parameters. The effect of the deterioration precision m is particularly pronounced, as a higher precision refines the discretization of the ML and thus enlarges the state space underlying both the P-DP evaluation and the bound computations. This increase is observed across all solution approaches, including those relying on dTH policies for sequence evaluation, since their exact evaluation iterates over all possible deteriorations, which on the implementation level amounts to a forward DP evaluation that likewise grows with m. For BaB(DFS), the step from $m = 1$ to $m = 2$ raises the run time from about 24 to 119 sec and thus constitutes the single most demanding intensification in the experiments conducted so far.

Overall, all three parameters increase the computational effort of the proposed approaches, although to different extents. While the number of jobs n primarily drives the combinato-

rial complexity of the sequencing decisions, the number of deterioration realizations n_{det} and the precision m directly affect the size of the state space that must be evaluated within the P-DP. Among the considered parameters, increasing the precision has the strongest impact on the run times, indicating that the evaluation of ML states constitutes the dominant computational challenge.

5.3.2. Solution quality

We evaluate solution quality in terms of the achieved ETWC. Since BaB(DFS) and BaB(CBFS,g_{CBFS}) are exact approaches, both provide identical optimal solutions whenever they terminate successfully. We therefore use BaB(DFS) as the reference approach throughout this section. Besides the equivalent solution quality, BaB(DFS) remains applicable for all instances with $n \leq 10$, whereas some BaB(CBFS,g_{CBFS}) variants exceed the available memory for $n = 10$ as reported in Section 5.3.1. Additional tests confirm that BaB(CBFS,g_{CBFS}) yields no further insights beyond BaB(DFS), so we omit reporting its results in the following. Note that approaches employing dTH policies during sequence generation are assessed based on the exact ETWC of their final sequence using the P-DP policy. An instance is counted as solved to optimality if the resulting ETWC coincides with the optimum ETWC obtained by BaB(DFS).

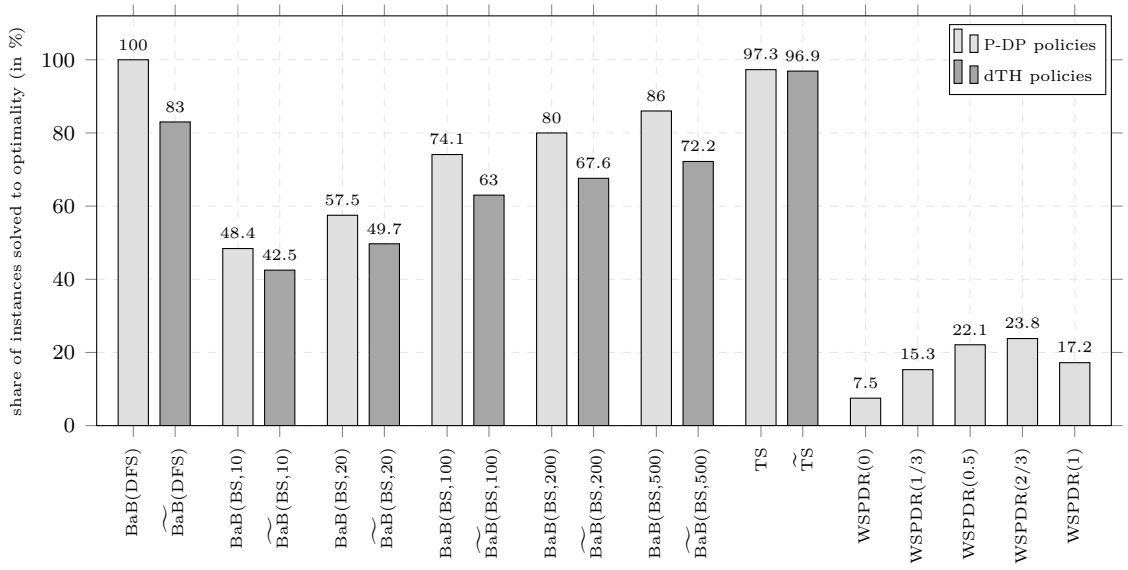


Figure 4. Share of instances solved to optimality per solution approach for JSMPP.

