Integrating Competencies into Preventive Maintenance Scheduling with Answer Set Optimization

Anssi Yli-Jyrä[1(✉)], Heini Ikävalko[2], and Tomi Janhunen[1,2]

[1] Tampere University, Tampere, Finland
{anssi.yli-jyra,tomi.janhunen}@tuni.fi
[2] Aalto University, Espoo, Finland
heini.ikavalko@aalto.fi

Abstract. The maintenance optimization of multi-component machines has been recently formalized as an Answer Set Optimization (ASO) problem based on component selection and grouping of overlapping maintenance intervals. The motivation of the current work is to develop an extension that would integrate resources and availability constraints into this maintenance model. This article outlines an extended ASO model with the primary focus on modeling and optimizing costly maintenance resources, culminating in cost savings facilitated by the progressive development of workforce competence. The model presented in this work extends the cost function of the prior ASO formalization in a modular way with additional cost priorities concerning parallelism, workforce, and expertise. Due to the presented extensions, the complexity of the integrated maintenance model increases compared to the prior formalization.

1 Introduction

Maintenance optimization [16,25,27] contributes to the availability and reliability of industrial production lines by optimizing the cost-efficiency of the maintenance activities. For example, a lazy, evenly distributed care plan ensures a healthier machinery condition over the lifetime than a restlessly exhaustive plan that is completed early, leaving the aging machinery without spare parts. Therefore, the maintenance scheduling problem is dissimilar to the makespan minimization which is a common objective in job-shop scheduling problems.

It is natural to view optimized maintenance as a digitalized *product-service system* [20] that integrates into its industrial context. The focus of maintenance is then not limited to corrective maintenance operations but it extends to scheduled, opportunistic, condition-based, predictive, and prescriptive maintenance [25]. It also integrates tightly with the operation environment, taking into account the availability for production scheduling and the allocation of scarce maintenance resources [6,8,9,16,26], as illustrated in Fig. 1. Therefore, *integrated maintenance scheduling* differs vastly from classical planning and

A. Meier and M. Ortiz (Eds.): FoIKS 2024, LNCS 14589, pp. 381–400, 2024.
https://doi.org/10.1007/978-3-031-56940-1_21

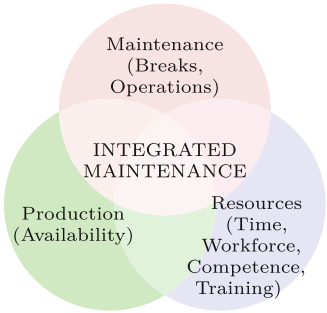

Fig. 1. Integrated maintenance in the context of production and resource allocation

scheduling problems and presents a new set of layers and challenges to modeling and efficiency: Besides the stochastic nature of the failure occurrences that the models must reflect, there are challenges related to knowledge representation and scheduling algorithms. In this article, scheduling algorithms constitute the infrastructure on top of which more knowledge on the resource and production environment can be added.

The maintenance scheduling algorithms [16,37] can be viewed through the dichotomy of *online* and *offline* algorithms. Online algorithms such as *reinforcement learning* (RL) [32] can take significant advantage of the partial knowledge of the problem instance and the function computations before the rest of the problem is presented and the policy function is queried to determine the actions of a maintenance schedule. The system's state space is obtained from the machine specifications and state values and the policy function are then computed in advance, through a time-consuming iteration that estimates the value and the action policies at each state. Since the state space of the system grows exponentially to the number of components, a compact representation of the large state space is necessary (cf. the curse of dimensionality) [13]. The downside is that a compacted state space no longer guarantees to find the globally optimal schedule through its local policy. However, recent developments in *deep RL* approaches [7,33] help close the gap between the full and compact state-space representations. In line with this, reinforcement learning has been concluded as one of the most promising algorithms for maintenance optimization [25].

Logic-based frameworks such as *mixed-integer programming* (MIP) [34], *answer set programming/optimization* (ASP/ASO) [3,15,30], and *constraint programming* (CP) [28] are different from state-based methods such as RL. Solvers in logic-based frameworks are typically used as offline algorithms: the whole problem instance, including the machine specification, is provided as a input when the scheduling starts. After the start of the algorithm, some constraints on the feasible states are learned during the optimization. These algorithms can find global optima. However, since many formalizations of maintenance scheduling give rise to NP-hard decision problems (cf. [16] and [35]), the computation of optimal schedules is non-deterministic by nature. Although the search can

be heavily constrained, the performance and scalability of these algorithms are rightful concerns in practice, giving a motivation for heuristic bounds, pruning constraints, and approximations in logic-based approaches. Nevertheless, logic-based optimization is widely used in practice. It is a promising alternative for interpretable model formulation, problem solving, and multi-target optimization as logical formalisms allow a high-level description of the problem knowledge.

The current work is methodologically framed within the logic-based ASO framework that is well-suited for scheduling and planning tasks [1,10,12,15,21]. The approach is based on an existing formulation [35] of the *preventive maintenance scheduling* (PMS) problem of multi-component machines. Due to the general intractability of the PMS problem [35], we assume parallel development of efficient solving techniques for the problem, potentially exploiting fixed parameters and machine specifications. For example, more scalable encodings based on decompositions, symmetry breaking and pruning constraints have contributed towards this direction [35,36]. More specifically, we extend existing ASP encodings and the related concepts of PMS [35] with some aspects of resource allocation. Maintenance requires sufficient resources to implement maintenance operations and to avoid delays and additional costs, such as hiring an outsourced workforce. For example, the number and duration of maintenance breaks must be adequate for the machine. Each break appears as an *opportunity* in maintenance scheduling, as a *bounded resource* in resource scheduling, and as a *lack of availability* in production scheduling. Seen this way, all three aspects of integrated maintenance (Fig. 1) will be visited although our extensions do not properly combine the planning or scheduling aspects of production to maintenance.

On a conceptual level, the current work advocates *competent maintenance professionals* as the most important type of resource in maintenance optimization. The claim is that the scheduling of the maintenance professionals can be easily integrated into maintenance scheduling. This integration is an important extension as employees form an infra-like long-term investment. The competencies of the employees should be developed and managed to reduce outsourced services and improve the company's efficiency. The recent study [22] recognizes that a manufacturer can develop its capability for service delivery, especially through infrastructure development and management. The hired employees represent a socially complex combination of human resources that are deployed to achieve a desired end goal in product-service systems [31]. At the operational level, developing maintenance capability deals with managing the competence and behavior of the workforce and the diversity in the individual *capabilities* of the maintenance personnel [18,23]. Overall, the current model divides personnel-related resources into three categories further characterized below:

1. **Quantitative Resources** – Realizing maintenance operations within a bounded time frame (i.e., the maximum duration of the maintenance break) by a bounded number of maintenance professionals.
2. **Competence Resources** – Delivering maintenance service with a bounded pool of experts for each component.

