Alp Darendeliler[a]*, Dieter Claeys[a,b], and El-Houssaine Aghezzaf[a,b]

[a]Department of Industrial Systems Engineering and Product Design, Ghent University, Ghent, Belgium; [b]Industrial Systems Engineering (ISyE), Flanders Make, Kortrijk, Belgium

*corresponding author: Alp Darendeliler; alp.darendeliler@ugent.be

Dieter Claeys (ORCID ID): https://orcid.org/0000-0002-7666-2479

El-Houssaine Aghezzaf (ORCID ID): https://orcid.org/0000-0003-3849-2218

Integrated condition-based maintenance and multi-item lot-sizing with stochastic demand

Abstract: This paper studies the problem of integrated lot-sizing and maintenance decision making in case of multiple products and stochastic demand. The problem is formulated as a Markov Decision Process (MDP), in which the goal is to find a joint production and maintenance policy that minimizes the long run expected total discounted cost. Therefore, the classic Q-learning algorithm is adopted, and a decomposition-based approximate Q-value heuristic is developed to obtain near-optimal solutions in a reasonable time. To accelerate the convergence of the Q-learning algorithm, a hybrid Q-learning method is proposed in which the Q-values are initiated by the output of the decomposition-based approximate Q-value heuristic. The numerical experiments reveal that the approximate Q-value heuristic is outperformed by the classic and hybrid Q-learning algorithms in terms of accuracy and that the hybrid Q-learning method converges much faster than the classic Q-learning method. However, these so-called tabular methods do not scale to larger problems with more than four products. Hence, based on the problem structure, three state aggregation schemes are developed and applied to the Q-learning algorithm to solve the large-scale problems. The numerical study demonstrates that Q-learning with the third state aggregation scheme performs nearly as good as the hybrid Q-learning method while significantly reducing the computational time and being scalable to large-scale problems.

Keywords: Condition-based maintenance; Markov decision process; inventory; multi-product; lot sizing; stochastic demand; reinforcement learning

1. Introduction

In many manufacturing systems, preventive maintenance is proactively performed in order to reduce the likelihood of unplanned breakdowns, increase production uptime. In particular, Condition-based maintenance (CBM) recommends performing maintenance based on the real-time information collected through condition monitoring. If a CBM program is properly established and effectively implemented, it can significantly reduce maintenance costs by reducing the number of unnecessary scheduled preventive maintenance operations (Jardine et al., 2006).

In developing a preventive maintenance policy, the stock levels of the products should be taken into account in addition to the equipment condition to avoid unmet demand when the system is under maintenance. In addition, production decisions influence the degradation behaviour of manufacturing systems. Thus, equipment condition also needs to be considered when planning production. Otherwise, breakdowns may interrupt the production, leading thus to production losses. To address this interdependency, in the literature, joint production lot-sizing and CBM policies have been proposed with the aim of minimizing total production and maintenance costs. Most of these studies assume that a single product type is produced, and that the demand is deterministic, which is usually not the case in practice. Hence, a static policy that optimizes the lot-size and maintenance threshold is developed. Nevertheless, due to the trend of shifting from mass production to mass customization, machines have to produce multiple products instead of only one. In such systems, dynamic policies are required to be responsive to the stochasticity of the degradation and the product demands.

This study considers a single-machine, multiple products, periodic-review production/inventory system with stochastic demand, implementing CBM. The machine stochastically degrades with usage, and the degradation behaviour differs depending on the item being produced. Based on the system state, consisting of the product inventories and degradation level, the production and maintenance decisions are dynamically made. The problem is formulated as an MDP, in which upon observing the system state at the beginning of each period, if the system is operational, it is decided whether to produce a particular product, perform preventive maintenance, or keep the machine idle. When the equipment has failed at an observation epoch, corrective maintenance is performed.

To solve the underlying MDP, the Q-learning algorithm is adopted. Q-learning estimates the state-action values via simulated samples instead of using complete transition probabilities. Hence, it can solve relatively large problems that cannot be solved by dynamic programming methods. Moreover, an approximate Q-value heuristic is proposed to quickly obtain a suboptimal solution. To speed up the learning process, the state-action values in the Q-learning algorithm are initialized by those obtained by the approximate Q-value heuristic method. However, the tabular Q-learning and the heuristic method are also subject to the curse of dimensionality due to state space explosion. In the numerical experiments (Section 5), the dynamic programming methods become intractable for problems with more than two products, while the Q-learning and the heuristic methods can solve three and four-product problems, but not more than four products. To be able to solve larger problems in a reasonable time, three state aggregation schemes are proposed that exploit the problem structure and apply Q-learning on the aggregated state space. In Section 5, it is observed that one of the proposed Q-learning with state aggregation methods is accurate while still being tractable for more than four products.

The remainder of the paper is organised as follows. Section 2 provides the related literature. Then, in Section 3, the problem is formulated as a Markov decision process (MDP). Section 4 describes the methods to solve the MDP. The performance of these methods is evaluated in Section 5, and conclusions and directions for future work are mentioned in Section 6.

2. Literature review

The integration of production lot-sizing and maintenance has been extensively studied by many researchers. Ben-Daya and Makhdoum (1998) study the effect of various preventive maintenance policies on the joint optimization of EPQ and the economic design of the control chart. Ben-Daya (2002) and El-Ferik (2008) propose models to determine optimal EPQ and age-based maintenance policies for production systems under imperfect preventive maintenance. Liao and Sheu (2011) develop an EPQ model that considers perfect and imperfect preventive maintenance. The probability that a preventive maintenance operation is perfect depends on the number of imperfect maintenance operations carried out since the last renewal cycle. Aghezzaf et al. (2007) and Shamsaei and Van Vyve (2017) propose models for the integrated multi-item production and maintenance planning problem. However, the above studies employ traditional time-based maintenance (TBM) approaches; thus, they do not take the equipment degradation status into account in determining production and maintenance policies.

In recent years, CBM is incorporated into the EPQ problem. Jafari and Makis (2015) propose a model for the joint optimization of EPQ and preventive maintenance policy, where the deterioration of the system is described by a proportional hazards model to consider the age and condition monitoring (CM) information. Zheng et al. (2021) extend the work of Jafari and Makis (2015) by considering a CBM policy with multiple maintenance actions and dynamic control limits in their model. Peng and van Houtum (2016) develop a model to optimize EPQ and CBM policy, where the degradation follows continuous time and continuous state stochastic process. Khatab et al. (2018) investigate the integration of production quality and CBM and propose a model to determine the optimal inspection cycle and degradation threshold, which initiates preventive maintenance. Cheng et al. (2018) address an integrated problem of production lot-sizing, quality control and CBM. In all these models, it is assumed that a single product is produced, and the demand rate is constant.

For the single-product systems with random demand, Markov decision process (MDP) models have been developed to optimize the joint production and CBM policies

(Iravani and Duenyas, 2002; Sloan, 2004; Xiang et al., 2014; Jafari and Makis, 2019). In these studies, the degradation is modelled as a Markov chain with limited number of states.

Table 1. Summary of literature on the integration of lot-sizing and maintenance

Authors	Maintenance	Single or multiple items	Demand
Aghezzaf et al. (2007)	TBM	Multiple	Deterministic and dynamic
Ben-Daya (2002)	TBM	Single	Deterministic and constant
Ben-Daya and Makhdoum (1998)	TBM	Single	Deterministic and constant
Cheng et al. (2018)	CBM	Single	Deterministic and constant
El-Ferik (2008)	TBM	Single	Deterministic and constant
Iravani and Duenyas (2002)	CBM	Single	Stochastic and stationary
Jafari and Makis (2015)	CBM	Single	Deterministic and constant
Jafari and Makis (2019)	CBM	Single	Stochastic and stationary
Khatab et al. (2019)	CBM	Single	Deterministic and constant
Liao and Sheu (2011)	TBM	Single	Deterministic and constant
Peng and van Houtum (2016)	CBM	Single	Deterministic and constant
Shamsaei and Van Vyve (2017)	TBM	Multiple	Deterministic and dynamic
Sloan (2004)	CBM	Single	Stochastic and stationary
Xiang (2014)	CBM	Single	Stochastic and stationary
Zheng et al. (2021)	CBM	Single	Deterministic and constant
Proposed model	CBM	Multiple	Stochastic and stationary

The literature on the joint optimization of production lot-sizing and maintenance, summarized in Table 1, suggests that multiple products have only been considered in a non-CBM maintenance setting with deterministic demand, and in particular that the integration of CBM and lot sizing in a stochastic demand setting has been considered only for single-product systems. Hence, to the best of our knowledge, the integrated lot-sizing and CBM decision making has not been studied for multi-item production systems with stochastic demand. This paper aims at developing an integrated production and maintenance model that fills this gap.

The common methodology to deal with the stochastic demand is to model the problem as an MDP which is solved to optimality by classical dynamic programming methods as in single-product models with tractable state spaces (Iravani and Duenyas, 2002; Sloan, 2004). In this paper, the problem is extended to multiple products and formulated as an MDP, but the main challenge is solving the MDP due to the exponential increase of the state space (and thus computational time and space) in the number of products. Hence, reinforcement learning (also called approximate dynamic programming) techniques are adopted to tackle this challenge.

Scheduling the production of multiple products on a single machine under stochastic demand over an infinite horizon–but without considering maintenance decision making–is in literature categorized as the stochastic economic lot-scheduling problem (SELSP) (Sox et al. 1999). Winands et al. (2011) classify the SELSPs based on their sequencing and the lot-sizing decisions. The proposed model's production policy could be classified under the category of global lot-sizing (lot-sizing decisions depend not only on the stock level of the current product being produced but also on the complete system state) and dynamic production sequencing (decisions are made based on current state rather than in a fixed order) within the framework of the SELSP literature. However, the SELSP literature does not consider deterioration of the underlying systems and maintenance activities, while this paper considers them.

There are few studies in the literature that consider SELSP under the category of global lot-sizing and dynamic production sequence. Qiu and Loulou (1995) propose a semi-Markov decision process model (SMDP) and obtain near-optimal policies by the successive approximation algorithm for two-product problems. They mention that their procedure is not efficient and accurate for larger problem instances due to the curse of dimensionality. Furthermore, Li et al. (2023) investigate the joint multi-item capacitated line balancing and lot-sizing problem with random demand. The problem is formulated as a risk-averse two stage stochastic programming model.

The main reason for the application of reinforcement learning or approximate dynamic programming to the SELSP is tackling the curse of dimensionality problem. Wang et al. (2012) propose two reinforcement learning algorithms to solve SELSP with random demand and processing times. Since the proposed algorithms are tabular, they could solve only limited-sized problems. Löhndorf and Minner (2013) formulate the SELSP as a SMDP. To deal with the curse of dimensionality, they propose two linear approximate value functions and apply approximate value iteration to find the weights of them. For small problems, the approximate value iteration performs better than globally optimized simple base-stock and fixed-cycle policies, but as the problem size increases, it is significantly outperformed. This study proposes dynamic sequencing, lot-sizing and maintenance policies based on the degradation state and the inventory levels. Hence, unlike in the SELSP, the sequencing and lot-sizing decisions are influenced by the equipment degradation. Moreover, the numeric experiments reveal that Q-learning with the proposed state aggregation scheme provides well-performing policies while being scalable to large-scale problems.

Deep reinforcement learning (DRL) is also recently applied to complex maintenance and production/inventory control problems. Huang et al. (2020) propose a DRL approach to find a preventive maintenance policy for a complex serial production line with intermediate buffers. Gijsbrechts et al. (2021) apply Asynchronous Actor Critic (A3C) DRL algorithm to the lost sales, dual sourcing and multi-echelon inventory problems. They conclude that the performance of A3C can match the performance of the state-of-the-art heuristics and other approximate dynamic programming methods. Vanvuchelen et al. (2020) utilize proximal policy optimization algorithm to the joint replenishment problem. The algorithm approaches the optimal policy for small-scale problems and gives comparable results with the well-known heuristic. Oroojlooyjadid et al. (2021) propose a DRL algorithm for the beer game and obtain near optimal policies. In the above studies, the DRL algorithms give plausible results. However, a challenge in DRL algorithms is the significant time and effort required for hyperparameter tuning (Boute, Gijsbrechts, Jaarsveld and Vanvuchelen, 2021). On the contrary, the proposed Q-learning method with state aggregation (QLA) is practical to implement and does not require significant effort to tune the hyperparameters. The numerical study shows that it converges to a well-performing policy in a reasonable computational time.

Nomenclature

QL	Q-learning
IQL	hybrid Q-learning
QLA1	Q-learning with the first state aggregation scheme
QLA2	Q-learning with the second state aggregation scheme
QLA3	Q-learning with the third state aggregation scheme
AVC	average cost per period
M	set of products
X_n	degradation level at the beginning of period n
I_n^m	inventory level of product m at the beginning of period n
I_{max}^m	maximum allowed stock level of product m
q^m	planned lot-size of product m
C	time length of a period
ρ_m	production rate of product m
s_n	system state at the beginning of period n
$\bar{s}_n$	aggregated state at the beginning of period n
D_n^m	random variable denoting the demand of product m in period n
d_n^m	sampled demand of product m in period n
λ^m	mean demand of product m
F	failure state
$P_{ij}(m)$	one-step transition probability of degradation from state i to j when producing item m
$\boldsymbol{P(m)}$	probability transition matrix under the production of item m
$T_F(m)^{(i)}$	first passage time to F from state i for product m
c_s^m	setup cost for product m
c_m	unit production cost of product m
c_h^m	inventory holding cost of product m
c_l^m	lost sales cost of product m
c_c	corrective maintenance cost
c_p	preventive maintenance cost
Y_n^m	binary variable that equals 1 if and only if product m is produced in period n
$C(s,a)$	one-period cost in state s under action a
$V(s)$	value of being in state s
$Q(s,a)$	state action value
μ	production and maintenance policy
γ	discount factor
u	index of the most urgent product
$r^{u'}$	the second most urgent product's runout time
$J^i(s)$	state aggregation function of the i^{th} aggregation scheme
$\bar{\pi}(s)$	estimated steady state probability of being in state s
d_{opt}	percentage error with respect to the optimal policy
d_r	percentage of the relative difference of the value functions

3. Problem formulation

Consider a manufacturing system producing multiple items to meet the uncertain demand. All items are produced by a single machine, which deteriorates while it is producing. The time axis is divided in periods, and at the beginning of each period, inventory levels of the products and the degradation level of the equipment are observed. The observed state at the start of period n is denoted by $s_n = (X_n, I_n^1, \dots, I_n^{|M|})$, where X_n designates the degradation level, I_n^m the inventory level of product m, and M the set of products. Based on the observed state, a decision is made to take one of the following actions during the period: (1) stay idle; (2) perform corrective maintenance (only an option and the only option if a failure occurred in the previous period); (3) perform preventive maintenance; (4) produce a particular product m in a certain quantity q^m. In each period, only one product type can be produced; if maintenance is to be performed, then production cannot take place in that period. It is assumed that the equipment becomes "as good as new" after preventive or corrective maintenance. The equipment is used at full capacity within a period in which production occurs. This means that the equipment produces $q^m = C\rho_m$ units of product m during a period, with C the period length and ρ_m the production rate of item m (units per unit time).