Figure 4 reports the share of instances solved to optimality for all considered solution approaches. The TS performs best among all heuristic solution approaches and solves about 97.3% of $\mathcal{I}$ to optimality. Thereby, we observe only a minor difference of about 0.4% when the candidate sequences in the TS iterations are evaluated using dTH policies instead of the exact P-DP policies. In contrast, for $\widetilde{\text{BaB}}$(DFS) the share of instances solved

to optimality is reduced to 83%. For the BaB(BS,w_{BS}) approach variants the expected pattern can be observed. With increasing beam width w_{BS} the share of optimally solved instances increases monotonically for both sequence evaluation approaches. It ranges from 48.4% for $w_{\text{BS}} = 10$ to 86% for $w_{\text{BS}} = 500$. Here it is noticeable that the optimally solved share for BaB(BS,500) exceeds the share for $\widetilde{\text{BaB}}$(DFS) by 3%. This indicates that the accuracy of the sequence evaluation can be as decisive as the completeness of searching the solution space. On the other hand, the results for $\widetilde{\text{BaB}}$(BS, w_{BS}) are worse than using the exact sequence evaluation whereby the gap increases from 5.9% for $w_{\text{BS}} = 10$ to 13.8% for $w_{\text{BS}} = 500$ which quantifies the loss in solution quality caused by the approximate sequence evaluation.

The solution approaches WSPDR(α) perform worst among all solution approaches and at most 23.8% of the instances are solved to optimality using $\alpha = 2/3$. Excluding deterioration by setting $\alpha = 1$ in WSPDR(α) yields the optimal solution for 17.2% of the instances, whereas the share drops to 7.5% when, conversely, processing times are excluded by setting $\alpha = 0$. This observation can be explained by the fact that the effective processing times are actual processing delays, while the expected deterioration is merely an equivalent for the delay that a PMA of this extent would cause, without any guarantee that this delay will actually occur. These results reveal that a balanced combination of both components is essential.

We next investigate the magnitude of the ETWC deviations from the optimum solution. Table 7 reports the average gap over all instances (gap_{all}) and over all not optimally solved instances ($\text{gap}_{>0}$).

	$\widetilde{\text{BaB}}$(DFS)	BaB(BS,w_{BS})					$\widetilde{\text{BaB}}$(BS,w_{BS})					TS	$\widetilde{\text{TS}}$	WSPDR(α)				
		10	20	100	200	500	10	20	100	200	500			0	1/3	0.5	2/3	1
$\text{gap}_{>0}$ (in %)	0.51	2.46	2.15	1.88	1.59	1.51	2.49	2.08	1.66	1.34	1.14	0.52	0.50	6.68	3.11	2.58	2.68	9.25
gap_{all} (in %)	0.09	1.27	0.92	0.49	0.32	0.21	1.44	1.05	0.61	0.44	0.32	0.01	0.10	6.18	2.64	2.01	2.05	7.67

Table 7. Average ETWC gap (in %) for the heuristic solution approaches over all instances (gap_{all}) and over all not optimally solved instances ($\text{gap}_{>0}$) respectively.

TS and $\widetilde{\text{TS}}$ clearly dominate the heuristic solution approaches, yielding both average gaps close to zero. The heuristic $\widetilde{\text{BaB}}$(DFS) follows with an average gap of 0.09% over all instances and an average gap of 0.51% when restricting the averaging on not optimally solved instances. The observed gaps for BaB(BS,w_{BS}) decrease steadily with increasing w_{BS}, and for every beam width the exact P-DP evaluation yields a lower gap over all instances than its dTH counterpart, which quantifies the quality loss of the approximate evaluation. The solution approaches WSPDR(α) perform weakest, with WSPDR(1) standing out at a $\text{gap}_{>0}$ of 9.25%. As soon as deterioration is accounted for, i.e. for $\alpha < 1$, this gap drops sharply to 2.68% for $\alpha = 2/3$ and reaches its minimum with 2.58% for $\alpha = 0.5$. Reducing α further, however, increases $\text{gap}_{>0}$ to 3.11% for $\alpha = 1/3$ and up to 6.68% for $\alpha = 0$, where processing times are disregarded entirely. These observations can also be applied to gap_{all},