3. **Training Resources** – Developing professional competencies of the employees progressively via training.

These resource categories are dealt with by three extensions to the ASP encoding of the core PMS model [35]. Section 2 recaps the core model and outlines the formal interfaces between the three extensions. The three extensions will be called *resource models* (RM1, RM2, RM3) and described in the subsequent three sections. Their purpose is to formalize a minimal understanding of scheduling concerning the three categories of personnel resources as follows:

– Section 3 presents the time-bound service realization as RM1;
– Section 4 formalizes competence requirements as RM2; and
– Section 5 introduces apprenticeship and training as RM3.

The cost-based integration of the three resource models under the cost-based scheduling objective is outlined in Sect. 6. The ultimate goal of maintenance optimization is to improve the cost-effectiveness of maintenance operations from the product-service point of view. This goal assumes that everything can be uniformly measured as financial rewards or costs. It is easy to postulate some relative costs among different quantitative resources and individuals with the disclaimer that they are nearly arbitrary. Since maintenance scheduling, availability, and resources appear in the cost-based objective function side by side, it is perhaps fair to summarize that the current work *presents a uniformly weighted ASP-based model for integrated maintenance scheduling.*

Finally, Sect. 7 concludes this article with discussion. The article extends an existing logic-based approach to PMS optimization by formalizing a few aspects of resource allocation. The main contribution of the article resides in the incorporation of management insights into preventive maintenance scheduling.

2 Overview of Maintenance and Resourcing Problems

In the sequel, the preventive maintenance scheduling (PMS) problem is exclusively understood in the abstract sense of [35]. The problem plays a fundamental role as it lays out the scene for further constraints and objective functions, and it is parameterized by the number of maintenance breaks ($\mathtt{b}$) and the time horizon ($\mathtt{h}$). A *solution* to the PMS problem assigns for each break $i \in \{1, \ldots, \mathtt{b}\}$,

– its discrete location b_i on the timeline, i.e., $1 \leq b_i \leq \mathtt{h}$,
– a group of components $g_i = \{c \in C \mid \mathtt{s}(i, c)\}$ to be maintained at i.

An optimal solution minimizes the objective function that measures the overall *miscoverage* of components in the schedule, reflecting the absolute deviation of component-wise renewals from their recommendations that are given in machine specifications. Miscoverage may take place in two different ways. *Under-coverage* means overdue component renewals, increasing the risk for aging-related component failures. *Over-coverage* arises if renewals occur too early in the component's lifetime, increasing the cost of maintenance. The optimal schedule on the left of

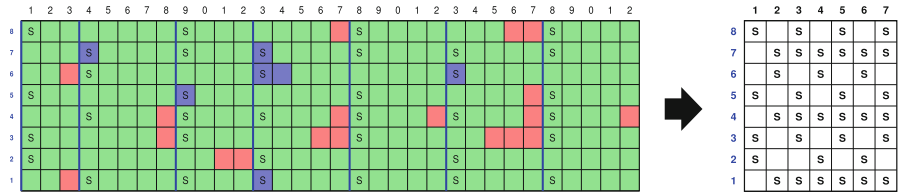

Fig. 2. A scheduling timeline with 32 time steps and 7 maintenance breaks [35].

Fig. 2 covers the maintenance of an 8-component machine over 32 time steps. Red and blue cells denote under- and over-service, respectively, for a particular component and time step. The maintenance breaks are indicated by blue vertical lines while the cells marked with letter "S" denote the renewal of the component in question. As a result, the component is valid for a component-specific time indicated by sequences of (blue or) green cells starting from cells where a component is renewed. For the components 1–8 in question, the respective recommended service intervals are 5, 10, 7, 4, 9, 11, 5, and 8.

The *integrated maintenance problem* of the present work focuses on combining the core maintenance scheduling model with maintenance-related resource models. Constraints related to production planning and scheduling are left for further work but the availability of the production machinery will be measured indirectly, through the total number of maintenance breaks that are excepted from the time available for production operation. In the following, we define an *interface* that extracts the core of an (optimal) solution, i.e., an optimal maintenance schedule in the space of feasible candidate schedules. The core is a projection of a full schedule. It views time as a sequence of breaks $1, \ldots, \mathtt{b}$ rather than points of time, ignoring the corresponding time steps $b_1, \ldots, b_{\mathtt{b}}$ in the scheduling timeline, see Fig. 2. The schedule on the right of Fig. 2 condenses service selection information $g_1, ..., g_{\mathtt{b}}$ as the basis for resource allocation.

Several ASP encodings of the maintenance scheduling problem exist [35,36]. The shared core of the encodings is now assumed to define three predicates:

- `i(I)` – there is a maintenance break indexed by `I`;
- `c(C)` – there is a component whose identity is `C`; and
- `s(I,C)` – the component `C` is to be maintained (serviced) during a break `I`.

The maintenance scheduling problem is extended with maintenance-related resource scheduling that is divided into three scheduling needs, namely the quantitative resources (Sect. 3), competence resources (Sect. 4), and training resources (Sect. 5). Each of these extensions can be seen as a successive refinement to the maintenance scheduling problem, see Fig. 3. Each refinement narrows the search toward a smaller space of feasible maintenance schedules. The interface of core maintenance scheduling is shared by all refinements. The refinements, i.e. the resource models, constrain the space of feasible schedules through the interface. The operation scheduler extends the interface of maintenance scheduling with one predicate `op(I,C,Q)` denoting that during the break `I`, component `C` is being

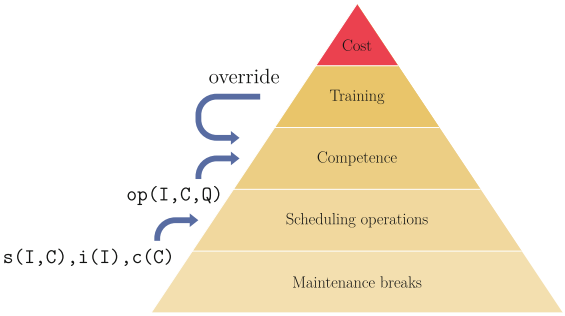

Fig. 3. Refining the maintenance scheduling problem

maintained by the required number of professionals at time slice `Q`. The resulting interface builds on four predicates (i.e., `i/1`, `c/1`, `s/2`, and `op/3`) and it is employed by both the competence model and the training model. Since the training model is a proper extension of the competence model, it overrides and partially replaces the latter model if deployed.