A fixed production setup cost c_s^m is incurred when item m is produced in a period, and a cost of c_m occurs per unit of item m produced. Hence, when q_n^m products m are produced in period n, the total production cost is $c_s^m + q_n^m c_m$. If a failure occurs during production in a period, then a production cost is incurred for the produced items up to the failure. If a failure occurs after k units of product m have been produced, then the total production cost is $c_s^m + kc_m$. An inventory holding cost c_h^m is charged per item m in the stock at the end of a period, that is after the product demand is realized. The items produced in a period can already be used to satisfy the demand in that period. If the demand cannot be satisfied in a period, then lost sales cost c_l^m is incurred per unit of shortage. Inventory levels of the products are always equal to or greater than zero since backlogging is not allowed. In addition, they must be less than or equal to the maximum allowed inventory level I_{max}^m, where $m \in M$. As a consequence, producing product m in period n is only an allowed action if $I_n^m + q^m \leq I_{max}^m$. The preventive maintenance cost is denoted by c_p and the cost for corrective maintenance by c_c. The preventive maintenance cost is assumed to be much less than the corrective maintenance cost. The demand for product m during period n is denoted by the random variable D_n^m. It is assumed that the demand for a product during consecutive periods forms a sequence of independent and identically distributed random variables. The demand distributions of the products are also independent of each other. The problem is modelled as an infinite-horizon MDP with a discount factor γ. The objective is to minimize the total expected discounted costs over the long run.

The evolution of the degradation is modelled as a discrete-time stochastic process. If product m is produced within a period, then the k^{th} epoch corresponds to the planned completion epoch of the k^{th} unit of item m. As a result, the time in between two production epochs within the same period equals $1/\rho_m$ if no failure occurs in between of those epochs. The degradation level of the equipment at epoch k is denoted by Z_k. Within a period n, the process $\{Z_k, k = 0,1, ...\}$, behaves as an absorbing Markov chain with $Z_o = X_n$ and with state space $\{X_n, X_n + 1, ..., F\}$, where F denotes the absorbing (failure) state. The transition probabilities of degradation level transitioning to state j at the next epoch if the degradation level is equal to i at the current epoch are denoted by $P_{ij}(m)$:

$$P_{ij}(m) = P\{Z_{k+1} = j | Z_k = i, Y_n^m = 1\} \quad for\ i \leq j \leq F, m = 1, ..., |M|, \tag{1}$$

with Y_n^m being a binary decision variable that equals 1 if and only if product m is produced during period n. As degradation can only increase during production, $P_{ij}(m) = 0$ if $j < i$. $\mathbf{P(m)}$ denotes the matrix of one-step transition probabilities $P_{ij}(m)$ when producing product m. The k-step transition probability of the Markov chain under the production of item m transitioning from state i to j corresponds to the probability that the degradation is at level j right after the production of the k^{th} item within the same period. It is given by

$$P_{ij}(m)^k = P\{Z_k = j | Z_0 = i, Y_n^m = 1\} \quad for\ i \leq j \leq F, m = 1, \ldots, |M|. \tag{2}$$

Since F is the absorbing state of the Markov chain,

$$P_{FF}(m)^k = P\{Z_k = F | Z_0 = F, Y_n^m = 1\} = 1 \quad \forall\ k \in \{1,2,\ldots\}, m = 1, \ldots, |M|. \tag{3}$$

$P_{ij}(m)^k$ is equal to the entry at the i^{th} row and j^{th} column of the $k - step$ transition probability matrix $\mathbf{P(m)^k}$ for product m. If product m is chosen to be produced in period n, and the degradation level at the beginning of the period is i, then $P_{ij}(m)^{q^m}$ is the probability that state $j \leq F$ will be observed at the end of the production run, and thus at the beginning of the next period. If a preventive maintenance is carried out in a period, then the next period's degradation level is 1, that is the "as good as new" state.

It is assumed that failures occur at the epochs when the production of a unit of an item has just completed. If a failure occurs right after the production of the k^{th} unit, production is stopped at epoch k, and the degradation level of the next state is F. The first passage time $T_F(m)^{(i)}$ from state i to the failure state F for product m, if no preventive maintenance would be carried out has the phase-type distribution $Ph(e_i,\boldsymbol{T(m)})$, that is

$$P\left\{T_F(m)^{(i)} = k\right\} = e_i.\boldsymbol{T(m)^{k-1}}.\boldsymbol{t(m)}, \tag{4}$$

$$P\left\{T_F(m)^{(i)} \le k\right\} = 1 - e_i.\boldsymbol{T}(\boldsymbol{m})^k.\boldsymbol{1}, \tag{5}$$

where $\boldsymbol{T}(\boldsymbol{m})$ is the probability transition matrix of the transient states of the one-step probability transition matrix $\boldsymbol{P}(\boldsymbol{m})$ for product m (first F rows and columns of $\boldsymbol{P}(\boldsymbol{m})$) , $\boldsymbol{t}(\boldsymbol{m})$ is the column vector showing the probabilities from each state $i < F$ to the failure state F (first F rows of the last column of $\boldsymbol{P}$) and e_i is the i^{th} unit vector.

At the beginning of a period n, the system is in state $s_n = \left(X_n, I_n^1, \dots, I_n^{|M|}\right)$. Given the degradation level X_n and the inventory levels $I_n^1, \dots, I_n^{|M|}$, the next period's state s_{n+1} and the one period cost $C(s_n, a)$, depend on the selected action a, the realized product demands $d_n^1, \dots, d_n^{|M|}$, and the evolution of the degradation if a product is being produced. This behaviour is formalized in the following system equations:

1. If $X_n = i < F$ and action a corresponds to the preventive maintenance decision, or $X_n = F$ (corrective maintenance is only option), then

$$s_{n+1} = \left(X_{n+1} = 1, I_{n+1}^1 = (I_n^1 - d_n^1)^+, \dots, I_{n+1}^{|M|} = \left(I_n^{|M|} - d_n^{|M|}\right)^+\right),$$

$$C(s_n, a) = \sum_{u=1}^{|M|} c_h^u (I_n^u - d_n^u)^+ + \sum_{u=1}^{|M|} c_l^u (d_n^u - I_n^u)^+ + \begin{cases} c_p & if \;\; X_n < F \\ c_c & if \;\; X_n = F \end{cases}.$$

2. If $X_n = i < F$ and action a corresponds to the "$stay\ idle$" decision, then

$$s_{n+1} = \left(X_{n+1} = i, I_{n+1}^1 = (I_n^1 - d_n^1)^+, \dots, I_{n+1}^{|M|} = \left(I_n^{|M|} - d_n^{|M|}\right)^+\right),$$

$$C(s_n, a) = \sum_{u=1}^{|M|} c_h^u (I_n^u - d_n^u)^+ + \sum_{u=1}^{|M|} c_l^u (d_n^u - I_n^u)^+.$$

3. If $X_n = i < F$ and action a corresponds to the production of item m with $q^m = C\rho_m$ (producing item m is only a valid option when $q^m = C\rho_m \le I_{max}^m - I_n^m$), then

$$s_{n+1} = (j, I_{n+1}^1 = (I_n^1 - d_n^1)^+, \dots, I_{n+1}^m = (I_n^m + k - d_n^m)^+, \dots, I_{n+1}^{|M|} = \left(I_n^{|M|} - d_n^{|M|}\right)^+\Big),$$

$$C(s_n, a) = \sum_{u \in M \setminus \{m\}} c_h^u (I_n^u - d_n^u)^+ + \sum_{u \in M \setminus \{m\}} c_l^u (d_n^u - I_n^u)^+ + c_s^m$$

$$+ c_m k + c_h^m (I_n^m + k - d_n^m)^+ + c_l^m (d_n^m - I_n^m - k)^+,$$

where $k = T_F(m)^{(i)}\ when\ j = F$, and $k = q^m$ when $j < F$.

The goal is to find an optimal policy μ that maps each state $s \in S$ to action $\mu(s) \in A(s)$ to minimize the total expected discounted costs over the long run:

$$V^*(s) = \min_{\mu} E^{\mu} \left[\sum_{n=0}^{\infty} \gamma^n \, C\big(s_n, \mu(s_n)\big) \right], \tag{6}$$

where $s = s_0$ is the initial state, $0 < \gamma < 1$ is the discount factor, and $V^*(s)$ denotes the optimal value function of state s. The optimal policy can be found by solving the well-known Bellman optimality equations:

$$V^*(s) = \min_{a \in A(s)} \{EC(s, a) + \gamma E[V^*(s') | s, a]\} \qquad \forall s \in S, \tag{7}$$

where $EC(s, a)$ is the expected one-period cost of taking action a in state s. For the set of Bellman equations given in (7), there exists a unique optimal solution $V^*(s)$, for all states $s \in S$ (The detailed version of (7) for the present problem is given in Appendix D). Given the optimal value functions, any policy that minimizes the right-hand side of the (7) for all $s \in S$, is an optimal policy for the MDP. The dynamic programming methods exactly solve (7) by using the Bellman equation as an updating rule.

4. Solution methods

Dynamic programming (DP) methods such as value iteration or policy iteration can be used to solve MDPs with finite state and action spaces. However, due to the curse of dimensionality, they might require an exponential amount of computational time/space for large and even for moderate-size problem instances. The present problem has a state space size of $F\prod_{k=1}^{|M|}(I_{max}^{k}+1)$ for $|M|$ products which increases exponentially in the number of products. DP methods involve frequent expected updating of the value functions in all states. Hence, for more than two products, the problem becomes quickly computationally intractable using DP methods. Therefore, in this section, three alternative techniques are proposed which will allow to find approximate solutions within a reasonable time for problems with more than two products. In Section 5, the performance of these techniques will be evaluated.

4.1. The decomposition-based approximate Q-value heuristic

The main idea of the heuristic method is to decompose the n-product problem into n single-product problems which are computationally tractable. The decomposition approach is explained in Bertsekas (2019) and illustrated on the example of the restless multiarmed bandit problem. In the present problem, the optimal state-action values (Q-values) are approximated by combinations of the optimal state-action values of the single-product problems, called subproblems.

The approximate Q-value of the main problem is denoted by $\bar{Q}(s,a)$ for state $s=(X=i,I^1,\dots,I^{|M|})$ and action $a\in A=\{stay\ idle\ ,\ produce\ product\ 1,\dots,$ $produce\ product\ |M|\ ,\ do\ preventive\ maintenance,$ $do\ corrective\ maintenance\}$. In the subproblem m, only item m is taken into account and other products are ignored. The optimal Q-values $Q_m(s_m,a_m)$ and optimal value functions $V_m(s_m)$ for each subproblem $m\in M$, can be computed by the value-iteration algorithm where $s_m=(X=i,I^m)$ and $a_m\in A_m=\{stay\ idle\ ,\ produce\ product\ m,$ $do\ preventive\ maintenance,\ do\ corrective\ maintenance\}$. For the subproblems, the detailed Bellman optimality equation is given in Appendix E.

The main problem's approximate Q-values are expressed as follows:

1. If action $a = "stay\ idle"$ is taken, then

$$\bar{Q}\left(i, I^1, \dots, I^{|M|}, a\right) = \sum_{k=1}^{|M|} Q_k(i, I^k, a_k), \tag{8}$$

where $a_k = "stay\ idle",\ \forall\ k \in M.$

2. If action $a = "produce\ product\ m"$ is taken, then,

$$\bar{Q}\left(i, I^1, \dots, I^{|M|}, a\right) = \sum_{k \in M \setminus \{m\}} Q_k(i, I^k, a_k) + Q_m(i, I^m, a_m) \tag{9}$$

where $a_m = "produce"$ and $a_k = "stay\ idle",\ \forall\ k \in M \setminus \{m\}$.

3. If $X = i < F$ and action $a = "do\ preventive\ maintenance"$ is taken, or $i = F$ ($a = "do\ corrective\ maintenance"$ is the only valid option), then

$$\bar{Q}\left(i, I^1, \dots, I^{|M|}, a\right) = \begin{cases} c_p & if\ i < F \\ c_c & if\ \ i = F \end{cases} + \sum_{k=1}^{|M|} Q_k(1, I^k, a_k) \tag{10}$$

where $a_k = "stay\ idle",\ \forall\ k \in M.$

The resulting suboptimal policy is defined by $\bar{\mu}(s) = \underset{a \in A(s)}{argmin}\, \bar{Q}(s, a)\ \ \forall s \in S.$

4.2. Tabular Q-learning

Q-learning is one of the most widely used reinforcement learning methods and is shown in pseudo-code in Figure 1. Instead of updating value functions, Q-learning updates state-action values (Q-values) giving the value of taking an action a in a state s and from then on following the best policy learned so far. Thus, the algorithm iteratively solves the Bellman optimality equation

$$Q^*(s, a) = EC(s, a) + \gamma E\left[\min_{a' \in A(s')} Q^*(s', a') | s, a\right]\ \forall s \in S\,, a \in A(s),$$

where $Q^*(s, a)$ is the optimal state-action value. Instead of performing expected updates as in the DP methods, Q-learning estimates the optimal Q-values by the sampled values.

In the first step of the Q-learning algorithm, the Q-values are initialized. As these are unknown, the common approach is adopted which sets them to zero (Powell, 2011, Chapter 15). To speed up the learning process, the Q-learning algorithm is studied in case the output of the heuristic method is used to initiate the Q-values. It will be examined whether this approach leads to a good approximation in fewer iterations and whether the additional time to first run the heuristic algorithm pays off. In the sequel, the Q-learning method where the Q-values are initiated to 0 is denoted as "QL", and where the Q-values are initiated by the heuristic is denoted as "IQL". The detailed pseudocodes of the QL and the IQL are shown in Figure A1 and Figure A2 (Appendix A). It is proved that Q-learning converges to the optimal Q-values if each state-action pair is visited infinitely often and the step size satisfies certain conditions (Bertsekas and Tsitsiklis, 1996, Chapter 5). However, the convergence is slow when the discount factor is close to 1 (Evan-Dar and Mansour, 2003) as in the present case. Therefore, in the numerical examples, it will be examined whether IQL can expedite the convergence.

In the second step, an action $a \in A(s)$ is chosen for state $s = \left(X, I^1, \dots, I^{|M|}\right)$ in a way that balances exploitation (choosing what appears to be the best action with current experience) and exploration (learning more about effects of other actions). Then, the algorithm samples a realization of the degradation path based on the state s and the action chosen a; and the stationary product demands d^1,…,$d^{|M|}$. Using these values, the evolution of the one-period cost $C(s, a)$ and the next state s' are governed by the system equations given in section 3; thus, no expected update is required as in the DP methods. Based on this current and past experience, the Q-value is updated. Hence, state-action pairs are updated when they are visited. The visited Q-value $Q(s, a)$ at time t is updated based on the temporal difference between the target value $\left(C(s,a)+\gamma \min_{a'\in A(s')} Q(s',a')\right)$ minus old Q-value $Q(s,a)$, and α is the learning rate. The algorithm runs for T iterations which is a sufficiently large number.