with slightly smaller gaps. Although, the average gaps of all other solution approaches are lower than the gap of the best WSPDR(α) configuration with $\alpha = 0.5$, the clear advantage of the WSPDR(α) lies in the simplicity when determining the job sequence. Thereby, a gap of 2.58% and more than 20% optimally solved instances furthermore supports the defined priority rule.

There are two effects observable in the results presented in Table 7 that demand further explanation. First, comparing two approaches, whereby, one solves fewer instances to optimality, may result in a larger proportion of instances solved nearly optimal. This reduces $\text{gap}_{>0}$ whereby the share of optimally solved instances and consequently also gap_{all} is larger. This effect is, e.g., observed for the direct comparisons of BaB(BS,w_{BS}) and $\widetilde{\text{BaB}}$(BS, w_{BS}) for $w_{\text{BS}} \geq 20$. To be more concrete, $\widetilde{\text{BaB}}$(BS, 500) solves fewer instances to optimality than its exact counterpart, so that its non optimal subset contains many instances that the exact variant solves exactly. These near optimal instances pull down the average, so that the variant using dTH policies attains a smaller $\text{gap}_{>0}$ of 1.14% against 1.51%, whereas its gap over all instances is larger with 0.32% against 0.21%. The valid comparison across approaches should therefore be exclusively based on gap_{all}. Second, $\widetilde{\text{BaB}}$(BS,500) attains the same gap_{all} as BaB(BS,200). This indicates that the completeness of the search and the accuracy of the sequence evaluation act as two distinct levers on solution quality.

We now investigate how solution quality behaves restricted on the disjoint instance subsets $\mathcal{I}_l, \mathcal{I}_e$ and $\mathcal{I}_h$ with *lower*, *equal* and *higher* deterministic processing times (see Table 4). We do so, since we expect the WSPT order to be a good (initial) solution for instances with dominant deterministic processing times compared to the jobs' deterioration.