The integration of the maintenance scheduler with the extensions is implemented first as hard resource models RM1–RM3. The *hard integration* restricts the space of feasible schedules since the resources are limited and non-negotiable. These restrictions may result in over-constrained problem instances and the failure of integrated scheduling, as demonstrated in Sect. 6. In contrast, *soft integration*, to be developed in the end of Sect. 6, assumes the availability of further resources, but with extra cost. There is a trade-off between additional resources and the miscoverage of a schedule. Some extra resources can be deployed, but the high costs of such resources force one to find a balance between an excellent but costly schedule and a moderate and affordable schedule.

3 Scheduling Service Operations During Breaks

Our first extension and refinement of maintenance scheduling (RM1) deals with quantitative personnel resources required by the implementation of maintenance operations during maintenance breaks. To characterize the organization of maintenance operations serially and in parallel, and the workforce to carry out the operations, we assume three numeric parameters:

- `q` – the number *time slices* inside each maintenance break,
- `t` – the number of maintenance *tracks* running in parallel, and
- `p` – the number of *professionals* available for service operations.

In addition to these global parameters, there is a database of resource requirements for each component. For simplicity, we assume that the resource requirements of a particular component remain the same for every break when the component is maintained. Thus, it is possible to estimate the overall need for

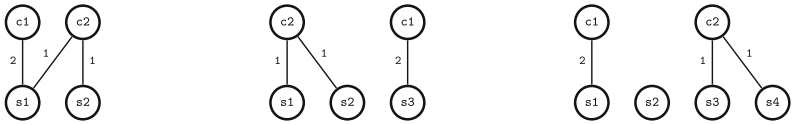

Fig. 4. Some ways to schedule the maintenance of components `c1` and `c2` during a maintenance break. Left: 2 tracks, 2 time slices, 3 professionals. Center: 1 track, 3 time slices, 2 professionals. Right: 1 track, 4 time slices, 2 professionals.

service personnel although for each break, the actual number of professionals depends on the set of components being serviced. The required resources are specified with predicates taking the component `C` as the first argument:

- `durreq(C,D)` – the maintenance of component `C` requires `D` time slices and
- `proreq(C,P)` – the maintenance of component `C` requires `P` professionals.

Example 1. Consider two components `c1` and `c2` whose requirements are given by `durreq(c1,1)` and `proreq(c1,2)`, and `durreq(c2,2)` and `proreq(c2,1)`, respectively. Then, suppose that these components are being maintained during the same break. If the number of professionals is two, the tasks during the break will take at least three time slices. However, if the number of professionals is increased by one, then the tasks can be completed within 2 time slices.

This situation is illustrated by the first diagram in Fig. 4. The upper row of vertices denotes components and the lower row denotes time slices. An edge between a component and a time slice indicates that the component is being maintained during the slice. The label of the edge indicates the resource taken, i.e., how may professionals are working on the component. ■

Following the idea in Example 1, the goal of the service operation scheduler is, for each maintenance break `I`, to

1. choose, for each scheduled service operation `s(I,C)`, a time slice during which the service of `C` is started and, given the request `durreq(C,D)`, reserve `D` consecutive time slices in total for completing the service of `C`;
2. check that at most `t` parallel operations are allocated to each time slice; and
3. ensure for each time slice `Q` that the number of professionals is not less than the number requested by components `C` being maintained during `Q`.

The allocation of a time slice `Q` to the component `C` during break `I` is encoded by a fact `op(I,C,Q)`. If the component requires multiple time slices for service, there will be several such facts. It is also worth observing that there is no need to allocate tracks explicitly and, hence, it is possible to avoid certain symmetries arising from the identities of components. Since one component may require multiple professionals, the number `t` of parallel tracks can be exceeded by the total number `P` of professionals working simultaneously.

Listing 1. ASP encoding of the operation scheduler

```
% Assumes q, s/2, durreq/2, i/1, proreq/2. Provides: op/3.
% Choose one time slice to start each operation
resource_model1 :- not resource_model2, not resource_model3.
slice(1..q).
{ opstarts(I,C,Q) : slice(Q), slice(Q+D-1) } = 1 :- s(I,C), durreq(C,D).

% Reserve the track from the starting slice to the slice of completion
op(I,C,J) :- opstarts(I,C,Q), durreq(C,D), J=Q..Q+D-1.

% Limit the tracks during any given slice
:- i(I), slice(Q), t < #count{ C : op(I,C,Q) }.

% Limit the professionals during any given slice
p(I,Q,M) :- i(I), slice(Q), M = #sum{ P,C: op(I,C,Q), proreq(C,P) }.
:- i(I), slice(Q), p(I,Q,P), p < P.
```

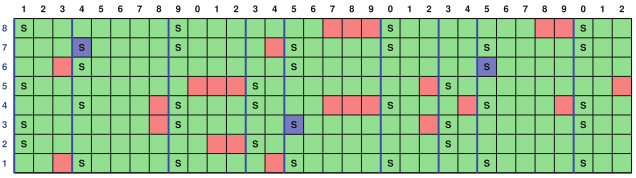

Fig. 5. The effect of enforcing resource constraints when the number of breaks is 9.

As regards encoding the service operation scheduler in ASP, it can be implemented in a few lines of code[1] (see Listing 1). The present encoding is not intended to be an efficient encoding of a knapsack-style packing problem. It merely demonstrates the purpose of the operation scheduler when checking the feasibility of service operations as well as the required personnel. Although some symmetries are already avoided, some remain as can be recalled from Fig. 4.

When it comes to more realistic industrial settings, one could introduce additional constraints that control the order in which service operations can be performed. To settle such further needs, an encoding that combines features from scheduling and planning problems might be in order.

Example 2. To illustrate the effect of resource constraints, we provide a maintenance schedule in Fig. 5. Due to resource requirements, fewer components can be serviced during maintenance breaks. To compensate, we increase the number of maintenance breaks to 9. The miscoverage of an optimal schedule is 27 which is one more than previously (cf. Fig. 2). This reveals that the number of service breaks is also an essential resource affecting the quality of schedules obtained. ■

[1] The ASP encodings are publicly available under https://github.com/asptools and we have used Clingo (v. 5.6.2, under MacBook Pro) as the solver in experiments.