---

1. Initialize a starting state s, and $Q(s,a)$ for all $s \in S$ and $a \in A(s)$

2. For $t = 1,\ 2, \dots, T$

    2.1. Choose action $a \in A(s)$ for state s $(\epsilon - greedy)$

    2.2. Sample the cost $C(s,a)$ and the next state s' based on the current state s and action a

    2.3. Update the Q-values by the equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(C(s,a) + \gamma \min_{a'\in A(s')} Q(s',a') - Q(s,a)\right)$$

    2.4. Update $s \leftarrow s'$

3. Return $\mu(s) = arg \min_{a\in A(s)} Q(s,a)\ \ \forall\, s \in S$

Figure 1. Steps of the Q-learning algorithm

4.3. Q-learning with state aggregation

The tabular Q-learning and the heuristic method store the Q-value of each state-action pair in memory. Therefore, as the state space grows exponentially with the number of products, they become inapplicable in case of a large number of products. To deal with this issue, a Q-learning algorithm with state aggregation is developed, and three aggregation schemes are proposed to reduce the state space. In essence, only Q-values for aggregate state-action pairs are computed and stored, and the disaggregated states are used in the transition process and the calculation of the one-period costs.

The results of the numerical experiment in Section 5 will show that the inventory level of the product with the lowest expected runout time (i.e. the expected duration until the inventory of an item falls to zero (Gascon et al., 1994)), has the biggest impact on the optimal Q-values among all items, that is, if the decision is to produce a product, that 'most urgent' product will be produced. Therefore, in the state aggregation methods, the index and inventory level of the most urgent product having the lowest expected runout time, is stored in the aggregated states while the other products' inventories are not kept as separate variables. If multiple products would have the same minimum expected runout time, then the item with the highest expected lost sales cost, in case of stock out, is selected as most urgent; if there would still be a tie, then one of these urgent products is randomly selected.

In the first aggregation scheme, the degradation level, the index and inventory level of the most urgent product, $u = \arg\min_{m \in M}(I^m/\lambda^m)$, are stored; where λ^m is the mean demand for product m during a period. For a complete state $s = \left(X, I^1, \dots, I^{|M|}\right)$, the corresponding aggregated state is given by $J^1(s) = (X, u, I^u)$, where $J^1(s)$ is the first aggregation function that maps the complete state s to the aggregated state (X, u, I^u).

To consider the impact of the products' inventories other than the most urgent on the Q-values, and thus the underlying policy, two alternative state aggregation methods are proposed. In the second state aggregation scheme, the second most urgent product's runout time, $r^{u'} = \left[\min_{m \in M \setminus \{u\}} I^m/\lambda^m\right]$, is stored in addition to u and I^u ($r^{u'}$is rounded to the nearest integer so as to be used as a discrete state). For a complete state s, the second aggregation function is given by $J^2(s) = \left(X, u, I^u, r^{u'}\right)$. The third aggregation approach stores the total inventory level in addition to u and I^u. Hence, in making production and maintenance decisions, the overall stock of the system is considered. Corresponding to the system state s, the aggregate state is given by $J^3(s) = \left(X, u, I^u, \sum_{m=1}^{|M|} I^m\right)$, where $J^3(s)$ is the third aggregation function.

At the beginning of a period n, after observing the aggregated state $\bar{s}_n$, one of the following actions is chosen: (1) keep the machine idle; (2) perform corrective maintenance (if and only if $X_n = F$); (3) perform preventive maintenance; (4) produce the urgent product u_n. Based on the system state s_n, the sampled demand values $d_n^1, \ldots, d_n^{|M|}$, and the selected action a, the one-period cost $C(s_n, a)$ and the next state s_{n+1} are determined by the system equations (Section 3). The steps of the Q-learning for the aggregation schemes $i = 1, 2, 3$ are shown in Figure 2. Note that $\bar{s}_t$ is the aggregated state associated with the complete system state s_t at iteration t.

The tabular representation of the state-action values has exponential worst-case state complexity $O(F|M|I_{max}^{|M|})$ while the first, second and third aggregation schemes have pseudo-polynomial complexities of $O(F|M|I_{max})$, $O(F|M|I_{max}^2)$, and $O(F|M|^2 I_{max}^2)$ respectively, where $I_{max} = \max_{k \in M} I_{max}^k$.

---

1. Choose aggregation scheme $i \in \{1,2,3\}$. Initialize
   $s_1 = \left(X_1, I_1^1, \ldots, I_1^{|M|}\right)$
   $\bar{s}_1 = J^i(s_1)$
   $Q(\bar{s}, a) = 0$ for all $\bar{s} \in \bar{S}^i$ and $a \in A(\bar{s})$
2. For $t = 1, 2, \ldots, T$
   2.1. Choose action $a_t \in A(\bar{s}_t)$ for state $\bar{s}_t$ ($\epsilon - greedy$)
   2.2. Sample $d^1, \ldots, d^{|M|}$ and X_{t+1},
   Compute $C(s_t, a_t)$, s_{t+1} by the system equations
   2.3. Update the next state
   $\bar{s}_{t+1} \leftarrow J^i(s_{t+1})$
   2.4. Update the Q-value by the equation:
   $$Q(\bar{s}_t, a_t) \leftarrow Q(\bar{s}_t, a_t) + \alpha \left(C(s_t, a_t) + \gamma \min_{a' \in A(\bar{s}_{t+1})} Q(\bar{s}_{t+1}, a') - Q(\bar{s}_t, a_t) \right)$$
3. Return $\mu(\bar{s}) = arg \min_{a \in A(\bar{s})} Q(\bar{s}, a) \quad \forall\, \bar{s} \in \bar{S}^i$

Figure 2. Steps of Q-learning with state aggregation

5. Numerical study

This section presents computational experiments on the performance of the proposed methods. Subsection 5.1 shows the performance evaluation of the tabular methods (the heuristic method (the decomposition-based Q-value heuristic), QL and IQL) on two, three and four-product examples. In Subsection 5.2, the Q-learning with state aggregation (QLA) methods are compared to the IQL, for the problem settings that are tractable to the

IQL, and the numeric result of a 10-product example is presented.

In all examples, machine degradation follows a Gamma process with shape and scale parameters α and β, and with the failure threshold L. By the method proposed by De Jonge (2019), the Gamma process is approximated by a discrete-time Markov chain having 21 states. The resulting one-step transition probability matrices can be found in the Supplementary Material of this article. A discount factor $\gamma = 0.9$ is used in the calculations. The cost parameters used in the experiments do not come from real cases. However, inspired by Wang et al. (2012) and Löhndrorf and Minner (2013), numerical experiments are carried out, and thus a wide set of scenarios are tested.

For the Q-learning algorithms (QL, IQL and QLA), a harmonic learning rate $\alpha = b_0 b/(b + N_t(s,a) - 1)$, is used with $b > 0$, $0 < b_0 \leq 1$ and $N_t(s,a)$ denoting the number of times a certain state-action pair (s,a) has been visited up to the time t (George and Powell, 2006). The parameters of the learning rate (b_0, b) are tuned separately for each method based on the guidelines for choosing step size formulas proposed by Powell (2011, Chapter 11).

5.1. Performance evaluation of the tabular methods

This part starts with the performance evaluations of the IQL, QL and heuristic method over 32 problem instances with two products which are generated by a 2^5 full factorial design. The problem instances are still computationally tractable, implying that the optimal policy can be found by the dynamic programming methods. As a result, the accuracy of the approximate policies obtained from Q-learning and the heuristic method can be evaluated by comparing them to the optimal policy. Since the Q-learning algorithms update the Q-values in each iteration and thus potentially update the policy, the author also examine how fast the Q-learning algorithms evolve to a near-optimal policy. In the problem instances with two products, the evolution of the percentage difference of the value functions between a proposed policy and the optimal policy is examined. This percentage difference d_{opt}, as proposed by Powell (2011, Chapter 15), is defined as

$$d_{opt} = 100 \sum_{s \in S} \bar{\pi}(s)\, |\bar{V}(s) - V(s)|/V(s)\,,$$

where

- $\bar{V}(s)$ is the value of being in state s under the approximate policy; $\bar{V}(s)$ is computed by the policy evaluation algorithm that takes the policy $\mu(s) = arg \min_{a \in A(s)} Q(s,a) \;\; \forall\, s \in S$, as input. The pseudocode of the algorithm is shown in Figure A4 in Appendix A.
- $V(s)$ is the optimal value function that is computed via classic dynamic programming algorithms such as the value iteration algorithm.
- $\bar{\pi}(s)$ is the estimated steady-state probability of being in state s, computed as: $\bar{\pi}(s) = N_t(s)/t$.
- In order to evaluate the heuristic algorithm for the two-product problem instances, d_{opt} is also computed, but there are two differences:
    - The heuristic algorithm approximates Q-values as a sum of Q-values of the corresponding one-product cases. Hence, the method is not iterative, thus d_{opt} only has to be calculated once.
    - The steady-state probabilities $\bar{\pi}(s)$ under the heuristic policy are estimated via simulation (in the examples, the system is simulated for $10{,}000{,}000$ iterations while the heuristic policy is being executed).

For the examples with more than two products (Section 5.2), the exact dynamic programming methods are not computationally tractable any more. Hence, different methods are needed to evaluate the performance of the Q-learning and heuristic algorithms in these cases. Two methods are employed. The first computes d_r every K iterations, which is similar to computing d_{opt} except that $V(s)$ is unknown and therefore replaced by the estimated value functions for the states up to the K iterations ago:

$$d_r = 100 \sum_{s \in S} \bar{\pi}(s) \left| V^u(s) - V^{(u-1)}(s) \right| / V^{(u-1)}(s) ,$$

where $V^{(u-1)}(s)$ and $V^u(s)$ are the estimated value functions for state $s \in S$ after respectively $(u-1)K$ and uK iterations, and $\bar{\pi}(s) = N_{uK}(s)/uK$ denotes the estimated steady-state probability at iteration uK of being in state s. In other words, d_r is a measure for the relative change in the value function during the last K iterations.

The second method calculates the average cost per period (AVC). Although the goal is to minimize the values, and not the AVC, a policy that aims at small value functions is likely to entail small AVC. The main advantage to consider AVC is computational: it can easily be calculated online (no need to go over all states) via the following update formula:

$$AVC \leftarrow (AVC\ (t-1) + C(s,a))/t$$

Another advantage of AVC is that it enables to compare the performance of the policies generated from the proposed methods even in the case where the optimal policy and value function are not known. The pseudocodes of QL and IQL shown in Figure A1 and Figure A2 (Appendix A) also include this update rule (step 2.4), and the formula to compute d_r (step 2.5). Note that AVC is updated each iteration, while d_r is updated every K iterations (K is chosen as 100,000 in the two and three-product examples and 1,000,000 in the four-product example). The AVC calculation steps for the heuristic method are shown in Figure A3 (Appendix A).

For IQL and QL, an epsilon-greedy exploration policy is used with decreasing exploration probability; for a visited state s at time t, a random action is selected with probability $\epsilon_t(s) = 1/(N_t(s)+1)$ to explore, and the greedy action $a^* = \operatorname*{argmin}_{a \in A} Q(s,a)$ is chosen with probability $\left(1-\epsilon_t(s)\right)$ and $N_t(s)$ is the number of times state s is visited up to the time t. The step size parameters for IQL and QL are given in Appendix F.

5.1.1. Two-product examples

In this part, the policies obtained from the IQL, QL and the heuristic methods are compared with the optimal policy. Tractable instances of two-product problems are solved by the proposed methods and the value-iteration algorithm that gives the exact optimal value functions. The accuracy of the policies obtained from the IQL, QL and the heuristic methods are compared based on d_{opt}.

A 2^5 full factorial design is constructed for which the preventive maintenance cost, the inventory holding costs and the lost sales costs are chosen as the two-level factors (Table 2). The corresponding factor values for 32 problem instances are given in Appendix B. The Gamma distributed machine degradation has shape and scale parameters $\alpha = 0.5$ and $\beta = 1$ with the failure threshold $L = 10$. The corrective maintenance cost is $c_c = \$800$ and the period length is $C = 4$ hours. For each product, the production amount per period, maximum allowed stock level, demand distribution, setup cost and unit production cost are given in Table 3.

IQL and QL run for 10,000,000 iterations after a warm-up period of 2,000,000 iterations. In the warm-up period, a constant epsilon $\epsilon = 0.1$, is used for the $\epsilon - greedy$ exploration to encourage exploration actions in the initial phase. In addition, due to frequent exploration, the actual initial performance is not yet representative and therefore the authors only start computing d_r and d_{opt}, and updating AVC after the warmup period.

Table 2. 2-level factors for the experimental design

Level	c_p(\$)	c_h^1(\$)	c_h^2(\$)	c_l^1(\$)	c_l^2(\$)
1	300	1	1	100	90
2	400	3	3	200	180

Table 3. Data for the experimental design

Product	q^k(units/period)	I_{max}^k	D^k	c_s^k (\$)	c_k (\$/unit)
1	4	12	Unif [0,2]	50	1
2	8	20	Unif [0,4]	50	1

The results of the experiments are presented in Appendix C, where the d_{opt}, d_r and AVC values are presented for every one million iterations for IQL and QL, and where the AVC value is given for the heuristic method, as well as for the optimal policy obtained policy from value iteration. Similar results are observed in all problem instances. The following results can be observed:

- Both QL and IQL lead to very accurate solutions (which is in agreement with the theoretical results on convergence from literature; Bertsekas and Tsitsiklis, 1996; Evan-Dar and Mansour, 2003).
- The heuristic method is reasonably accurate, but is – in terms of accuracy - outperformed by IQL and QL.
- IQL is much more efficient, that is, it reaches an accurate solution much faster.
- Small and stable values for d_r seem to be an indicator for having found a good policy.

- Stable average cost (AVC) values can also indicate that a good policy has been found.
- AVC as a criterion is able to identify the best method in terms of accuracy. Consequently, it is used to compare the performance of the proposed methods in three-product and four-product examples which suffer from the curse of dimensionality.