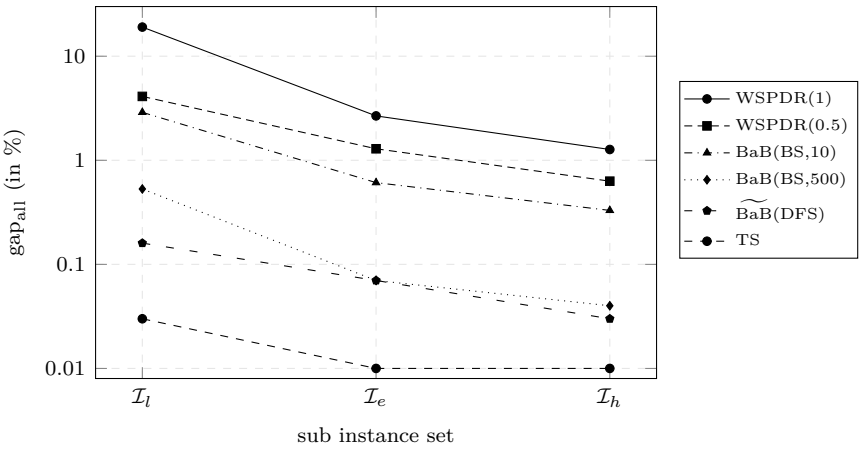


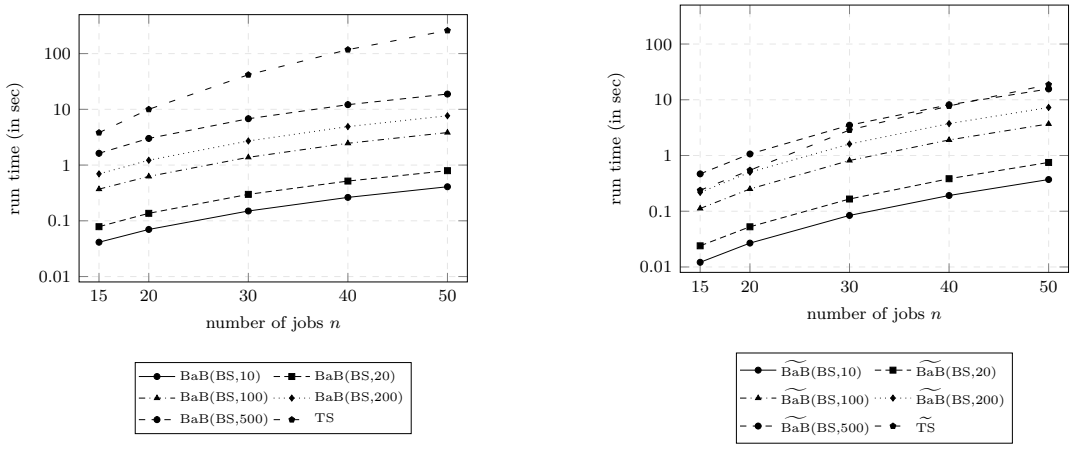
Figure 5. Average ETWC gaps of representative solution approaches across the three instance subsets $\mathcal{I}_l$, $\mathcal{I}_e$ and $\mathcal{I}_h$.

Figure 5 shows the average gap over all instances across the three instance classes. TS and the stronger BaB variants remain robust across all classes, whereas narrow BS configurations suffer more visibly when PMA induced delays dominate deterministic processing

times. All approaches struggle most on $\mathcal{I}_l$, where the deteriorations exceed the deterministic processing times in absolute figures. In these instances we expect a large contribution to the ETWC originating from the durations of MAs rather than from the deterministic processing times of the jobs. A job sequence based on the WSPT order, which corresponds to the selected job sequence in WSPDR(1) ignores deteriorations entirely and is therefore placed at a clear disadvantage, reaching a gap of about 19% on $\mathcal{I}_l$. Accounting for the deteriorations through $\alpha < 1$ mitigates this effect, yet even the best WSPDR(α) variant for $\alpha = 0.5$ retains a gap of about 4% on $\mathcal{I}_l$. For all approaches, the gaps decrease consistently from $\mathcal{I}_l$ to $\mathcal{I}_h$. As deterministic processing times become increasingly dominant, sequencing decisions based on processing time information become more effective and the maintenance related effects lose relative importance.

Overall, TS provides the highest solution quality among the heuristic approaches, while the performance of BaB(BS,w_{BS}) improves steadily as the beam width increases. At the same time, the exact BaB approach supplies the optimal benchmark solutions required for a comprehensive evaluation of heuristic methods. The following section examines whether the observed performance patterns remain stable for larger instances.

5.3.3. Results for the heuristic approaches on larger instances



(a) Sequence evaluation using P-DP policies.

(b) Sequence evaluation using dTH policies.

Figure 6. Average run times of the heuristic solution approaches on instance set $\mathcal{I}^L$ depending on the number of jobs n.

To assess scalability of the proposed heuristic solution approaches, we extend the computational study to instance set $\mathcal{I}^L$ with number of jobs $n \in \mathcal{N}^L$. As the exact approach BaB(DFS) is no longer applicable for instances of these sizes, we consider the heuristic solution approaches BaB(BS,w_{BS}), TS and WSPDR(α). Since no optimal reference is available, the following analysis examines the development of the run times and compares the ETWC for the solutions derived using the heuristic approaches among themselves.