4 Competence-Aware Maintenance

Our second resource-based extension of maintenance scheduling (RM2) addresses competencies required from the workforce. There are different kinds of skills and competencies that professionals can have. Some skills are specifically required by the machinery itself or necessary for maintenance professionals in general (cf. [19]). Key skills, such as language proficiency, professional certificates, safety and risk management cards and so on, are identified already in the job descriptions and interviews of the employees or sorted out during introductory courses. Further skills are identified through industrial benchmarks, external references, simulator hours, track records, and interviews.

Knowledge about crucial skills is commonly represented as a *competence matrix*, affecting permits, teams, and the financial rewards of individuals. In a simplified ASP encoding, a competence matrix can be represented by facts `expert(P,C)` stating that professional `P` has an expert-level proficiency of maintaining component `C`. In addition, predicate `expreq(C,E)` specifies how many experts are required by the component `C`. Then, based on such knowledge, the goal of *competence-aware* maintenance scheduling is to assign professionals to service operations so that component-specific requirements are met.

To find a schedule that implements the requirements in practice, there must be a sufficiently large pool of experts able to service the components of a machine as scheduled. Let us now assume that `op(I,C,Q)` holds, i.e., component `C` is being serviced during break `I` and at slice `Q`. The RM1 model ensures that there are sufficiently many professionals available for all maintenance operations occurring at each break `I` and slice `Q`. Among these, there are, according to `expreq(C,E)`, `E` such experts of component `C` that are dedicated to `C` during the time slice.

The assignment of static experts in RM2 is formalized on lines 3–4 in Listing 2. Section 5 generalizes this assignment to dynamic experts that develop over time. To ensure the allocation of exactly `E` experts for each operation `op(I,C,Q)`, the assignment of a professional `P` to a component `C` at break `I` and time slice `Q` is encoded (lines 3–4) with an atom `e(I,(C,Q),P)`, where the pair `(C,Q)` links the component `C` to the slice `Q`. Similarly, the assignment of professionals that are not acting as experts is expressed using an atom `o(I,(C,Q),P)` (o=other professional) on lines 8–9. Experts (and other professionals) should work only on single component at a time (lines 5 and 10). Multi-tasking as an expert and a non-expert is also ruled out on lines 13–14.

The presented model of competence-awareness is simplistic and based on certification-based listings in the competence matrix. Such matrices tend to forget tacit, proactive knowledge and generic skills, such as the ability to deploy a growing range of digital services:

– **Proactive skills.** Maintenance work includes proactive tasks to foresee the breakdown of the system [29]. The daily tasks in maintenance involve monitoring the current situation and the overall quality status of the production line. As an example, a maintenance worker describes their role at work: "*Here, when production is underway, it is monitored for the condition of the line to*

Listing 2. ASP encoding of the competence scheduler

```
p(1..p).
resource_model2 :- not resource_model3.
{ e(I,(C,Q),P) : p(P), expert(P,C) } = EN :-
    op(I,C,Q), expreq(C,EN), resource_model2.
:- e(I,(C1,Q),P), e(I,(C2,Q),P), C1 < C2.

% For each C with PN>EN>0 assign PN-EN other pros to op(I,C,Q)
{ o(I,(C,Q),P) : p(P), not e(I,(C,Q),P) } = PN-EN  :-
    op(I,C,Q), expreq(C,EN), proreq(C,PN).
:- o(I,(C1,Q),P), o(I,(C2,Q),P), C1 <  C2.

% Check that nobody works in two roles at the same time
iso(I,Q,P) :- o(I,(C,Q),P).
:- iso(I,Q,P), e(I,(C,Q),P).
```

ensure everything is okay. I walk around, observe, and listen. Even the nose is in use, as it can reveal certain things. If I notice something, I record it in the fault log, noting down the symptoms and the location. It then moves on to job planning, and from there, we begin to determine the urgency of the matter."

– **Generic digital skills.** Individuals with strong digital skills and those who quickly adapt to new things can assist colleagues and support others at work. Their active engagement often makes them role models and educators in the workplace, beyond their designated roles. There is evidence of individual differences in using digital systems, based on workers' experiences. According to [18], the enactment of digital systems varies based on attitudes, competence, and individual activities. The study also notes instances where individuals, despite positive attitudes and strong skills, may not engage in helping others in their work due to being occupied with their own.

To model generic and proactive skills or a variety of skill types and their role in maintenance in general, the respective generalizations to the present model would be needed. Such generalizations go beyond our illustrative purposes.

5 Scheduling Training During Breaks

The resource model RM2 is naive in the sense that competencies are considered as static properties. A static competence model is quite limited as it ignores the dynamic development of skills and continuous learning:

– **Continuous learning.** The competencies of individuals develop over time. The longer one practices, the more competent the person becomes at a job. Similarly, the maintenance team's competence to execute maintenance actions changes dynamically over time.
– **Knowledge and experience sharing.** Sharing tacit and generic knowledge is crucial for the effectiveness of maintenance teams [18,29]. These skills are difficult to formalize as competence matrices, but they can be deployed and facilitated through adequate procedures and experiential learning [24].

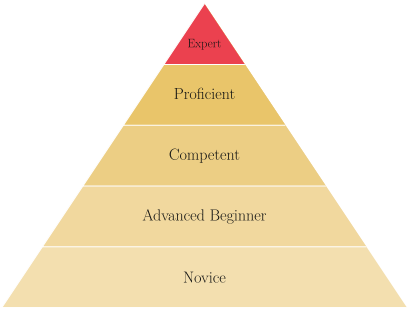

Fig. 6. Five stages of proficiency according to Benner [2].

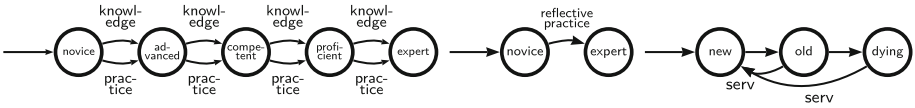

Fig. 7. States of the *From novice to expert* model (Left) and its simplification (Middle) vs. a simple model of component-wise deterioration and reset service (Right).