I^2 \ I^1	0	1	2	3	4	5	6	≥7
0	P1	P2	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	P2	P2	P2
4	P1	P1	P1	I	I	I	I	I
5	P1	P1	I	I	I	I	I	I
6	P1	P1	I	I	I	I	I	I
7	P1	P1	I	I	I	I	I	I
8	P1	P1	I	I	I	I	I	I
9	P1	P1	I	I	I	I	I	I
10	P1	P1	I	I	I	I	I	I
11	P1	P1	I	I	I	I	I	I
≥12	P1	P1	I	I	I	I	I	I

$1 \le X \le 8$

I^2 \ I^1	0	1	2	3	4	5	6	≥7
0	P1	P2	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	I	I	I
4	P1	P1	P1	I	I	I	I	I
5	P1	P1	I	I	I	I	I	I
6	P1	P1	I	I	I	I	I	I
7	P1	P1	I	I	I	I	I	I
8	P1	P1	I	I	I	I	I	I
9	P1	P1	I	I	I	I	I	I
10	P1	P1	I	I	I	I	I	I
11	P1	P1	I	I	I	I	I	I
≥12	P1	P1	I	I	I	I	I	I

$9 \le X \le 10$

I^2 \ I^1	0	1	2	3	4	5	6	≥7
0	P1	P2	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	I	I	I
4	P1	P1	P1	I	I	I	I	I
5	P1	P1	P1	I	I	I	I	I
6	P1	P1	I	I	I	I	I	I
7	P1	P1	I	I	I	I	I	I
8	P1	P1	I	I	I	I	I	I
9	P1	P1	I	I	I	I	I	I
10	P1	P1	I	I	I	I	I	I
11	P1	P1	I	I	I	I	I	I
≥12	P1	P1	I	I	I	I	I	I

$11 \le X \le 13$

I^2 \ I^1	0	1	2	3	4	5	6	≥7
0	P1	P2	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	M	P2	P2	P2	P2
3	P1	P1	P2	M	P2	I	I	I
4	P1	P1	P1	M	I	I	I	I
5	P1	P1	P1	I	I	I	I	I
6	P1	P1	I	I	I	I	I	I
7	P1	P1	I	I	I	I	I	I
8	P1	P1	I	I	I	I	I	I
9	P1	P1	I	I	I	I	I	I
10	P1	P1	I	I	I	I	I	I
11	P1	P1	I	I	I	I	I	I
≥12	P1	P1	I	I	I	I	I	I

$X = 14$

I^2 \ I^1	0	1	2	3	4	5	6	≥7
0	P1	P2	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2	P2
2	P1	P1	M	M	P2	P2	P2	P2
3	P1	P1	M	M	P2	I	I	I
4	P1	P1	M	M	M	I	I	I
5	P1	P1	M	M	M	I	I	I
6	P1	P1	M	M	I	I	I	I
7	P1	P1	M	I	I	I	I	I
8	P1	P1	I	I	I	I	I	I
9	P1	P1	I	I	I	I	I	I
10	P1	P1	I	I	I	I	I	I
11	P1	P1	I	I	I	I	I	I
≥12	P1	P1	I	I	I	I	I	I

$X = 15$

I^2 \ I^1	0	1	2	3	4	5	6	≥7
0	P1	P2	P2	P2	P2	P2	P2	P2
1	P1	M	M	M	M	P2	P2	P2
2	P1	M	M	M	M	M	M	M
3	P1	M	M	M	M	M	M	M
4	P1	M	M	M	M	M	M	M
5	P1	M	M	M	M	I	I	I
6	P1	M	M	M	I	I	I	I
7	P1	M	M	I	I	I	I	I
8	P1	M	M	I	I	I	I	I
9	P1	M	M	I	I	I	I	I
10	P1	M	M	I	I	I	I	I
11	P1	M	M	I	I	I	I	I
≥12	P1	M	M	I	I	I	I	I

$X = 16$

	I^1							
I^2	0	1	2	3	4	5	6	≥7
0	M	M	M	M	M	M	M	M
1	M	M	M	M	M	M	M	M
2	M	M	M	M	M	M	M	M
3	M	M	M	M	M	M	M	M
4	M	M	M	M	M	M	M	M
5	M	M	M	M	M	I	I	I
6	M	M	M	M	I	I	I	I
7	M	M	M	I	I	I	I	I
8	P1	M	M	I	I	I	I	I
9	P1	M	M	I	I	I	I	I
10	P1	M	M	I	I	I	I	I
11	P1	M	M	I	I	I	I	I
≥12	P1	M	M	I	I	I	I	I

$X = 17$

	I^1							
I^2	0	1	2	3	4	5	6	≥7
0	M	M	M	M	M	M	M	M
1	M	M	M	M	M	M	M	M
2	M	M	M	M	M	M	M	M
3	M	M	M	M	M	M	M	M
4	M	M	M	M	M	M	M	M
5	M	M	M	M	M	I	I	I
6	M	M	M	M	I	I	I	I
7	M	M	M	I	I	I	I	I
8	M	M	M	I	I	I	I	I
9	M	M	M	I	I	I	I	I
10	M	M	M	I	I	I	I	I
11	M	M	M	I	I	I	I	I
≥12	M	M	M	I	I	I	I	I

$18 \leq X \leq 20$

P1: prod. item 1**, P2:** prod. item 2**, M:** prv. maintenance**, I:** idle

Figure 3. Optimal policy for case 9

Before moving on to the three- and four-product examples, it is worth examining the structure of the optimal policy and the policies resulting from the proposed methods. Several of these insights have been used in developing the state aggregation schemes in Section 4.3. Figure 3 shows the optimal action for each state for Case 9. The following structural properties can be observed from the optimal policy:

- If it is not optimal to produce an item with stock level y, then given the same degradation and the stock level of the other product, it is also not optimal to produce that product when its stock level is $y' \geq y$.
- If it is optimal to produce an item with stock level y, then given the same degradation and the stock level of the other product, it is also optimal to produce that product when its stock level is $y' \leq y$.
- If the optimal action is to perform preventive maintenance for a state with degradation level i, then given the same stock levels, it is also optimal to conduct preventive maintenance for degradation level $F > i' \geq i$.
- If the optimal action is to produce item $j \in \{1,2\}$, then either item j has the earliest runout time $min\,\{I^1/\lambda^1, I^2/\lambda^2\}$, where λ^k is the mean demand for item $k = 1,2$;

if $I^1/\lambda^1 = I^2/\lambda^2$ the item with the highest expected lost sales cost $max\,\{c_l^1\lambda^1, c_l^2\lambda^2\}$, in case of stock out, is produced.

- It is not sufficient to only consider the inventory level of the most urgent product to select the optimal decision. For example, under the first aggregation method, the states $(X = 14, I^1 \in \{2, \dots, I_{max}^1\}, I^2 = 3)$ correspond to the same aggregated state, $(X = 14, u = 2,\ I^u = 3)$, where u is the index of the most urgent product; however, the optimal decision differs depending on the inventory level of the first item.

Note that the above properties are observed in all problem instances.

Figure 4 shows the policy resulting from the heuristic method. When the degradation level is $X \leq 10$, the heuristic method does not initiate production in some states (e.g., $(X \leq 10, I^1 = 2, I^2 = 4)$) for which the optimal policy prescribes to produce. Also, in some states (in particular those where the degradation levels $12 \leq X \leq 15$), the heuristic method dictates to conduct preventive maintenance while it is optimal to produce. The main reason for the heuristic method's relatively poor performance is the lost sales resulting from the inaccuracy in the approximation of the state-action values. The heuristic method computes the Q-values as a summation of the corresponding Q-values of the single-product problems that ignore the stock levels of the other products. When both products are reasonably but not very urgent to produce, the heuristic will decide to postpone production, which may later lead to a situation where both products are very urgent to produce, leading to lost sales for the product that has to wait until the other product has been produced. Extending this reasoning to more than two products, it is hypothesized that the heuristic method performs worse for more products. This will be confirmed in Subsection 5.1.2.

Figure 5 shows the policy obtained from the IQL method. The structure of IQL's policy resembles the optimal policy as expected. The difference with the optimal policy is mainly that the IQL rarely visits and updates the Q-values of the states that have very little impact on the optimal policy. Note that the states that are not visited by the underling policies are not shown in Figure 4 and Figure 5.

	I^1					
I^2	0	1	2	3	4	5
0	P1	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	P2
4	P1	P1	I	I	I	I
5	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I
7	P1	P1	I	I	I	I
8	P1	P1	I	I	I	I
9	P1	P1	I	I	I	I
10	P1	P1	I	I	I	I
11	P1	P1	I	I	I	I

$1 \leq X \leq 8$

	I^1					
I^2	0	1	2	3	4	5
0	P1	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2
3	P1	P1	I	I	I	I
4	P1	P1	I	I	I	I
5	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I
7	P1	P1	I	I	I	I
8	P1	P1	I	I	I	I
9	P1	P1	I	I	I	I
10	P1	P1	I	I	I	I
11	P1	P1	I	I	I	I

$9 \leq X \leq 10$

	I^1					
I^2	0	1	2	3	4	5
0	P1	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2
3	P1	P1	M	P2	P2	I
4	P1	P1	M	I	I	I
5	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I
7	P1	P1	I	I	I	I
8	P1	P1	I	I	I	I
9	P1	P1	I	I	I	I
10	P1	P1	I	I	I	I
11	P1	P1	I	I	I	I

$X = 11$

	I^1					
I^2	0	1	2	3	4	5
0	P1	P2	P2	P2	P2	P2
1	P1	M	M	P2	P2	P2
2	P1	M	M	M	M	P2
3	P1	M	M	M	M	M
4	P1	M	M	M	M	M
5	P1	M	M	M	M	M
6	P1	M	M	M	M	M
7	P1	M	M	M	M	M
8	P1	P1	M	M	M	M
9	P1	P1	M	M	M	M
10	P1	P1	M	M	M	M
11	P1	P1	M	M	M	M

$X = 12$

	I^1					
I^2	0	1	2	3	4	5
0	M	M	M	P2	P2	P2
1	M	M	M	M	M	M
2	M	M	M	M	M	M
3	P1	M	M	M	M	M
4	P1	M	M	M	M	M
5	P1	M	M	M	M	M
6	P1	M	M	M	M	M
7	P1	M	M	M	M	M
8	P1	M	M	M	M	M
9	P1	M	M	M	M	M
10	P1	M	M	M	M	M
11	P1	M	M	M	M	M

$X = 13$

	I^1					
I^2	0	1	2	3	4	5
0	M	M	M	M	M	M
1	M	M	M	M	M	M
2	M	M	M	M	M	M
3	M	M	M	M	M	M
4	M	M	M	M	M	M
5	M	M	M	M	M	M
6	M	M	M	M	M	M
7	M	M	M	M	M	M
8	M	M	M	M	M	M
9	M	M	M	M	M	M
10	M	M	M	M	M	M
11	M	M	M	M	M	M

$X = 14$

I^2 \ I^1	0	1	2	3	4	5
0	M	M	M	M	M	M
1	M	M	M	M	M	M
2	M	M	M	M	M	M
3	M	M	M	M	M	M
4	M	M	M	M	M	M
5	M	M	M	M	M	M
6	M	M	M	M	M	M
7	M	M	M	M	M	M
8	M	M	M	M	M	M
9	M	M	M	M	M	M
10	M	M	M	M	M	M
11	M	M	M	M	M	M

$15 \le X \le 20$

P1: prod. item 1, **P2:** prod. item 2,

M: prv. maintenance, **I:** idle

Figure 4. Policy resulting from the heuristic method for Case 9

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	P2	P2
4	P1	P1	P1	I	I	I	I
5	P1	P1	I	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$1 \le X \le 3$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	P2	P2
4	P1	P1	P1	I	I	I	I
5	P1	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 4$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	I	P2
4	P1	P1	P1	I	I	I	I
5	P1	P1	I	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 5$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	I	I
4	P1	P1	P1	I	I	I	I
5	P1	P1	I	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$6 \le X \le 7$

I^2 \ I^1	0	1	2	3	4	5	6
0	P2	P2	P2	P2	P2	P2	P2
1	P2	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	I	I	I
4	P1	P1	P1	I	I	I	I
5	P1	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 8$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	I	I	I
4	P1	P1	P1	I	I	I	I
5	P1	P1	I	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 9$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P2	P1	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	P2	I
4	P1	P1	P1	I	I	I	I
5	P1	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 10$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	P2
3	P1	P1	P2	P2	P2	P2	I
4	P1	P1	I	I	I	I	I
5	P1	P1	I	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 11$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P1	P2	P2	P2	P2	P2
2	P1	P2	P2	P2	P2	I	P2
3	P1	P1	P2	P2	P2	I	I
4	P1	P1	P1	I	I	I	I
5	P1	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 12$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P1	P2	P2	P2	P2	P2
2	P1	P2	P2	P2	P2	I	P2
3	P1	P1	P1	P2	P2	I	I
4	P1	P1	P1	I	I	I	I
5	P1	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 13$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	P2	P2	P2	P2
2	P1	P1	P2	P2	P2	P2	I
3	P1	P1	P1	P2	I	I	I
4	P1	P1	P1	M	I	I	I
5	P1	P1	P1	I	I	I	I
6	P1	P1	I	I	I	I	I
7	P1	P1	I	I	I	I	I
8	P1	P1	I	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 14$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	P2	P2	P2	P2	P2
1	P1	P2	P2	M	P2	P2	P2
2	P1	P1	M	P2	M	P2	P2
3	P1	M	M	M	M	I	I
4	M	M	M	M	M	I	I
5	P1	P1	M	M	I	I	I
6	P1	P1	M	M	I	I	I
7	P1	P1	M	I	I	I	I
8	P1	P1	M	I	I	I	I
9	P1	P1	I	I	I	I	I
10	P1	P1	I	I	I	I	I
11	P1	P1	I	I	I	I	I

$X = 15$

I^2 \ I^1	0	1	2	3	4	5	6
0	P2	M	P2	P2	P2	P2	M
1	P2	M	M	M	M	P2	M
2	M	M	M	M	M	M	M
3	P1	M	M	M	M	M	M
4	M	M	M	M	M	I	M
5	P1	M	M	M	I	I	M
6	P1	M	M	M	I	I	I
7	P1	M	M	I	I	I	I
8	P1	M	M	I	I	I	I
9	P1	M	M	I	I	I	I
10	P1	M	M	I	I	I	I
11	P1	M	M	I	I	I	I

$X = 16$

I^2 \ I^1	0	1	2	3	4	5	6
0	P1	P2	M	M	M	M	M
1	M	M	M	M	M	M	M
2	P2	M	M	M	M	M	M
3	M	M	M	M	M	M	M
4	M	M	M	M	M	M	M
5	M	M	M	M	M	I	M
6	M	M	M	M	I	I	I
7	M	M	M	I	I	I	I
8	M	M	M	I	I	I	I
9	M	M	M	I	I	I	I
10	P1	M	M	I	I	I	I
11	P1	M	M	I	I	I	I

$X = 17$

I^2 \ I^1	0	1	2	3	4	5	6
0	M	M	M	M	M	M	M
1	M	M	M	M	M	M	M
2	M	M	M	M	M	M	M
3	M	M	M	M	M	M	M
4	M	M	M	M	M	M	M
5	M	M	M	M	M	I	I
6	M	M	M	M	I	I	I
7	M	M	M	I	I	I	I
8	M	M	M	I	I	I	I
9	M	M	M	I	I	I	I
10	M	M	M	I	I	I	I
11	M	M	M	I	I	I	I

$18 \leq X \leq 19$

I^2 \ I^1	0	1	2	3	4	5	6
0	M	M	M	M	M	M	M
1	M	M	M	M	M	M	M
2	M	M	M	M	M	M	M
3	M	M	M	M	M	M	M
4	M	M	M	M	M	M	M
5	M	M	M	M	M	I	I
6	M	M	M	M	I	I	I
7	M	M	M	M	I	I	I
8	M	M	M	I	I	I	I
9	M	M	M	I	I	I	I
10	M	M	M	I	I	I	I
11	M	M	M	I	I	I	I

$X = 20$

P1: prod. item 1**, P2:** prod. item 2**, M:** prv. maintenance**, I:** idle

Figure 5. Policy resulting from the IQL method for Case 9

5.1.2. Three and four-product examples

In this section, the results for three and four-product examples are presented. The shape and scale parameters of the degradation are $\alpha = 0.5$ and $\beta = 1$, and the failure threshold is $L = 10$. The period length is $C = 4$ hours. The corrective maintenance cost is $c_c = \$800$, preventive maintenance cost is $c_c = \$300$; and the setup and unit productions costs are $c_s^k = \$50,\ c_k = \$1,\ k = 1,2,\dots,4$. For each product, the production amount per period, maximum allowed stock level, demand distribution, inventory holding and lost sales costs are given in Table 4 and Table 5 corresponding to the three and four-product examples respectively.