Figure 6 reports the run times on the instance set $\mathcal{I}^L$. All approaches continue to scale in the same manner as on instance set $\mathcal{I}$, without any abrupt increase. The TS is now the most expensive approach and takes about 260 sec for solving instances with $n = 50$, whereas BaB(BS,500) stays below 19 sec. The choice of the policies for sequence evaluation remains the dominant lever, reducing the run time of TS by roughly a factor of 14 and that of BaB(BS,w_{BS}) by about a factor of 5, so that $\widetilde{\text{TS}}$ and $\widetilde{\text{BaB}}$(BS,500) end up at a comparable level of below 20 sec for $n = 50$.

Since optimal solutions are unavailable for these instance sizes, we report in Table 8 the relative deviations from the best solution found across all tested heuristics rather than optimality gaps. The lowest ETWC results from the solution derived using TS for almost all instances. TS attains an average relative deviation close to zero, followed by BaB(BS,w_{BS}), whose relative deviation decreases steadily with increasing w_{BS} from 1.53% and 1.61% for BaB(BS,10) and $\widetilde{\text{BaB}}$(BS, 10) to 1.09% and 1.16% for BaB(BS,500) and $\widetilde{\text{BaB}}$(BS, 500). The solution approaches based on priority rules WSPDR(α) perform weakest, with WSPDR(1) reaching an average gap of about 7.93% that drops to 1.71% when accounting equally for processing times and deterioration through $\alpha = 0.5$. The instance type effect defined on the deterioration and processing time ratio also persists, as all approaches struggle most on the class $\mathcal{I}_l^L$ where the deteriorations exceed the deterministic processing times, reaching for instance an average deviation of 18.96% for WSPDR(1) and about 1.9% for BaB(BS,500).

	BaB(BS,w_{BS})					$\widetilde{\text{BaB}}$(BS,w_{BS})					TS	$\widetilde{\text{TS}}$	WSPDR(α)				
	10	20	100	200	500	10	20	100	200	500			0	1/3	0.5	2/3	1
$\mathcal{I}^L$	1.53	1.43	1.23	1.17	1.09	1.61	1.51	1.31	1.24	1.16	0.02	0.07	6.11	2.30	1.71	1.90	7.93
$\mathcal{I}_l^L$	2.53	2.38	2.11	2.02	1.90	2.68	2.54	2.25	2.15	2.04	0.03	0.11	5.97	3.97	3.59	3.87	18.96
$\mathcal{I}_e^L$	1.19	1.11	0.93	0.86	0.78	1.26	1.17	1.00	0.93	0.84	0.02	0.06	7.52	1.93	1.00	1.24	3.17
$\mathcal{I}_h^L$	0.87	0.80	0.66	0.62	0.58	0.90	0.83	0.68	0.64	0.58	0.01	0.03	4.83	0.99	0.55	0.59	1.66

Table 8. Average ETWC gap over all instances (gap_{all}, in %) for the heuristic solution approaches for the instance set $\mathcal{I}^L$ and its disjoint subsets $\mathcal{I}_l^L, \mathcal{I}_e^L, \mathcal{I}_h^L$.

Overall, the ranking of the approaches and the qualitative patterns carry over from the smaller instances, which confirms that the conclusions drawn on instance set $\mathcal{I}$ remain valid for instance set $\mathcal{I}^L$ with higher numbers of jobs. On the other hand the analysis proves that the developed heuristic solution approaches can handle instances of these more realistic sizes even if the proposed exact solution approach is no longer computationally viable.

6. Conclusion

In this paper, we studied an integrated job sequencing and maintenance planning problem on a single machine subject to stochastic job-dependent deterioration. The problem extends previous work on maintenance planning for fixed job sequences by incorporating the sequencing decision into the optimization process. A solution for JSMPP, therefore, consists of both a job sequence and a PMA policy, with the objective of minimizing the ETWC.