Our third resource model (RM3) addresses the limitations of the static competence model. The goal is to extend the notion of expertise to a more dynamic direction—partly capturing the effects of continuous learning and experience sharing. This process can be supported in many respects:

- **Instruction.** Some individuals may feel that following instructions is sufficient for their tasks. For instance, individuals with initially weak digital skills but a positive attitude may actively strive to enhance the efficient use of digital systems in their work. To cope with instructions, some individuals proactively record conversations with key users to learn from them [18].
- **E-Learning.** Competence of maintenance workers can also be enhanced through digitally facilitated knowledge processes [17]. Increasing knowledge through activities like reading instructions in ERP or watching videos represent such ways. According to one maintenance worker: "*Often, the video would be the best option* [to develop competence] *because, in my opinion, converting it into written form takes more time, and conveying every little nuance with pictures and words is much more challenging.*"
- **Experience.** Competence can be developed through practical experience: "*Yes, I mainly believe that there's no other way than by doing, observing, walking around, and analyzing. One must understand the state of the process, whether this operating sound is now abnormal or if it is normal. In this, I have noticed that there are differences among people. It purely depends on long-term practice and experience.*"
- **Proactive behavior.** The same person, working in the maintenance describes the role of proactivity in developing their competence: "*One who asks won't get lost; in that case, help and assistance are provided. I don't*

Listing 3. ASP encoding of the training scheduler

```
1  % Define expertise as a function of time, progressively:
2  expert(P,C,1)   :- p(P), expert(P,C),     i(1).
3  expert(P,C,I+1) :- p(P), expert(P,C,I),   i(I+1).
4  expert(P,C,I+1) :- p(P), practice(P,C,I), i(I+1).
5
6  % For each operation op(I,C,Q), with expreq(C,EN), assign EN dynamic experts
7  resource_model3.  % Turn off the earlier rule implementation
8  { e(I,(C,Q),P) : p(P), expert(P,C,I) } = EN :- op(I,C,Q), expreq(C,EN).
9
10 % Record apprenticeship as practice.
11 practice(P,C,I) :- o(I,(C,Q),P), expreq(C,EN), EN > 0.
```

know where one would suddenly go to tell someone what they should do if the person doesn't ask. It mainly requires the recipient's initiative to remember to ask, and assistance is provided if the information is not readily available, for example, in work instructions."

– **Shared meetings.** The person also brings up the role of shared meetings with their colleagues: "*Often, in the morning, we have group discussions. If any issues arise or if there has been something showing symptoms, we monitor it through production displays. This leads to discussions, and we address it as a group.*"

These learning means are activated efficiently when an individual is in charge of carrying out maintenance activities with an expert. Novices can develop their competence with the support of experts sharing their tacit knowledge. The expert can explain the tasks in advance, request the novice to study on-line material, and provide support when the task is performed for the very first time.

Benner's *from novice to expert* model (Fig. 6) describes how an individual gains new knowledge and skills and develops from novice to expert [11]. The model identifies five stages of development: *novice, advanced beginner, competent, proficient*, and *expert* (Fig. 7a). The development takes place through knowledge acquisition and reflective practice. Since the immediate goal of this paper is to demonstrate the incorporation of proficiency development into integrated maintenance, two levels of proficiency are distinguished in a simplified version of Benner's model: *novice* and *expert* (Fig. 7b). The development is assumed progressive, i.e., the proficiency of an individual can only increase and no iteration or declining proficiency is considered. This differs from the stages of machine condition development that involves a reset mechanism (Fig. 7c).

An ASP encoding that takes into account progressive expertise and the effects of training is presented in Listing 3 that contains a streamlined *from novice to expert* model. In this model, the progression from novice to expert is easy: During operation `op(I,C,Q)`, a novice professional `P` can gain practice on component `C` as an apprentice (line 11) if the component `C` requires some other professionals to be present besides the experts. If professional `P` was not an expert at break `I`, but gets practice under the supervision of an expert, `P` is considered an expert during the next break (line 4). The worker's existing expertise is propagated

Listing 4. Description of 8 components and the competence matrix of 8 professionals

```
1  comp(1, 5,2).  durreq(1,2).  proreq(1,2).  expreq(1,1).
2  comp(2,10,0).  durreq(2,1).  proreq(2,2).  expreq(2,1).
3  comp(3, 7,0).  durreq(3,1).  proreq(3,1).  expreq(3,1).
4  comp(4, 4,3).  durreq(4,1).  proreq(4,1).  expreq(4,1).
5  comp(5, 9,0).  durreq(5,3).  proreq(5,2).  expreq(5,2).
6  comp(6,11,2).  durreq(6,2).  proreq(6,3).  expreq(6,3).
7  comp(7, 5,4).  durreq(7,1).  proreq(7,1).  expreq(7,1).
8  comp(8, 8,0).  durreq(8,1).  proreq(8,2).  expreq(8,2).
9  cost(1,5). expert(1, 1..8).  cost(2,2). expert(2, 2..6).
10 cost(3,0). expert(3, 7..8).  cost(4,2). expert(4, (1;4;7)).
11 cost(5,2). expert(5, (2;6)). cost(6,0). expert(6, (1;5)).
12 cost(7,2). expert(7, 6..8).  cost(8,0). expert(8, 4..5).
```

further (lines 2–3). These kinds of dynamically computed experts replace static experts when the requirements are checked (lines 7–8).

6 Integration

Hard Integration. Integration with a hard, restrictive resource model increases the risk of an over-constrained problem instance that does not have any solution. A central question is whether there is a way to preserve at least one candidate solution even when all candidates violate at least one constraint. If the miscoverage function, i.e., the search space for core PMS problem [35], is itself total and not restricted by pruning constraints [36], the space contains even schedules that do not consume any resources and are, therefore, not affected by resource models. However, if the solution space is narrowed by both pruning constraints and resource restrictions, the search may be over-constrained, leading to no solution. This is likely in integrated maintenance because the two types of constraints are almost complementary: pruning constraints eliminate many suboptimal solutions while resource restrictions eliminate unrealizable optimal solutions.

Example 3. This example illustrates how resource restrictions degrade the miscoverage of schedules. The small database in Listing 4 specifies the recommended maintenance intervals `R` and the due times `D` of components (`comp(C,R,L)`) for the same eight components whose maintenance schedule appears in Fig. 2a. Resource requirements are also given for each component with `durreq/2`, `proreq/2`, and `expreq/2`. In addition, the expertise (`expert/2`) and hourly salary (`cost/2`) of the eight professionals ($\mathtt{p} = 8$) is explicated by further facts in the listing.

Table 1 samples the output of the integrated maintenance scheduler when hard resource models RM1, RM2, and RM3 are activated. For the sake of easier comparison to prior work [35], assume $\mathtt{h} = 32$ time steps and fix $\mathtt{b} = 7$ breaks to the same locations of the timeline as in Fig. 2a. However, the maintained components at every break are reselected for optimization under integrated maintenance. This is executed by setting controlling parameters `t`, `q`, `p` as indicated in the table.