Table 4. Data for the three-product example

Product	q^k(units/period)	I^k_{max}	D^k	c_h^k(\$/unit)	c_l^k(\$/unit)
1	4	12	Unif [0,2]	2	200
2	8	20	Unif [0,3]	1	180
3	4	12	Unif [0,2]	2	200

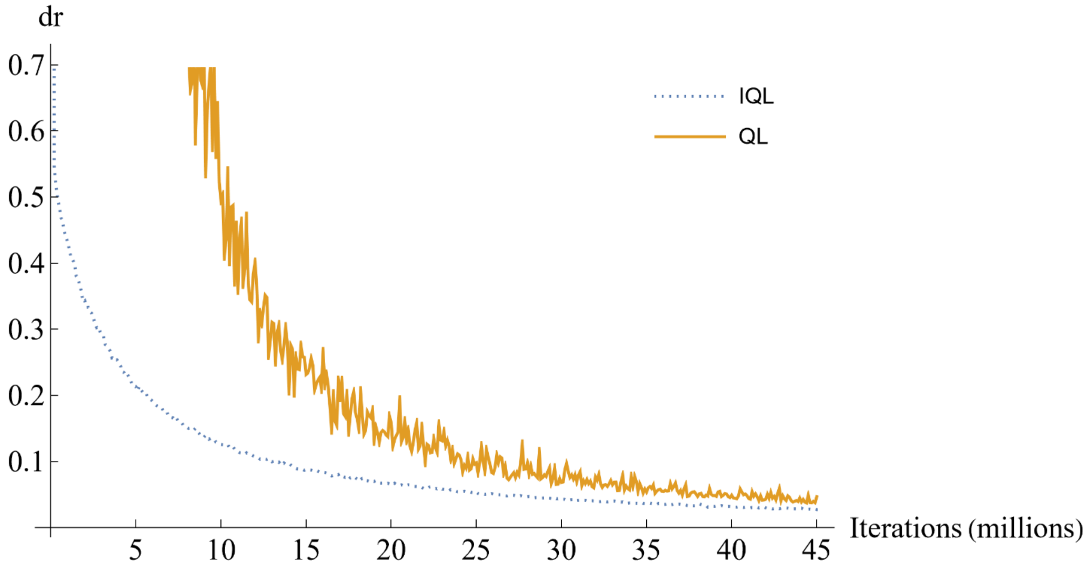

Figure 6. d_r values for IQL and QL with respect to the number of iterations for the three-product problem

The d_r curves of IQL and QL are given in Figure 6. It is calculated for iterations $n = 2K, 3K \ldots, T$, with $K = 100{,}000$ and $T = 45{,}000{,}000$. The d_r of QL is 0.046% at the end of the simulation while IQL reaches that value already at iteration 28,000,000. In Figure 7, the total average cost per period is shown for IQL, QL, and the heuristic method. The average cost of the heuristic method is \$156.20. At the end of the simulation run, the average costs are \$133.29 and \$135.35 for IQL and QL respectively. IQL reaches the average cost \$135.33 already at iteration 16,800,000, which is slightly lower than the value QL converges at the end.

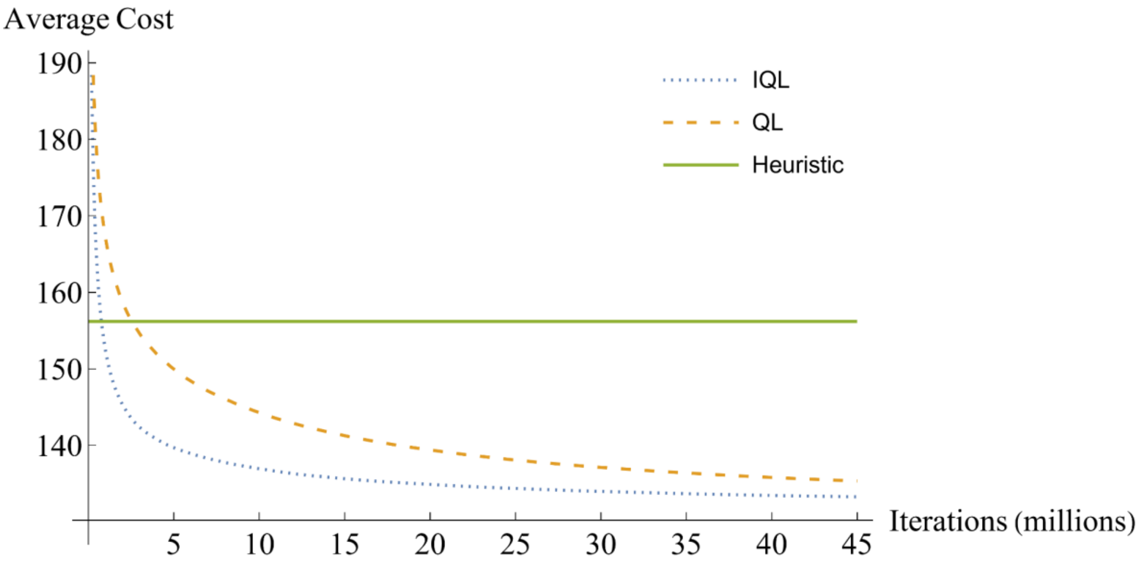

Figure 7. Average cost curves of IQL, QL and the heuristic method for the three-product problem

Table 5. Data for the four-product example

Product	q^k(units/period)	I_{max}^k	D^k	c_h^k($/unit)	c_l^k($/unit)
1	8	20	Unif [0,3]	1	180
2	4	10	Unif [0,2]	2	200
3	4	12	Unif [0,2]	2	200
4	8	20	Unif [0,2]	1	160

Figure 8 shows the d_r values of IQL and QL. It is calculated for iterations $n = 2K, 3K \ldots, T$, with $K = 1{,}000{,}000$ and $T = 200{,}000{,}000$. At the iteration T, d_r is 0.634% for QL, whereas IQL reaches 0.620% already at iteration 13,000,000. Thus, the convergence of the value functions of IQL is again significantly faster than QL. The total average costs per period versus the number of iterations for IQL, QL, and heuristic method are depicted in Figure 9. The average cost of the heuristic method is $235.38, while it is $178.03 and $185.81 for IQL and QL respectively.

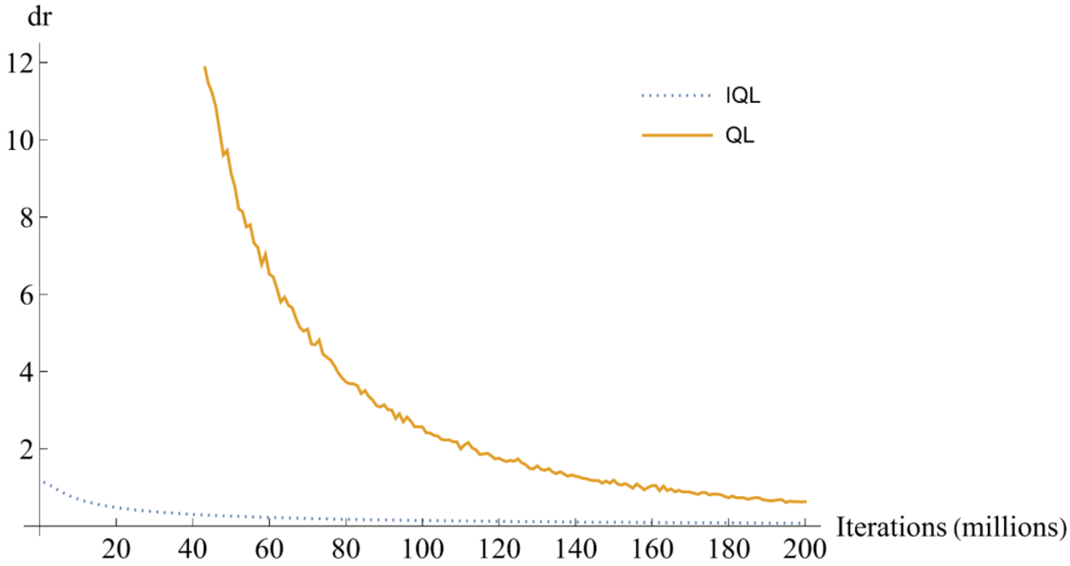

Figure 8. d_r values for IQL and QL with respect to the number of iterations for the four-product problem

As hypothesized in Subsection 5.1.1, the heuristic method's accuracy decreases for increasing number of products. The percentage differences between the average costs of the heuristic method and IQL are equal to 3.71%, 14.67%, 24.36% for the two, three and four-product problems respectively.

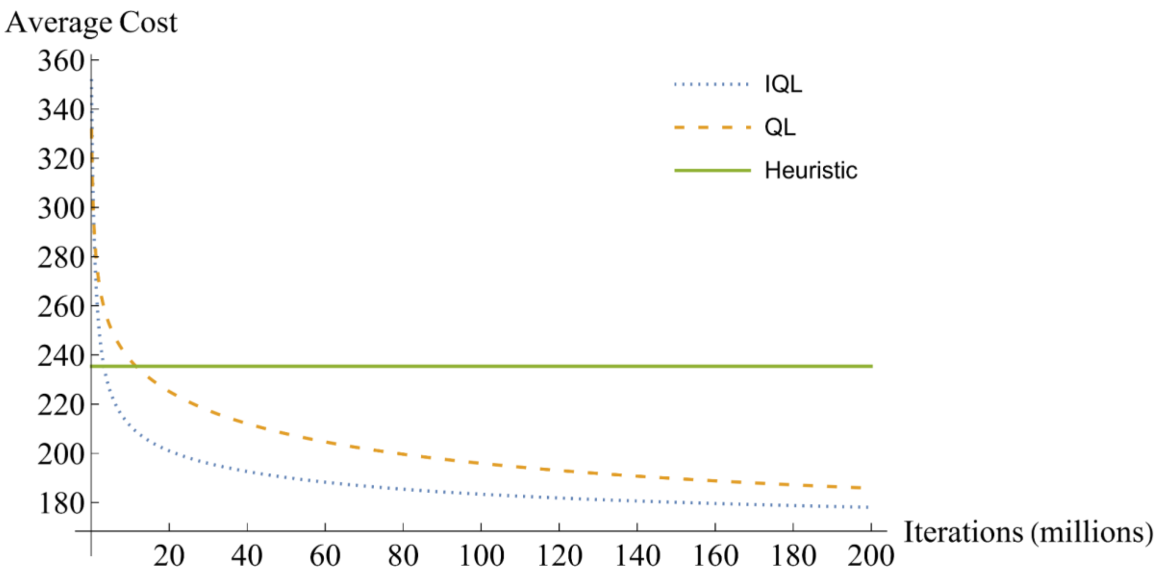

Figure 9. Average cost curves of IQL, QL and the heuristic method for the four-product problem

5.1.3 *Final evaluation*

From the numerical examples, the authors conclude that initializing Q-values to the output of the heuristic method significantly speeds up the learning process, especially for increasing number of products (the state space gets larger). In Table 6, the actual computational times are given for QL, IQL, and heuristic method. For QL, the times are given for T iterations, while for IQL, times are until it reaches the same accuracy as QL. These computations have been performed on a computer with four core processors running at 1.80 GHz and 16 GB memory. From Table 6, it can be concluded that the heuristic method is very fast and that initializing Q-values to the output of the heuristic method significantly speeds up the learning process especially for increasing number of products. In addition, IQL is a robust method since it gives considerably close results to the optimal solution. Hence, the authors conclude that IQL is the best choice. However, due to the exploding state space, even IQL (also the QL and the heuristic method) cannot solve problems with more than four products.

Table 6. Comparison for the computational times of IQL and QL

Number of products	Time (in seconds)		
	IQL	QL	Heuristic method
2	363.80	2,281.6	55.78
3	3688.48	5,800.95	79.02
4	2457.05	35,444.50	153.16

5.2. Performance evaluation of QLA

In this part, the performances of the state aggregation methods are evaluated. The methods with the first, second, and third aggregation schemes are called QLA1, QLA2, and QLA3, respectively. For all methods, the algorithmic parameters are given in Appendix F.

5.2.1. Four-product examples

The performances of QLA1, QLA2, and QLA3 are compared against IQL with the eight four-product instances summarized in Table 7. The resulting policies are evaluated via a simulation run of $5{,}000{,}000$ steps. Table 7 presents the percentage differences in the average costs of the QLA methods ($AVC^{QLAk}, k = 1{,}2{,}3$) and IQL ($AVC^{IQL}$), i.e., $Gap(\%) = 100\,(AVC^{QLAk} - AVC^{IQL})/AVC^{IQL}$ for $k = 1{,}2{,}3$.

QLA1 performs worst among the proposed methods. Hence, the first aggregation method, which only contains the inventory level of the most urgent product in the aggregated state, is not a sufficient approach for capturing the problem structure. The reason is similar as to why the heuristic method is the worst proposed tabular method. In all cases, QLA3 outperforms QLA2, thus the authors conclude that the third aggregation method is the best choice.