The main methodological contribution of this paper is an exact BaB framework based on a problem specific LB. The proposed bounding scheme exploits an instance split that separates a full sequence into a concatenation of disjoint subsequences. This decomposition yields a valid lower bound taking advantage of the structure of the P-DP used to determine optimal PMA policies for fixed job sequences. Consequently, the developed BaB framework provides two exact approaches capable of solving instances of manageable size and forms the basis for several heuristic BaB variants.

The computational study shows that one developed exact BaB approach solves all considered instances with up to $n = 10$ jobs to optimality, although the computational effort increases substantially with increasing instance size parameters. Among the heuristic approaches, TS achieves the strongest solution quality and solves nearly all instances to optimality. At the same time, the BaB(BS,w_{BS}) variants remain highly competitive and provide a convincing compromise between solution quality and computational effort, especially as the beam width increases. The results further indicate that the accuracy of sequence evaluation is an important driver of solution quality, however, at the cost of higher run times.

Further, the analysis highlights the relevance of accounting for deterioration explicitly in the job sequencing decision. The suggested heuristic priority rules perform poorly when MA induced delays dominate the deterministic processing times. Incorporating expected deterioration into the priority rules improves the solution quality, but does not reach the performance of TS or the heuristic BaB approaches. This once more confirms that the integrated nature of the problem at hand is particularly relevant in settings with MAs having a substantial impact on the ETWC.

A useful property is that almost all solution approaches build on the P-DP to determine PMA policies as a modular component. As long as an extension can be incorporated into the P-DP, the resulting method remains applicable and solutions can be generated for JSMPP using the proposed solution approaches. Possible interesting extensions include stochastic PMA durations, setup times for PMAs, or alternative assumptions on interrupted jobs, such as mandatory restarts after EMAs. Further, we can imagine settings with deterioration distributions being not static but depending on the production process

so far. Such settings are conceivable especially if the deterioration distributions are estimated a priori and the collected information during the processing of jobs gives insights that can help to re-estimate the distribution functions and types. Further, it might be interesting to transfer the results of this work to real-world applications and investigate the applicability using real data sets.

References

Alaswad, S., Xiang, Y. (2017). A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability engineering & system safety*, *157*, 54–63.

Allahverdi, A., Ng, C.T., Cheng, T.E., Kovalyov, M.Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*(3), 985–1032.

Arter, A., Briskorn, D., Gönsch, J. (2026a). Scheduling maintenance activities subject to continuous stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *335*(1), 67–82.

Arter, A., Briskorn, D., Hipp, A., Jaehn, F. (2026b). State-dependent machine availability in scheduling. *International Transactions in Operational Research*, *33*(6), 3769–3799.

Arts, J., Boute, R.N., Loeys, S., van Staden, H.E. (2025). Fifty years of maintenance optimization: Reflections and perspectives. *European Journal of Operational Research*, *322*(3), 725–739.

Bock, S., Briskorn, D., Horbach, A. (2012). Scheduling flexible maintenance activities subject to job-dependent machine deterioration. *Journal of Scheduling*, *15*(5), 565–578.

Briskorn, D., Gönsch, J., Thiemeyer, A. (2024). Scheduling maintenance activities subject to stochastic job-dependent machine deterioration. *European Journal of Operational Research*, *319*(1), 62–78.

Brucker, P. (2004). Scheduling Algorithms. Springer. 4th ed.

Choi, B.C., Park, M.J. (2022). Single-machine scheduling with resource-dependent processing times and multiple unavailability periods. *Journal of Scheduling*, *25*(2), 191–202.

Cui, W.W., Lu, Z. (2017). Minimizing the makespan on a single machine with flexible maintenances and jobs' release dates. *Computers & Operations Research*, *80*, 11–22.

De Jonge, B., Scarf, P.A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*(3), 805–824.

Dehnar Saidy, H.R., Taghavi-Fard, M.T. (2008). Study of scheduling problems with machine availability constraint. *Journal of Industrial and Systems Engineering*, *1*(4), 360–383.

Gawiejnowicz, S., Kononov, A. (2010). Complexity and approximability of scheduling resumable proportionally deteriorating jobs. *European Journal of Operational Research*, *200*(1), 305–308.