Table 1. The effect of hard constraints on miscoverage

PMS Core	b	t	q	p	Pruned?	RM1	RM2	RM3
26	7	3	4	5	yes	26	26	26
26	7	3	4	4	yes	27	UNSAT	UNSAT
26	7	3	4	4	no	27	54	54
26	7	2	4	5	no	31	31	31
26	7	2	4	4	no	31	55	55
26	7	2	3	4	no	54	77	**69**
26	7	2	3	3	no	74	83	83
26	7	2	2	2	no	124	140	140
26	7	1	1	1	no	206	206	206
26	7	0	0	0	no	245	245	245

The effect of over-constrained search is seen only when the search space is pruned with further constraints [36]. Pruning is harmless when the resource model is not restrictive ($\langle \mathtt{t}, \mathtt{q}, \mathtt{p} \rangle = \langle 3, 4, 5 \rangle$). The search fails, however, when the solution space is narrowed both by pruning and resources where $\mathtt{p} = 4$.

Consider now searching for globally optimal schedules without any pruning constraints [36]. The increasing resource sensitivity affects miscoverage clearly:

- In RM1, the bounded parallelism (t) and break duration (q) limit the size of groups g_i of maintained components, increasing miscoverage from the original 26 to the range 27–245.
- When RM2 is included, the limited competence of the small maintenance team restricts operational capacity even more. This degrades miscoverage even more, to the range 54–245, although sometimes the competence requirement has no effect on miscoverage.
- When RM2 is upgraded with RM3, the pool of experts grows dynamically towards the horizon. This reduces miscoverage when ($\langle \mathtt{t}, \mathtt{q}, \mathtt{p} \rangle = \langle 2, 3, 4 \rangle$).

If no resources are given, i.e. ($\langle \mathtt{t}, \mathtt{q}, \mathtt{p} \rangle = \langle 0, 0, 0 \rangle$), there still remains a trivial solution (no maintenance). In this empty schedule, 'green cells' occur at the beginning of the timeline only. ■

Soft Integration. In soft integration, the quality of schedules is a function (e.g., a weighted sum or polynomial) of miscoverage and resource consumption. This function is intended to emulate the cost-efficiency of maintenance activities: (i) Miscoverage is an indirect measure for *component-related* cost-efficiency of a maintenance schedule [35]. Over-coverage `oc(C,T)` at a timestep `T` means loss in the remaining lifetime of component `C` while under-coverage `uc(C,T)` correlates to an increased risk of failure for component `C` during timestep `T`. An abrupt failure seldom goes without harm and loss in availability for production. (ii) The

Listing 5. ASP encoding of the integrated objective function

```
1  % Assumes s/2, op/3, i/1, uc/2, oc/2, cost/2, o/3, e/3.
2  maxb(M) :- M = #count{ I: s(I,C)    }.                    % used breaks
3  t(T)    :- T = #count{ C: op(I,C,Q) }, i(I), slice(Q).    % implicit tracks
4  maxt(M) :- M =   #max{ T: t(T); 0 }.                      % used tracks
5  maxq(M) :- M = #count{ Q: op(I,C,Q) }.                    % used slices
6  maxp(N) :- N =   #max{ P: p(I,Q,P); 0 }.                  % used pros
7  __usage(B,T,Q,P) :- maxb(B), maxt(T), maxq(Q), maxp(P). #show __usage/4.
8
9  % Minimize MC and resources at @2 (primary key) and MC at @1 (secondary key)
10 #minimize{ 20@2,C,T,mc  : uc(C,T)                 ;  % uc
11            20@2,C,T,mc  : oc(C,T)                 ;  % oc
12            50@2,I,breaks: I=1..M, maxb(M)         ;  % breaks
13            50@2,Q,slices: Q=1..M, maxq(M)         ;  % slices
14            50@2,T,tracks: T=1..M, maxt(M)         ;  % tracks
15            50@2,P,pros  : P=1..M, maxp(M)         ;  % pros
16             K@2,I,C,P   : o(I,(C,Q),P), cost(P,K) ;  % non-experts
17             K@2,I,C,P   : e(I,(C,Q),P), cost(P,K) }. % experts
```

number of breaks has been previously fixed [35,36]. Now breaks can be seen as a similar quantitative resource as tracks `t`, time slices `q`, and professionals `p`. Every break $1, \ldots, $ `b` takes away some availability of the production machinery. (iii) Additional time slices (`q`) prolong maintenance breaks, and additional parallel tracks (`t`) in maintenance operations increase the simultaneous duties of professionals. Furthermore, simultaneous deployment of many professionals or experts increases personnel costs. Reductions in these resource quantities contribute directly to the *resource-related* cost-efficiency of maintenance as the saved time or salary costs can be used for production. (iv) Finally, the deployment of in-house novices is typically cheap, while general experts and outsourced consultants cost more. This salary differentiation goes back to `cost(P,M)`-facts in the database.

In ASO, we can encode the integrated objective function as a *pseudo-Boolean expression* (Boolean variables with integer coefficients) that specifies costs being minimized. For sake of flexibility, the hard bounds are first set sufficiently high to make resource bounds compatible with acceptable (not necessarily optimal) solutions to the core PMS problem. Second, a sufficient number of experts (employees or consultants) are ensured in the database to avoid infeasible schedules. This would allow resource models to allocate resources whose balanced consumption could be controlled by an objective function that compares schedules.

A possible ASP formalization of the objective function is given in Listing 5. The unary predicates `maxb/1`, `maxt/1`, and `maxq/1` indicate how many breaks, tracks, and time slices are utilized in the schedule. The summands of the objective function are specified on lines 10–17 where notation `@2` tells that this objective function forms the primary (i.e. higher) key to the optimizer. The core PMS scheduler [35] contains another objective function (miscoverage) that will now become a lower ranked key to the optimizer. The primary key is, in principle, enough for soft integration. The secondary key (miscoverage) has two roles: It gives useful information about the solution and contributes to tie-breaking between multiple optimal schedules in a way that is consistent with the core PMS problem. The actual objective function (mc*20+resources*50+salaries) is merely

Table 2. Automatic parameter optimization in cost-based integration

Model	b	t	q	p	Used b	t	q	p	Cost (mc*20+resources*50+salaries)	Mc.
Core	7	7	7	5	7	0	0	0	870 $(26*20+(7+0+0+0)*50+0)$	26
RM1	7	7	7	5	7	2	5	4	1420 $(26*20+(7+2+5+4)*50+0)$	26
RM2	7	7	7	5	7	2	5	4	1535 $(26*20+(7+2+5+4)*50+115)$	26
RM3	7	7	7	5	7	2	5	4	1535 $(26*20+(7+2+5+4)*50+115)$	26
RM1	7	2	3	4	7	2	3	4	1880 $(54*20+(7+2+3+4)*50+0)$	54
RM2	7	2	3	4	7	2	3	4	2415 $(77*20+(7+2+3+4)*50+75)$	77
RM3	7	2	3	4	7	2	3	4	**2261** $(69*20+(7+2+3+4)*50+81)$	**69**

a suggestion based on our experience with the 8-component machine. Further details of the objective function are mainly ASP technicalities that implement the weighted sum. The behavior of the function can be tuned easily by changing the coefficients of the summands.