Table 7. Instances and the percentage gaps of QLA1, QLA2 and QLA3 with respect to IQL; the other parameters are set to the values in Table 5

Instance	Parameter					Gap (%)		
	c_p (\$)	c_l^1(\$/unit)	c_l^2(\$/unit)	c_l^3(\$/unit)	c_l^4(\$/unit)	QLA1	QLA2	QLA3
1	300	100	90	200	160	18.130	3.978	0.063
2	400	100	90	100	80	27.604	2.898	1.292
3	300	200	90	100	160	19.391	3.141	0.768
4	400	200	90	200	80	24.957	3.573	0.341
5	300	100	180	200	80	30.587	5.766	1.208
6	400	100	180	100	160	29.068	5.372	1.449
7	300	200	180	100	80	19.391	3.346	0.945
8	400	200	180	200	160	26.223	4.529	1.290

Figure 10 shows the average cost curves of QLA1, QLA2, QLA3, and IQL for Case 7. QLA2 and QLA3 converge much faster than IQL. At the end of $15{,}000{,}000$ iterations, the average costs of QLA2 and QLA3 have already converged while IQL is still improving. However, QLA3 converged to a better policy than the QLA2. Note that, to achieve convergence, IQL had to be run for 200,000,000 iterations for the four-product instances.

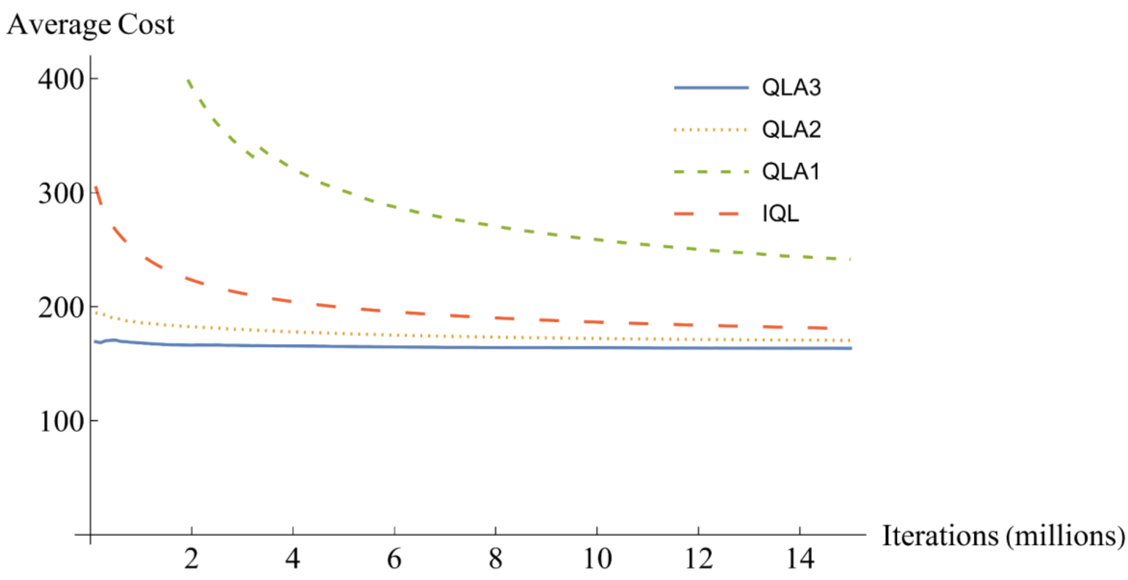

Figure 10. Average cost curves of QLA1, QLA2, QLA3 and IQL for Case 7

5.2.2. 10-product example

In this part, a 10-product example is considered with $q^k = 10$ units, $I_{max}^k = 20$ units for $k = 1,2,\ldots,10$. It is assumed that $c_c = \$800$, $c_p = \$300$, and the setup and unit production costs are $c_s^k = \$50$, $c_k = \$1$ for $k = 1,2,\ldots,10$. The failure threshold is $L = 10$. For each product, the demand distribution, the shape and scale parameters of the degradation, the lost sales and the holding costs are given in Table 8.

Table 8. Data for the 10-product example

Product	D^k	(α, β)	c_l^k (\$)	c_h^k (\$)
1	Unif [0,2]	0.5, 1	200	1
2	Unif [0,2]	1, 1	200	3
3	Unif [0,2]	1, 1	200	2
4	Unif [0,2]	0.5, 1	180	1
5	Unif [0,2]	0.5, 1	160	1
6	Unif [0,2]	1, 1	120	3
7	Unif [0,1]	1, 1	200	2
8	Unif [0,1]	0.5, 1	140	1
9	Unif [0,1]	1, 1	130	1
10	Unif [0,1]	1, 1	100	1

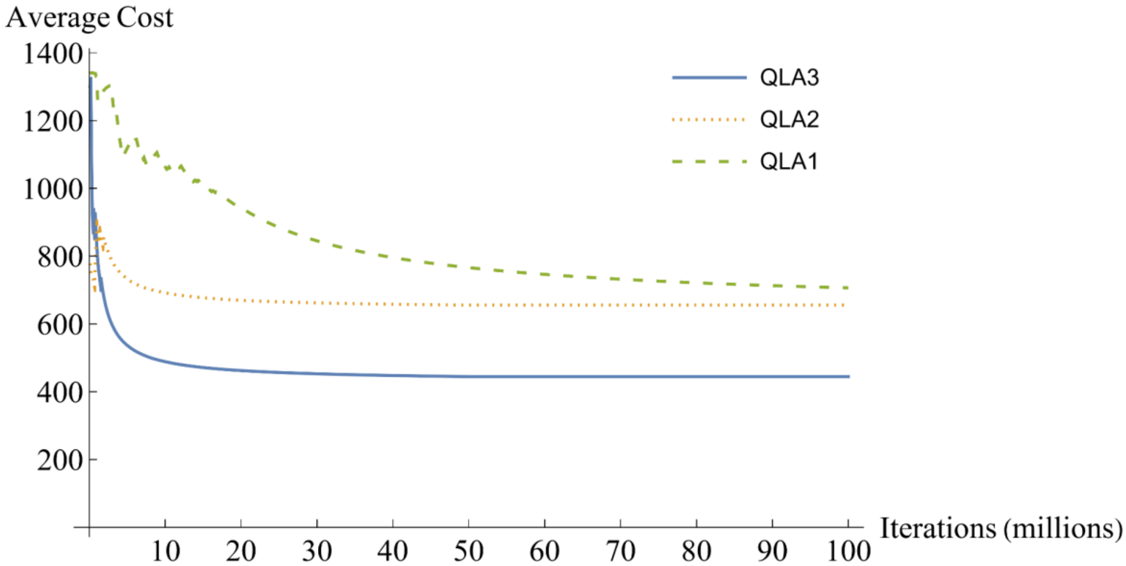

Figure 11. Average cost curves of QLA1, QLA2 and QLA3 for the 10-product problem

As Figure 11 demonstrates, the average cost of QLA2 and QLA3 have already converged before the end of the simulation run. However, QLA3 clearly outperforms QLA2. This confirms that QLA3 is the most accurate state aggregation method.

6. Conclusions

In this study, the integration of lot sizing and condition-based maintenance (CBM) has been studied in case of multiple products and stochastic demand. The problem has been formulated as a Markov Decision Process (MDP) in which production and maintenance decisions are made based on both equipment condition and product inventories. The tabular Q-learning algorithm has been adopted, and a decomposition-based approximate Q-value heuristic method has been proposed to solve the problem in a reasonable time. To speed up the convergence of the Q-learning algorithm, a hybrid (IQL) method has been proposed where the Q-values are initialized by the state-action values obtained by the heuristic method. The IQL and QL methods clearly outperform the heuristic method in terms of accuracy, and IQL converges much faster than QL.

However, the proposed tabular methods (the IQL, QL and the heuristic method) are not scalable to problems with more than four products due to the exponentially growing state space. Therefore, three state aggregation schemes, called QLA1, QLA2, and QLA3, have been developed based on the structure of the problem, and Q-learning has been applied to the aggregated state space. The performance of the state aggregation schemes has been tested for problems with up to ten products. The numeric results demonstrate that QLA3 outperforms QLA1 and QLA2. Moreover, QLA3 converges much faster while developing policies that perform very close to IQL.

The proposed model can be utilized for the joint production, inventory and maintenance control for single-machine multi-product manufacturing systems under CBM. The policies obtained from the solution methods and the analysis provided on the structure of the optimal policy can help the practitioners to reduce the total operating costs.

In this study, it is assumed that the system is reviewed at fixed epochs and the time length of the production lots and the maintenance activities are equal. In future research, this assumption will be relaxed by extending the model to a semi-Markov decision process framework, which allows to make decisions at non-equidistant epochs corresponding to the completion of unit production or completion of a maintenance action. Moreover, imperfect maintenance and its effects on the production and maintenance decisions as well as on the costs could be investigated. Other extensions could be to consider the inclusion of production setup times and/or stochastic maintenance durations.

Disclosure Statement

The authors declare no conflict of interest.

Supplementary Material

The supplementary material is uploaded with the paper.

References

Aghezzaf E-H., Jamali M., and Ait-Kadi D. (2007). An integrated production and preventive maintenance planning model. *European Journal of Operational Research, 181 (2)*, 679–685.

Ben-Daya M., and Makhdoum M. (1998). Integrated production and quality model under various preventive maintenance policies. *Journal of the Operational Research Society, 49(8)*, 840-853.

Ben-Daya M. (2002). The economic production lot-sizing problem with imperfect production processes and imperfect maintenance. *International Journal of Production Economics, 76 (3)*, 257–264.

Bertsekas D.P. (2019). *Reinforcement Learning and Optimal Control*. First Edition. Athena Scientific.

Bertsekas D.P., and Tsitsiklis J.N. (1996*). Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Cheng G.Q., Zhou B.H., and Li L. (2018). Integrated production, quality control and condition-based maintenance for imperfect production systems. *Reliability Engineering & System Safety, 175*, 251–264.

De Jonge B. (2019). Discretizing continuous-time continuous-state deterioration processes, with an application to condition-based maintenance, *Reliability Engineering & System Safety, 188*, 1-5.

El-Ferik S. (2008). Economic production lot-sizing for an unreliable machine under imperfect age-based maintenance policy. *European Journal of Operational Research, 186(1),* 150–163.

Even-Dar E., and Mansour Y. (2003). Learning rates for Q-learning. *Journal of Machine Learning Research*, *5*, 1–25.

Gascon, A., Leachman, R.C. and Lefrancois, P. (1994). Multi-item, single machine scheduling problem with stochastic demands: a comparison of heuristics. *International Journal of Production Research, 32(3),* 583–596.

George P.A., and Powell W.B. (2006). Adaptive stepsizes for recursive estimation with application in approximate dynamic programming. *Machine Learning, 65*, 167-198.

Gijsbrechts J., Boute R. N., Van Mieghem J. A., and Zhang D. J. (2021). Can Deep Reinforcement Learning Improve Inventory Management? Performance on Lost Sales, Dual Sourcing and Multi-Echelon Problems. *Manufacturing and Service Operations Management*, Available at SSRN: http://dx.doi.org/10.2139/ssrn.3302881.

Huang, Jing, Qing Chang, and Jorge Arinez. 2020. Deep Reinforcement Learning Based Preventive Maintenance Policy for Serial Production Lines. *Expert Systems with Applications,160: 113701*.

Iravani S., and Duenyas I. (2002). Integrated maintenance and production control of a deteriorating production system. *IIE Transactions*, *34(5)*, 423–435.

Jafari L., and Makis V. (2015). Joint optimal lot sizing and preventive maintenance policy for a production facility subject to condition monitoring. *International Journal of Production Economics*, *169*, 156-168.

Jafari L., and Makis V. (2019). Optimal Production and Maintenance Policy for a Partially Observable Production System with Stochastic Demand. *International Journal of Industrial and Systems Engineering, 13 (7)*.

Jardine A., Lin D., and Banjevic D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, *20*, 1483–1510.

Khatab A., Diallo C., Aghezzaf E-H., and Venkatadri U. (2018). Integrated production quality and condition-based maintenance for a stochastically deteriorating manufacturing system. *International Journal Production Research, 57(8),* 2480-2497.

Li Y., Gama F. S., Liu M., Yang Z. (2023). A risk averse two-stage stochastic programming model for a joint multi-item capacitated line balancing and lot-*sizing problem. European Journal of Operational Research, 304*, 353-365.

Liao G.L., and Sheu S.H. (2011). Economic production quantity model for randomly failing production process with minimal repair and imperfect maintenance. *International Journal of Production Economics, 130*, 118-124.

Löhndorf N., and Minner S. (2013). Simulation optimization for the stochastic economic lot scheduling problem. *IIE Transactions, 45,* 796-810.

Oroojlooyjadid, A., Nazari, M. , Snyder, L. V. and Takác, M. (2021). A deep Q-network for the Beer Game: deep reinforcement learning for inventory optimization *Manufacturing and Service Operations Management*, Available at https://doi.org/10.1287/msom.2020.0939.

Peng H., and van Houtum G.-J. (2016). Joint Optimization of Condition-based Maintenance and Production Lot-sizing. *European Journal of Operational Research 253*, 94–107.

Powell W.B. (2011). *Approximate Dynamic Programming*. Second edition. John Wiley & Sons.

Qiu J., and Loulou R. (1995). Multiproduct production/inventory control under random demands. *IEEE Transactions on Automatic Control, 40(2)*, 350–356.

Shamsaei F., and Van Vyve M. (2017). Solving integrated production and condition-based maintenance planning problems by MIP modeling. *Flexible Services and Manufacturing Journal, 29*, 184-202.

Sloan, T.W. (2004). A Periodic Review Production and Maintenance Model with Random Demand, Deteriorating Equipment, and Binomial Yield. *Journal of the Operational Research Society, 55(6)*, 647–656.

Sox C.R., Jackson P.L., Bowman A. and Muckstadt J.A. (1999). A review of the stochastic lot scheduling problem. *International Journal of Production Economics, 62*, 181-200.

Suliman S.M., and Jawad S.H. (2012). Optimization of preventive maintenance schedule and production lot size. *International Journal of Production Economics, 137*, 19-28.

Sutton R., and Barto A. (2018). *Reinforcement Learning: An Introduction*. Second edition. MIT Press.

Vanvuchelen, N., Gijsbrechts, J. and Boute, R. (2020). Use of proximal policy optimization for the joint replenishment problem. *Computers in Industry, 119*, 103239.

Wang. J., Xueping L., and Xiotan Z. (2012). Intelligent dynamic control of stochastic economic lot scheduling by agent-based reinforcement learning. *International Journal of Production Research, 50(16)*, 4381-4395.

Watkins C.J. (1989). *Learning from delayed rewards.* Thesis (PhD). Cambridge: Kings College.

Winands E., Adan I., and van Houtum G.J. (2011). The stochastic economic lot scheduling problem: *A survey*. *European Journal of Operational Research, 201(1)*, 1–9.

Xiang Y., Cassady C.R., Jin T., and Zhang C.W. (2014). Joint production and maintenance planning with machine deterioration and random yield. *International Journal Production Research, 52(6)*,1644–1657.

Zheng R., Zhou Y., Gu L., Zhang Z. (2021). Joint optimization of lot sizing and condition-based maintenance for production system using the proportional hazards model. *Computers and Industrial Engineering*, 154.