Geurtsen, M., Didden, J., B.H.C., Adan, J., Atan, Z., Adan, I. (2023). Production, maintenance and resource scheduling: A review. *European Journal of Operational Research*, *305*(2), 501–529.

Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey, in: Annals of Discrete Mathematics. Elsevier. volume 5, pp. 287–326.

Grigoriu, L., Briskorn, D. (2017). Scheduling jobs and maintenance activities subject to job-dependent machine deteriorations. *Journal of Scheduling*, *20*(2), 183–197.

Hsu, C.J., Cheng, T.E., Yang, D.L. (2011). Unrelated parallel-machine scheduling with rate-modifying activities to minimize the total completion time. *Information Sciences*, *181*(20), 4799–4803.

Ji, M., Hsu, C.J., Yang, D.L. (2013). Single-machine scheduling with deteriorating jobs and aging effects under an optional maintenance activity consideration. *Journal of Combinatorial Optimization*, *26*(3), 437–447.

Kaabi, J., Harrath, Y. (2014). A survey of parallel machine scheduling under availability constraints. *International Journal of Computer and Information Technology*, *3*(2), 238–245.

Kubzin, M.A., Strusevich, V.A. (2006). Planning machine maintenance in two-machine shop scheduling. *Operations Research*, *54*(4), 789–800.

Lee, C.Y. (1999). Two-machine flowshop scheduling with availability constraints. *European Journal of Operational Research*, *114*(2), 420–429.

Lee, C.Y., Lei, L., Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research*, *70*, 1–41.

Lee, C.Y., Leon, V.J. (2001). Machine scheduling with a rate-modifying activity. *European Journal of Operational Research*, *128*(1), 119–128.

Lee, C.Y., Lin, C.S. (2001). Single-machine scheduling with maintenance and repair rate-modifying activities. *European Journal of Operational Research*, *135*(3), 493–513.

Lee, W.C., Wang, J.Y., Lee, L.Y. (2015). A hybrid genetic algorithm for an identical parallel-machine problem with maintenance activity. *Journal of the Operational Research Society*, *66*(11), 1906–1918.

Lei, D., Liu, M. (2020). An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance. *Computers & Industrial Engineering*, *141*, Article 106320.

Lenstra, J.K., Kan, A.R., Brucker, P. (1977). Complexity of machine scheduling problems, in: Annals of discrete mathematics. Elsevier. volume 1, pp. 343–362.

Liu, C., Fan, Y., Zhao, C., Wang, J. (2017). Multiple common due-dates assignment and optimal maintenance activity scheduling with linear deteriorating jobs. *Journal of Industrial and Management Optimization*, *13*(2), 713–720.

Low, C., Ji, M., Hsu, C.J., Su, C.T. (2010). Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Applied Mathematical Modelling*, *34*(2), 334–342.

Luo, W., Xu, Y., Tong, W., Lin, G. (2019). Single-machine scheduling with job-dependent machine deterioration. *Journal of Scheduling*, *22*(6), 691–707.

Ma, Y., Chu, C., Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, *58*(2), 199–211.

Mor, B., Mosheiov, G. (2015). Scheduling a deteriorating maintenance activity and due-window assignment. *Computers & Operations Research*, *57*, 33–40.

Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete optimization*, *19*, 79–102.

Mosheiov, G., Sarig, A. (2009). Scheduling a maintenance activity to minimize total weighted completion-time. *Computers & Mathematics with Applications*, *57*(4), 619–623.

Mosheiov, G., Sidney, J.B. (2010). Scheduling a deteriorating maintenance activity on a single machine. *Journal of the Operational Research Society*, *61*(5), 882–887.

Ogunfowora, O., Najjaran, H. (2023). Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. *Journal of Manufacturing Systems*, *70*, 244–263.

Pinedo, M. (2009). Planning and scheduling in manufacturing and services. Springer. 2nd ed.