Example 4. Table 2 shows the results of applying cost-based optimization (Listing 5) in integrated maintenance. This example illustrates that cost-based integration balances between miscoverage and resources and adapts to the high resource costs by performing fewer operations. The pruning constraints [36] are *not* applied and the 7 breaks are still fixed to the same locations of the timeline as in Fig. 2a. In this experiment, resources are bounded from above by the input parameters `b`, `t`, `q`, and `p` as indicated in the table.

The scheduler returns two cost-function values: the integrated cost (primary key for optimization) and the miscoverage (secondary key for information and tie-breaking). No personnel-related resources are allocated with the core PMS scheduler. With enough resources, all resource models can find a schedule with the minimal miscoverage 26 and the table indicates that only a portion of the resources is actually used. But when the resources are bounded to 2 tracks, 3 slices, and 4 professionals, the miscoverage goes up in all resource models. First, RM1 introduces timing and personnel limits, and then RM2 enforces competence requirements. But, since RM3 can develop new competencies on the way, it pays off to hire some experts to teach. This increases the capacity of the professionals, and they can be more effective and perform more operations within the same time limits. The cost-efficiency of maintenance is improved as a result! ■

7 Evaluation and Conclusion

In this article, prior work on *preventive maintenance scheduling* [35,36] (PMS) is extended towards *integrated maintenance* (Fig. 1) that incorporates aspects of production and resource allocation into maintenance. In the extended model, production aspects are implicitly present, as the numbers of maintenance breaks (and time slices within) affect indirectly the availability of the production machinery. The main focus is, however, on resources required by maintenance

actions during maintenance breaks. Such resources relate to the availability of workforce, the recognition of the workers' competencies, and opportunities created by their enhancement via training.

These ingredients are embodied in the *resource models* RM1–3 whose formalization in Sect. 3–5 is our first contribution. These models reveal the desire for incorporating dynamic proficiency binding (i.e., *from novice to expert model* [2]), practice, and tacit knowledge into resource modeling. Continuous learning and experience sharing deserve much deeper and careful analysis than what was possible within this article. The RM3 model takes a step towards *from novice to expert model* in a quite simplified setting, nevertheless demonstrating the potential of logical approaches when modeling continuous learning in the context of PMS.

The encapsulation and interfacing of resource models is itself a technical challenge as this can be done in arbitrarily many ways. Our second contribution is to identify a *minimal interface* for coupling PMS with resource models. The given interface identifies the cores of maintenance schedules for resourcing purposes and enables the separation of concerns when it comes to implementing integrated maintenance. Such an abstraction is made possible by the ASO framework that supports modular knowledge representation and the incorporation of instance-specific information as simple collections of facts. The interface alleviates the development of more efficient encodings for both PMS and resource allocation in separation, although the overall performance depends on their combination.

On top of the three resource models, we further develop a *weight-based* approach to integrated maintenance that maps all resources to monetary values, making their uniform comparison and trade-off possible. Besides being immediately more conceivable by production economists and maintenance developers, it also addresses the consistency of integrated models when conflicting hard constraints cannot be strictly satisfied by any candidate solution. The benefits of this generic approach also include flexibility resulting from highly parameterized objective functions: the configuration of the scheduler can be delegated to the cost optimization process. This is enabled by the ASO framework that readily supports both integer-weighted constraints and objective functions.

In summary, the current article advances logic-based approaches to integrated maintenance scheduling by developing its underlying theory and knowledge presentation models. This research is interdisciplinary and promotes workforce and competence management in product-service applications such as maintenance optimization. It puts forth formal abstraction and knowledge representation related to these interests but does not compare formalisms, encodings, nor algorithms. Therefore, the formalism (ASO), existing encodings of PMS [35,36], and efficient solvers are merely prerequisites rather than contributions of this work.

In this work, we apply logic-based modeling and optimization in the context of scheduling and resource allocation. Due to logical representations, the scalability of our approach is expected to improve along the development of more efficient solver technology. Moreover, we anticipate the applicability of the presented resource models and ideas related to workforce allocation in other

contexts such as multi-agent systems, Markov decision processes, and (deep) reinforcement learning—already combined with logical representations [4,5,14].

Acknowledgment. This research has been supported by the Research Council of Finland under projects AI-ROT (#335718,#335719) and XAILOG (#345633).

References

1. Banbara, M., et al.: *teaspoon*: solving the curriculum-based course timetabling problems with answer set programming. Ann. Oper. Res. **275**, 3–37 (2019). https://doi.org/10.1007/s10479-018-2757-7
2. Benner, P.: From novice to expert. Am. J. Nurs. **82**(3), 402–407 (1982). https://doi.org/10.1002/nur.4770080119
3. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011). https://doi.org/10.1145/2043174.2043195
4. Camacho, A., Icarte, R.T., Klassen, T.Q., Valenzano, R.A., McIlraith, S.A.: LTL and beyond: formal languages for reward function specification in reinforcement learning. In: IJCAI 2019, pp. 6065–6073. ijcai.org (2019). https://doi.org/10.24963/IJCAI.2019/840
5. Cao, Y., et al.: GALOIS: boosting deep reinforcement learning via generalizable logic synthesis. In: NeurIPS 2022, pp. 19930–19943 (2022)
6. Chansombat, S., Pongcharoen, P., Hicks, C.: A mixed-integer linear programming model for integrated production and preventive maintenance scheduling in the capital goods industry. Int. J. Prod. Res. **57**(1), 61–82 (2019). https://doi.org/10.1080/00207543.2018.1459923
7. Chen, J., Wang, Y.: A deep reinforcement learning approach for maintenance planning of multi-component systems with complex structure. Neural Comput. Appl. **35**(21), 15549–15562 (2023). https://doi.org/10.1007/s00521-023-08542-9
8. Chen, X., An, Y., Zhang, Z., Li, Y.: An approximate nondominated sorting genetic algorithm to integrate optimization of production scheduling and accurate maintenance based on reliability intervals. J. Manuf. Syst. **54**, 227–241 (2020). https://doi.org/10.1016/j.jmsy.2019.12.004
9. Do, P., Vu, H.C., Innovation, A., Berenguer, C.: Maintenance grouping for multi-component systems with availability constraints and limited maintenance teams. Reliab. Eng. Syst. Saf. **142**, 56–67 (2015). https://doi.org/10.1016/j.ress.2015.04.022
10. Dodaro, C., Maratea, M.: Nurse scheduling via answer set programming. In: Balduccini, M., Janhunen, T. (eds.) LPNMR 2017. LNCS, vol. 10377, pp. 301–307. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61660-5_27
11. Dreyfus, H.L., Dreyfus, S.E.: Mind over Machine: The Power of Human Intuition and Expertise in the Age of the Computer. Basil Blackwell, Oxford (1986)
12. Eiter, T., Geibinger, T., Musliu, N., Oetsch, J., Skocovský, P., Stepanova, D.: Answer-set programming for lexicographical makespan optimisation in parallel machine scheduling. In: KR 2021, pp. 280–290 (2021). https://doi.org/10.24963/kr.2021/27
13. El Akraoui, B., Daoui, C., Larach, A., Rahhali, K.: Decomposition methods for solving finite-horizon large MDPs. J. Math. 1–8 (2022). https://doi.org/10.1155/2022/8404716