Appendix A. Pseudocodes

1. Initialize starting state s, $Q(s,a) = 0$ for all $s \in S$ and $a \in A(s)$; AVC (average cost)=0
2. For $t = 1,\ 2, \dots, T$
    2.1. Choose action $a \in A(s)$ for state s ($\epsilon - greedy$)
    2.2. Sample the cost $C(s,a)$ and the next state s' based on the current state s and action a
    2.3. Update the Q-values by the equation:
    $$Q(s,a) \leftarrow Q(s,a) + \alpha\left(C(s,a) + \gamma \min_{a' \in A(s')} Q(s',a') - Q(s,a)\right)$$
    2.4. Update $AVC \leftarrow (AVC\ (t-1) + C(s,a))/t$
    2.5 If $mod(t,K) = 0$ and, then
        $\bar{\pi}(s) = N_t(s)/t \ \ \forall\, s \in S$
        If $t = K$, then
            $$V'(s) = \min_{a \in A(s)} Q(s,a) \ \ \forall\, s \in S$$
        else
            $$d_r = \ 100 \sum_{s \in S} \bar{\pi}(s) \left| \min_{a \in A(s)} Q(s,a) - V'(s) \right| / V'(s)$$
            $$V'(s) = \min_{a \in A(s)} Q(s,a) \quad \forall\, s \in S$$
        end
    end
    2.6. Update $s \leftarrow s'$
3. Return $\mu(s) = arg \min_{a \in A(s)} Q(s,a) \ \ \forall\, s \in S$

Figure A1. Steps of QL including online calculation of AVC and d_r (every K steps)

1. Initialize starting state s, $Q(s,a) = \bar{Q}(s,a)$ for all $s \in S$ and $a \in A(s)$; AVC (average cost) = 0

2. For $t = 1, 2, \dots, T$

   2.1. Choose action $a \in A(s)$; for state s ($\epsilon - greedy$)

   2.2. Sample the cost $C(s,a)$ and the next state s' based on the current state s and action a

   2.3. Update the Q-values by the equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(C(s,a) + \gamma \min_{a' \in A(s')} Q(s',a') - Q(s,a) \right)$$

   2.4. Update AVC (average cost) $\leftarrow (AVC\ (t-1) + C(s,a))/t$

   2.5 If $mod(t,K) = 0$ and, then

$$\bar{\pi}(s) = N_t(s)/t \quad \forall\, s \in S$$

   If $t = K$, then

$$V'(s) = \min_{a \in A(s)} Q(s,a) \ \forall\, s \in S$$

   else

$$d_r = 100 \sum_{s \in S} \bar{\pi}(s) \left| \min_{a \in A(s)} Q(s,a) - V'(s) \right| / V'(s)$$

$$V'(s) = \min_{a \in A(s)} Q(s,a) \quad \forall\, s \in S$$

   end

   end

   2.6. Update $s \leftarrow s'$

3. Return $\mu(s) = arg \min_{a \in A(s)} Q(s,a) \ \ \forall\, s \in S$

Figure A2. Steps of IQL including online calculation of AVC and d_r (every K steps)

1. Initialize starting state s, $Q(s,a) = \bar{Q}(s,a)$ for all s ∈ S and $a \in A(s)$; AVC (average cost) = 0

2. For $t = 1, 2, \dots, T$

   2.1. Choose action $a = arg \min_{a \in A(s)} \bar{Q}(s,a)$

   2.2. Sample the cost $C(s,a)$ and the next state s' based on the current state s and action a

   2.3. Update $AVC \leftarrow (AVC\ (t-1) + C(s,a))/t$

   2.4. Update $s \leftarrow s'$ and increment t

Figure A3. Steps for calculating the average cost of the heuristic method

Input $\mu(s) = arg \min_{a \in A(s)} Q(s, a) \quad \forall\, s \in S$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation

Initialize $V(s),\ s \in S$

While $\Delta > \theta$

For each $s \in S$

$v \leftarrow V(s)$

$V(s) \leftarrow C\big(s, \mu(s)\big) + E[V(s')|s, \mu(s)]$

$\Delta \leftarrow max\{\Delta, |v - V(s)|\}$,

Figure A4. Steps for the policy evaluation

Appendix B. Factor values of tested problem instances

Table B1. Factor Values

	Factors				
Case	c_p	c_l^1	c_h^1	c_l^2	c_h^2
1	300	100	1	90	1
2	300	100	1	90	3
3	300	100	1	180	1
4	300	100	1	180	3
5	300	100	3	90	1
6	300	100	3	90	3
7	300	100	3	180	1
8	300	100	3	180	3
9	300	200	1	90	1
10	300	200	1	90	3
11	300	200	1	180	1
12	300	200	1	180	3
13	300	200	3	90	1
14	300	200	3	90	3
15	300	200	3	180	1
16	300	200	3	180	3
17	400	100	1	90	1
18	400	100	1	90	3
19	400	100	1	180	1
20	400	100	1	180	3
21	400	100	3	90	1
22	400	100	3	90	3
23	400	100	3	180	1
24	400	100	3	180	3
25	400	200	1	90	1
26	400	200	1	90	3
27	400	200	1	180	1
28	400	200	1	180	3
29	400	200	3	90	1
30	400	200	3	90	3
31	400	200	3	180	1
32	400	200	3	180	3

Appendix C. Results of the experiments

Table C1. Results of case 1

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.037	0.339	86.744	0.462	1.743	88.119	8.290	87.112	85.219
2	0.024	0.329	86.033	0.171	1.212	87.205			
3	0.019	0.278	85.860	0.132	1.088	86.815			
4	0.015	0.297	85.714	0.077	0.886	86.576			
5	0.013	0.274	85.583	0.062	0.584	86.382			
6	0.012	0.250	85.531	0.057	0.704	86.233			
7	0.010	0.266	85.506	0.051	0.638	86.135			
8	0.010	0.229	85.468	0.042	0.466	86.078			
9	0.009	0.242	85.439	0.037	0.513	86.030			
10	0.007	0.240	85.428	0.038	0.429	85.986			

Table C2. Results of case 2

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.042	0.380	96.749	0.458	1.180	97.992	9.380	96.978	95.382
2	0.029	0.442	96.062	0.144	1.324	97.122			
3	0.023	0.365	95.829	0.110	0.851	96.721			
4	0.018	0.286	95.677	0.086	0.752	96.506			
5	0.016	0.313	95.614	0.068	0.587	96.392			
6	0.014	0.297	95.584	0.060	0.536	96.266			
7	0.011	0.256	95.549	0.051	0.523	96.189			
8	0.011	0.219	95.517	0.044	0.730	96.117			
9	0.009	0.235	95.500	0.037	0.316	96.040			
10	0.009	0.230	95.519	0.038	0.435	95.993			

Table C3. Results of case 3

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.036	0.422	87.692	0.320	1.901	88.934	8.135	86.747	85.882
2	0.026	0.374	86.854	0.178	1.309	87.815			
3	0.021	0.344	86.518	0.132	1.003	87.421			
4	0.016	0.327	86.404	0.100	1.096	87.212			
5	0.013	0.304	86.338	0.086	0.637	86.921			
6	0.011	0.301	86.278	0.070	0.437	86.890			
7	0.011	0.302	86.241	0.065	0.443	86.800			
8	0.010	0.311	86.193	0.052	0.532	86.725			
9	0.008	0.307	86.159	0.047	0.553	86.688			
10	0.008	0.301	86.142	0.045	0.621	86.621			

Table C4. Results of case 4

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.032	0.358	98.789	0.243	1.372	99.681	7.850	98.238	96.673
2	0.024	0.284	97.921	0.156	1.187	98.539			
3	0.018	0.265	97.566	0.111	1.083	98.101			
4	0.015	0.247	97.386	0.089	0.718	97.861			
5	0.012	0.248	97.283	0.095	0.648	97.711			
6	0.010	0.248	97.233	0.063	0.520	97.613			
7	0.009	0.238	97.160	0.054	0.508	97.515			
8	0.008	0.226	97.110	0.043	0.431	97.455			
9	0.008	0.210	97.072	0.040	0.427	97.408			
10	0.008	0.204	97.049	0.037	0.379	97.359			

Table C5. Results of case 5

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.042	0.380	96.749	0.458	1.180	97.992	8.693	101.58	90.116
2	0.029	0.442	96.062	0.144	1.324	97.122			
3	0.023	0.365	95.829	0.110	0.851	96.721			
4	0.018	0.286	95.677	0.086	0.752	96.506			
5	0.016	0.313	95.614	0.068	0.587	96.392			
6	0.014	0.297	95.584	0.060	0.536	96.266			
7	0.011	0.256	95.549	0.051	0.523	96.189			
8	0.011	0.219	95.517	0.044	0.730	96.117			
9	0.009	0.235	95.500	0.037	0.316	96.040			
10	0.009	0.230	95.519	0.038	0.435	95.993			

Table C6. Results of case 6

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.029	0.373	101.753	0.219	1.149	103.097	7.818	101.58	100.190
2	0.020	0.318	101.011	0.199	0.977	102.150			
3	0.015	0.300	100.811	0.098	0.862	101.760			
4	0.013	0.297	100.707	0.100	0.795	101.487			
5	0.010	0.268	100.627	0.063	0.652	101.324			
6	0.009	0.270	100.574	0.054	0.434	101.223			
7	0.008	0.250	100.534	0.046	0.417	101.116			
8	0.007	0.254	100.508	0.040	0.361	101.081			
9	0.006	0.246	100.476	0.036	0.253	101.028			
10	0.006	0.256	100.474	0.033	0.296	100.969			

Table C7. Results of case 7

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.034	0.346	92.585	0.242	1.294	93.876	8.313	92.183	90.251
2	0.023	0.317	91.611	0.190	1.083	92.735			
3	0.018	0.321	91.308	0.112	0.757	92.265			
4	0.015	0.323	91.167	0.093	0.845	92.011			
5	0.012	0.340	91.059	0.071	0.837	91.857			
6	0.011	0.293	90.995	0.064	0.501	91.735			
7	0.009	0.302	90.917	0.055	0.561	91.614			
8	0.008	0.256	90.866	0.049	0.510	91.535			
9	0.008	0.271	90.844	0.042	0.514	91.475			
10	0.007	0.249	90.797	0.037	0.277	91.442			

Table C8. Results of case 8

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.030	0.380	103.254	0.218	0.893	104.860	7.554	102.872	101.334
2	0.021	0.345	102.422	0.176	0.972	103.701			
3	0.016	0.341	102.132	0.099	0.612	103.230			
4	0.013	0.341	101.966	0.078	0.545	102.966			
5	0.011	0.337	101.841	0.085	0.494	102.779			
6	0.010	0.354	101.810	0.054	0.470	102.685			
7	0.008	0.393	101.771	0.048	0.437	102.621			
8	0.008	0.335	101.745	0.041	0.292	102.554			
9	0.007	0.319	101.702	0.036	0.278	102.475			
10	0.006	0.303	101.673	0.034	0.349	102.405			

Table C9. Results of case 9

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.038	0.526	87.530	0.270	2.427	88.653	8.672	89.100	85.667
2	0.028	0.464	86.782	0.205	1.323	87.714			
3	0.022	0.520	86.596	0.140	0.955	87.340			
4	0.017	0.429	86.429	0.112	1.110	87.117			
5	0.016	0.412	86.347	0.085	0.752	86.955			
6	0.012	0.389	86.290	0.068	0.830	86.811			
7	0.011	0.446	86.254	0.064	0.620	86.734			
8	0.009	0.386	86.231	0.055	0.424	86.685			
9	0.009	0.388	86.207	0.053	0.524	86.637			
10	0.008	0.401	86.201	0.043	0.366	86.606			

Table C10. Results of case 10

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.034	0.465	97.655	0.245	1.182	98.783	7.951	99.437	95.691
2	0.023	0.417	96.832	0.162	1.200	97.695			
3	0.020	0.391	96.502	0.149	0.989	97.344			
4	0.014	0.310	96.341	0.093	1.597	97.150			
5	0.012	0.315	96.226	0.071	1.350	96.996			
6	0.011	0.287	96.167	0.060	1.294	96.899			
7	0.010	0.279	96.102	0.052	1.141	96.848			
8	0.008	0.281	96.082	0.045	0.396	96.774			
9	0.008	0.268	96.045	0.042	0.356	96.733			
10	0.007	0.259	96.020	0.039	0.393	96.705			

Table C11. Results of case 11

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.041	0.356	88.568	0.300	1.956	89.617	8.836	90.631	85.910
2	0.029	0.319	87.661	0.196	1.402	88.771			
3	0.023	0.334	87.310	0.143	0.912	88.255			
4	0.019	0.308	87.107	0.138	0.736	88.036			
5	0.015	0.253	86.974	0.096	0.599	87.819			
6	0.014	0.279	86.882	0.077	0.691	87.733			
7	0.011	0.331	86.839	0.067	0.699	87.611			
8	0.010	0.297	86.810	0.057	0.508	87.514			
9	0.010	0.258	86.786	0.054	0.549	87.499			
10	0.008	0.245	86.767	0.045	0.491	87.452			

Table C12. Results of case 12

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.035	0.328	99.284	0.319	1.293	100.636	7.764	98.342	96.650
2	0.024	0.313	98.290	0.171	1.090	99.398			
3	0.019	0.334	97.892	0.134	1.081	98.888			
4	0.016	0.298	97.753	0.090	0.611	98.598			
5	0.013	0.277	97.631	0.075	0.663	98.381			
6	0.012	0.285	97.540	0.066	0.430	98.214			
7	0.010	0.261	97.493	0.054	0.408	98.112			
8	0.009	0.267	97.427	0.048	0.338	98.023			
9	0.008	0.278	97.417	0.043	0.328	97.948			
10	0.008	0.256	97.390	0.041	0.384	97.888			

Table C13. Results of case 13

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.036	0.273	93.030	0.279	1.174	94.134	8.142	93.982	91.335
2	0.024	0.274	92.212	0.162	0.896	93.089			
3	0.020	0.277	91.863	0.122	0.807	92.703			
4	0.016	0.225	91.731	0.087	1.274	92.431			
5	0.013	0.215	91.615	0.075	1.169	92.254			
6	0.011	0.262	91.546	0.065	1.226	92.137			
7	0.010	0.213	91.493	0.058	0.350	92.056			
8	0.009	0.196	91.503	0.048	0.278	91.978			
9	0.008	0.193	91.458	0.043	0.413	91.930			
10	0.007	0.194	91.448	0.041	0.242	91.876			

Table C14. Results of case 14

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.031	0.255	103.085	0.226	0.897	104.536	8.817	103.763	101.212
2	0.022	0.212	102.267	0.137	0.784	103.444			
3	0.017	0.178	101.955	0.108	0.819	103.045			
4	0.013	0.177	101.787	0.084	0.395	102.773			
5	0.012	0.198	101.740	0.068	0.734	102.600			
6	0.010	0.158	101.715	0.057	0.407	102.468			
7	0.009	0.159	101.698	0.048	0.447	102.382			
8	0.008	0.155	101.659	0.042	0.603	102.334			
9	0.007	0.153	101.659	0.037	0.325	102.272			
10	0.006	0.140	101.644	0.033	0.273	102.242			