Rustogi, K., Strusevich, V.A. (2014). Combining time and position dependent effects on a single machine subject to rate-modifying activities. *Omega*, *42*(1), 166–178.

Rustogi, K., Strusevich, V.A. (2015). Single machine scheduling with time-dependent linear deterioration and rate-modifying maintenance. *Journal of the Operational Research Society*, *66*(3), 500–515.

Sanlaville, E., Schmidt, G. (1998). Machine scheduling with availability constraints. *Acta Informatica*, *35*(9), 795–811.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, *121*(1), 1–15.

Shabtay, D., Zofi, M. (2018). Single machine scheduling with controllable processing times and an unavailability period to minimize the makespan. *International Journal of Production Economics*, *198*, 191–200.

Smith, W.E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, *3*(1-2), 59–66.

Strusevich, V.A., Rustogi, K. (2017). Scheduling with time-changing effects and rate-modifying activities. Springer.

Sun, X., Geng, X.N. (2019). Single-machine scheduling with deteriorating effects and machine maintenance. *International Journal of Production Research*, *57*(10), 3186–3199.

Sun, X., Liu, T., Geng, X.N., Hu, Y., Xu, J.X. (2023). Optimization of scheduling problems with deterioration effects and an optional maintenance activity. *Journal of Scheduling*, *26*(3), 251–266.

Van Horenbeek, A., Pintelon, L., Muchiri, P. (2010). Maintenance optimization models and criteria. *International Journal of System Assurance Engineering and Management*, *1*(3), 189–200.

Wang, T., Baldacci, R., Lim, A., Hu, Q. (2018). A branch-and-price algorithm for scheduling of deteriorating jobs and flexible periodic maintenance on a single machine. *European Journal of Operational Research*, *271*(3), 826–838.

Woo, Y.B., Jung, S., Kim, B.S. (2017). A rule-based genetic algorithm with an improvement heuristic for unrelated parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Computers & Industrial Engineering*, *109*, 179–190.

Xue, P., Zhang, Y., Yu, X. (2014). Single-machine scheduling with piece-rate maintenance and interval constrained position-dependent processing times. *Applied Mathematics and Computation*, *226*, 415–422.

Yang, D.L., Cheng, T., Yang, S.J. (2014). Parallel-machine scheduling with controllable processing times and rate-modifying activities to minimise total cost involving total completion time and job compressions. *International Journal of Production Research*, *52*(4), 1133–1141.

Yang, D.L., Yang, S.J. (2013). Unrelated parallel-machine scheduling problems with multiple rate-modifying activities. *Information Sciences*, *235*, 280–286.

Yang, S.J., Yang, D.L. (2010). Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. *Omega*, *38*(6), 528–533.

Zhang, X., Wu, W.H., Lin, W.C., Wu, C.C. (2018). Machine scheduling problems under deteriorating effects and deteriorating rate-modifying activities. *Journal of the Operational Research Society*, *69*(3), 439–448.

Zhao, C.L., Tang, H.Y., Cheng, C.D. (2009). Two-parallel machines scheduling with rate-modifying activities to minimize total completion time. *European Journal of Operational Research*, *198*(1), 354–357.

Zhao, X., Chen, P., Tang, L.C. (2025). Condition-based maintenance via markov decision processes: A review. *Frontiers of Engineering Management*, *12*(2), 330–342.

Zhong, X., Ou, J., Wang, G. (2014). Order acceptance and scheduling with machine availability constraints. *European Journal of Operational Research*, *232*(3), 435–441.

Zhou, H., Tsai, Y.C., Huang, S., Chen, Y., Chou, F.D. (2021). Single-machine scheduling with fixed periodic preventive maintenance to minimise the total weighted completion times. *Mathematical Problems in Engineering*, *2021*(1), 8891322.

Zhu, Z., Zheng, F., Chu, C. (2017). Multitasking scheduling problems with a rate-modifying activity. *International Journal of Production Research*, *55*(1), 296–312.