14. Elsayed-Aly, I., Feng, L.: Logic-based reward shaping for multi-agent reinforcement learning. CoRR abs/2206.08881 (2022). https://doi.org/10.48550/ARXIV.2206.08881
15. Falkner, A.A., Friedrich, G., Schekotihin, K., Taupe, R., Teppan, E.C.: Industrial applications of answer set programming. Künstliche Intelligentz **32**(2–3), 165–176 (2018). https://doi.org/10.1007/S13218-018-0548-6
16. Geurtsen, M., Didden, J., Adan, J., Atan, Z., Adan, I.: Production, maintenance and resource scheduling: a review. Eur. J. Oper. Res. **305**(2), 501–529 (2023). https://doi.org/10.1016/j.ejor.2022.03.045
17. Hannola, L., Richter, A., Richter, S., Stocker, A.: Empowering production workers with digitally facilitated knowledge processes conceptual framework. Int. J. Prod. Res. **56**(14), 4729–4743 (2018). https://doi.org/10.1080/00207543.2018.1445877
18. Ikävalko, H., Saarelma, E., Martinsuo, M.: Innovation users profiles in implementing a digital innovation in maintenance. In: ISPIM Connects Salzburg The Sound of Innovation. LUT Scientific and Expertise Publications (2023)
19. Le Deist, F.D., Winterton, J.: What is competence? Hum. Resour. Dev. Int. **8**(1), 27–46 (2005). https://doi.org/10.1080/1367886042000338227
20. Lerch, C., Gotsch, M.: Digitalized product-service systems in manufacturing firms: a case study analysis. Res. Technol. Manag. **58**(5), 45–52 (2015). https://doi.org/10.5437/08956308X5805357
21. Luukkala, V., Niemelä, I.: Enhancing a smart space with answer set programming. In: Dean, M., Hall, J., Rotolo, A., Tabet, S. (eds.) RuleML 2010. LNCS, vol. 6403, pp. 89–103. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16289-3_9
22. Marcon, É., et al.: Capabilities supporting digital servitization: a multi-actor perspective. Ind. Mark. Manag. **103**, 97–116 (2022). https://doi.org/10.1016/j.indmarman.2022.03.003
23. Martinsuo, M., Ikävalko, H.: Innovation users' view to implementing digital innovations in industrial operations. In: IPDMC 2023. European Institute for Advanced Studies in Management (2023)
24. Paukkunen, J.: Enhancing the transfer of tacit knowledge in maintenance: strategies and implications. Master's thesis, Aalto University School of Business, Information & Service Management (2023)
25. Pinciroli, L., Baraldi, P., Zio, E.: Maintenance optimization in industry 4.0. Reliab. Eng. Syst. Saf. **234**, 109204 (2023). https://doi.org/10.1016/j.ress.2023.109204
26. Rajaprasad, S.V.S.: Investigation of reliability, maintainability and availability of a paper machine in an integrated pulp and paper mill. Int. J. Eng. Sci. Technol. **10**(3), 43–56 (2018). https://doi.org/10.4314/ijest.v10i3.5
27. Raza, S., Hameed, A.: Models for maintenance planning and scheduling a citation-based literature review and content analysis. J. Qual. Maint. Eng. **28**(4), 873–914 (2022). https://doi.org/10.1108/JQME-10-2020-0109
28. Rossi, F., van Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming, Foundations of Artificial Intelligence, vol. 2. Elsevier (2006)
29. Selcuk, S.: Predictive maintenance, its implementation and latest trends. J. Eng. Manuf. **231**(9), 1670–1679 (2017). https://doi.org/10.1177/0954405415601640
30. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. **138**(1–2), 181–234 (2002). https://doi.org/10.1016/S0004-3702(02)00187-X
31. Story, V., Raddats, C., Burton, J., Zolkiewski, J., Baines, T.: Capabilities for advanced services: a multi-actor perspective. Ind. Mark. Manag. **60**, 54–68 (2017). https://doi.org/10.1016/j.indmarman.2016.04.015

32. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. A Bradford Book. MIT Press, Cambridge (2018)
33. Tassel, P., Gebser, M., Schekotihin, K.: An end-to-end reinforcement learning approach for job-shop scheduling problems based on constraint programming. In: ICAPS 2023, pp. 614–622. AAAI Press (2023). https://doi.org/10.1609/ICAPS.V33I1.27243
34. Wolsey, L.A.: Mixed Integer Programming. Wiley, Hoboken (2008). https://doi.org/10.1002/9780470050118.ecse244
35. Yli-Jyrä, A., Janhunen, T.: Applying answer set optimization to preventive maintenance scheduling for rotating machinery. In: Governatori, G., Turhan, A.Y. (eds.) RuleML+RR 2022. LNCS, vol. 13752, pp. 3–19. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-21541-4_1
36. Yli-Jyrä, A., Rankooh, M.F., Janhunen, T.: Pruning redundancy in answer set optimization applied to preventive maintenance scheduling. In: Hanus, M., Inclezan, D. (eds.) PADL 2023. LNCS, vol. 13880, pp. 279–294. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-24841-2_18
37. Zhang, X., Zeng, J.: Joint optimization of condition-based opportunistic maintenance and spare parts provisioning policy in multiunit systems. Eur. J. Oper. Res. **262**(2), 479–498 (2017). https://doi.org/10.1016/j.ejor.2017.03.019