Table C15. Results of case 15

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.038	0.294	93.818	0.271	1.306	95.447	8.035	95.396	91.791
2	0.027	0.262	92.953	0.169	0.983	94.152			
3	0.021	0.226	92.580	0.128	0.713	93.562			
4	0.016	0.241	92.370	0.096	0.782	93.249			
5	0.014	0.236	92.255	0.086	0.521	93.078			
6	0.013	0.240	92.187	0.067	0.435	92.909			
7	0.011	0.235	92.144	0.060	0.525	92.778			
8	0.010	0.229	92.086	0.049	0.393	92.688			
9	0.009	0.211	92.040	0.046	0.328	92.598			
10	0.008	0.256	92.032	0.042	0.242	92.545			

Table C16. Results of case 16

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.033	0.298	105.675	0.232	1.102	106.708	7.229	104.656	102.591
2	0.024	0.250	104.699	0.151	1.413	105.508			
3	0.019	0.232	104.268	0.109	0.651	105.046			
4	0.016	0.191	104.094	0.086	0.721	104.750			
5	0.013	0.222	103.990	0.068	0.528	104.568			
6	0.011	0.237	103.904	0.062	0.401	104.455			
7	0.009	0.229	103.855	0.056	0.497	104.376			
8	0.009	0.202	103.837	0.045	0.329	104.288			
9	0.008	0.245	103.810	0.041	0.528	104.215			
10	0.007	0.209	103.770	0.038	0.271	104.147			

Table C17. Results of case 17

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.037	0.457	98.162	0.285	2.093	99.518	11.053	99.00	96.244
2	0.024	0.411	97.538	0.178	1.311	98.665			
3	0.019	0.393	97.251	0.125	0.995	98.022			
4	0.016	0.390	97.104	0.097	0.706	97.732			
5	0.013	0.459	97.044	0.082	0.798	97.397			
6	0.012	0.376	96.986	0.069	1.461	97.223			
7	0.010	0.355	96.924	0.059	1.108	97.198			
8	0.009	0.345	96.887	0.058	1.325	97.112			
9	0.008	0.368	96.861	0.047	1.038	97.092			
10	0.007	0.327	96.841	0.041	0.467	97.057			

Table C18. Results of case 18

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.030	0.328	107.812	0.237	1.171	108.975	9.885	108.621	106.249
2	0.022	0.302	107.144	0.210	1.198	108.045			
3	0.015	0.291	106.906	0.119	1.028	107.687			
4	0.013	0.251	106.781	0.091	0.884	107.440			
5	0.011	0.203	106.695	0.072	0.596	107.326			
6	0.009	0.214	106.670	0.059	0.593	107.212			
7	0.008	0.192	106.622	0.050	0.545	107.147			
8	0.007	0.213	106.599	0.046	0.508	107.067			
9	0.007	0.192	106.577	0.039	0.397	107.021			
10	0.006	0.172	106.576	0.039	0.407	106.983			

Table C19. Results of case 19

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.035	0.422	98.894	0.308	2.104	100.334	10.734	99.160	97.109
2	0.024	0.357	98.119	0.180	1.160	99.169			
3	0.019	0.328	97.797	0.138	1.259	98.707			
4	0.015	0.310	97.654	0.103	0.965	98.439			
5	0.013	0.287	97.567	0.081	0.803	98.270			
6	0.012	0.279	97.525	0.077	0.627	98.166			
7	0.010	0.281	97.492	0.058	0.580	98.073			
8	0.009	0.309	97.476	0.058	0.654	97.992			
9	0.008	0.273	97.437	0.046	0.367	97.944			
10	0.007	0.273	97.421	0.041	0.439	97.877			

Table C20. Results of case 20

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.031	0.392	109.869	0.251	1.424	111.184	9.624	110.212	107.754
2	0.021	0.360	109.010	0.162	1.248	110.144			
3	0.017	0.327	108.717	0.124	0.898	109.701			
4	0.014	0.364	108.528	0.113	0.910	109.403			
5	0.012	0.330	108.463	0.095	0.723	109.190			
6	0.010	0.303	108.399	0.061	0.598	109.035			
7	0.009	0.318	108.358	0.054	0.565	108.947			
8	0.009	0.285	108.314	0.048	0.386	108.874			
9	0.008	0.291	108.281	0.042	0.505	108.810			
10	0.006	0.264	108.259	0.038	0.381	108.757			

Table C21. Results of case 21

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.031	0.417	102.893	0.320	1.604	104.371	10.369	103.532	101.467
2	0.022	0.397	102.301	0.161	1.203	103.461			
3	0.017	0.373	102.089	0.116	0.969	103.098			
4	0.013	0.351	101.911	0.092	0.896	102.881			
5	0.012	0.321	101.794	0.076	0.661	102.708			
6	0.010	0.334	101.752	0.060	0.640	102.598			
7	0.010	0.324	101.745	0.055	0.634	102.515			
8	0.008	0.313	101.704	0.048	0.581	102.447			
9	0.007	0.273	101.658	0.043	0.554	102.391			
10	0.006	0.271	101.625	0.035	0.396	102.348			

Table C22. Results of case 22

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.028	0.346	113.279	0.223	1.291	114.246	9.468	113.259	110.932
2	0.020	0.301	112.536	0.138	1.532	113.338			
3	0.016	0.253	112.266	0.102	1.225	112.956			
4	0.013	0.238	112.152	0.076	0.739	112.716			
5	0.010	0.241	112.075	0.063	0.438	112.538			
6	0.009	0.231	112.002	0.056	0.974	112.386			
7	0.008	0.240	111.976	0.049	0.489	112.304			
8	0.007	0.236	111.962	0.042	0.384	112.222			
9	0.007	0.209	111.963	0.037	0.405	112.165			
10	0.006	0.204	111.940	0.046	0.522	112.118			

Table C23. Results of case 23

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.249	0.329	103.991	0.032	1.479	105.332	10.236	103.787	101.888
2	0.166	0.316	103.190	0.023	0.933	104.209			
3	0.116	0.298	102.907	0.018	0.788	103.705			
4	0.091	0.298	102.719	0.014	0.675	103.459			
5	0.073	0.288	102.618	0.012	0.578	103.265			
6	0.062	0.304	102.503	0.012	0.501	103.142			
7	0.055	0.264	102.452	0.009	0.5	103.006			
8	0.046	0.272	102.416	0.008	0.381	102.916			
9	0.045	0.254	102.380	0.008	0.427	102.844			
10	0.039	0.254	102.341	0.007	0.32	102.780			

Table C24. Results of case 24

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.029	0.234	114.472	0.224	1.708	116.066	9.189	114.70	112.417
2	0.020	0.218	113.675	0.174	0.856	114.976			
3	0.016	0.172	113.349	0.109	0.592	114.527			
4	0.013	0.158	113.169	0.083	0.755	114.265			
5	0.010	0.158	113.013	0.071	0.611	114.057			
6	0.010	0.144	112.941	0.058	0.491	113.941			
7	0.008	0.145	112.895	0.048	0.357	113.853			
8	0.008	0.141	112.852	0.044	0.475	113.766			
9	0.007	0.131	112.813	0.040	0.295	113.684			
10	0.006	0.170	112.809	0.036	0.260	113.637			

Table C25. Results of case 25

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.039	0.268	99.198	0.304	1.970	100.179	10.522	101.11	97.314
2	0.026	0.179	98.384	0.186	1.377	98.944			
3	0.021	0.179	98.108	0.136	0.948	98.538			
4	0.017	0.200	97.990	0.106	0.817	98.256			
5	0.014	0.157	97.911	0.085	0.704	98.117			
6	0.012	0.168	97.868	0.074	0.617	97.985			
7	0.012	0.155	97.820	0.063	0.550	97.917			
8	0.010	0.151	97.807	0.059	0.685	97.860			
9	0.009	0.148	97.770	0.053	0.562	97.803			
10	0.008	0.165	97.737	0.048	0.438	97.736			

Table C26. Results of case 26

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.033	0.387	109.249	0.256	1.531	110.404	9.497	110.939	106.951
2	0.025	0.304	108.377	0.165	1.771	109.282			
3	0.019	0.275	108.045	0.164	1.429	108.790			
4	0.014	0.276	107.830	0.099	0.869	108.509			
5	0.012	0.283	107.725	0.109	0.706	108.316			
6	0.011	0.289	107.666	0.062	1.100	108.119			
7	0.009	0.272	107.633	0.057	1.051	108.017			
8	0.008	0.293	107.584	0.048	0.955	107.928			
9	0.007	0.291	107.542	0.045	0.951	107.852			
10	0.006	0.289	107.508	0.037	1.027	107.786			

Table C27. Results of case 27

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.039	0.520	99.690	0.288	1.594	101.190	10.52	102.334	97.270
2	0.028	0.405	98.673	0.200	1.269	99.929			
3	0.023	0.386	98.291	0.154	0.994	99.368			
4	0.019	0.405	98.095	0.113	0.877	99.084			
5	0.015	0.357	97.970	0.089	0.576	98.849			
6	0.014	0.364	97.890	0.078	0.777	98.724			
7	0.012	0.375	97.833	0.066	0.676	98.591			
8	0.010	0.369	97.743	0.058	0.585	98.499			
9	0.010	0.353	97.700	0.049	0.519	98.395			
10	0.009	0.303	97.663	0.045	0.413	98.340			

Table C28. Results of case 28

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.035	0.231	110.831	0.385	1.593	112.270	9.324	113.111	108.049
2	0.024	0.259	109.831	0.171	1.098	110.856			
3	0.019	0.181	109.498	0.123	0.846	110.318			
4	0.015	0.200	109.315	0.099	0.752	109.999			
5	0.013	0.191	109.178	0.088	0.497	109.783			
6	0.011	0.195	109.092	0.065	0.368	109.643			
7	0.010	0.197	109.036	0.058	0.489	109.513			
8	0.010	0.174	108.961	0.050	1.052	109.428			
9	0.008	0.194	108.923	0.045	1.075	109.354			
10	0.008	0.216	108.875	0.041	1.014	109.286			

Table C29. Results of case 29

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.034	0.533	104.677	0.256	2.052	105.452	10.186	105.963	102.305
2	0.025	0.426	103.819	0.166	1.016	104.406			
3	0.019	0.368	103.455	0.117	0.968	104.020			
4	0.015	0.391	103.316	0.096	0.709	103.720			
5	0.014	0.356	103.213	0.077	0.650	103.556			
6	0.012	0.354	103.131	0.065	0.413	103.399			
7	0.010	0.357	103.087	0.061	0.505	103.287			
8	0.009	0.340	103.049	0.051	0.412	103.211			
9	0.008	0.344	103.019	0.044	0.312	103.120			
10	0.007	0.349	102.984	0.037	0.296	103.074			

Table C30. Results of case 30

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.029	0.211	114.400	0.246	1.258	115.640	7.760	115.607	112.51
2	0.022	0.198	113.541	0.145	0.931	114.474			
3	0.016	0.166	113.284	0.109	0.915	114.036			
4	0.013	0.185	113.138	0.085	0.446	113.823			
5	0.011	0.154	112.993	0.069	0.555	113.664			
6	0.010	0.147	112.959	0.056	0.513	113.539			
7	0.008	0.127	112.870	0.051	0.236	113.418			
8	0.007	0.122	112.853	0.043	0.329	113.328			
9	0.007	0.116	112.824	0.038	0.353	113.260			
10	0.006	0.097	112.779	0.033	0.331	113.212			

Table C31. Results of case 31

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.037	0.370	104.941	0.271	1.711	106.982	9.704	107.012	103.111
2	0.026	0.312	104.106	0.171	1.485	105.501			
3	0.021	0.277	103.835	0.166	1.290	104.949			
4	0.018	0.285	103.662	0.101	1.372	104.663			
5	0.014	0.226	103.585	0.098	0.968	104.450			
6	0.013	0.254	103.538	0.070	0.930	104.317			
7	0.011	0.253	103.496	0.061	0.818	104.241			
8	0.009	0.251	103.431	0.054	0.763	104.164			
9	0.009	0.167	112.408	0.049	0.765	104.086			
10	0.008	0.245	103.387	0.042	0.646	104.018			

Table C32. Results of case 32

Iteration (millions)	IQL			QL			Heuristic		Value iteration
	d_r	d_{opt}	AVC	d_r	d_{opt}	AVC	d_{opt}	AVC	AVC
1	0.033	0.178	117.085	0.245	1.083	118.038	8.465	117.716	114.493
2	0.023	0.172	116.087	0.153	1.087	116.719			
3	0.019	0.193	115.700	0.138	0.851	116.255			
4	0.014	0.173	115.509	0.091	0.679	115.927			
5	0.012	0.176	115.429	0.071	0.622	115.701			
6	0.011	0.170	115.336	0.058	0.492	115.582			
7	0.009	0.163	115.279	0.055	0.586	115.464			
8	0.009	0.165	115.239	0.048	0.289	115.386			
9	0.008	0.151	115.201	0.040	0.324	115.318			
10	0.007	0.152	115.174	0.038	0.305	115.276			

Appendix D

In order to try to establish the Bellman equation in a somewhat compact form, the following notation is introduced:

- $\mathbf{1}_{X=F}$ is the indicator variable with value equal to 1 if and only if $X = F$
- R_n is a binary variable and takes value 1 if preventive maintenance is carried out in period n, otherwise it is equal to 0.

Adopting this notation, the Bellman optimality equation can be established in the following form:

$$V\left(X=i, I^1, \ldots, I^{|M|}\right)$$

$$= \min_{\substack{q^m = C\rho_m Y^m \quad \forall m\in M \\ \sum_{m\in M} Y^m + R + \mathbf{1}_{X=F} \le 1 \\ q^m + I^m \le I^m_{max} \quad \forall m \in M}} \Bigg\{ c_c \mathbf{1}_{X=F} + R c_p + (R + \mathbf{1}_{X=F}) \sum_{m\in M} \Bigg[c_h^m E[(I^m - D^m)^+]$$

$$+ c_l^m E[(D^m - I^m)^+] + \gamma E\left[V\left(1, (I^1 - D^1)^+, \ldots, \left(I^{|M|} - D^{|M|}\right)^+\right) \right] \Bigg]$$

$$+ \left(1 - \sum\nolimits_{m\in M} Y^m - R - \mathbf{1}_{X=F}\right) \sum_{m\in M} \Bigg[c_h^m E[(I^m - D^m)^+]$$

$$+ c_l^m E[(D^m - I^m)^+] + \gamma E\left[V\left(i, (I^1 - D^1)^+, \ldots, \left(I^{|M|} - D^{|M|}\right)^+\right) \right] \Bigg]$$

$$+ \sum_{m\in M} Y^m \Bigg[\sum_{u \in M\setminus\{m\}} \left[c_h^u E[(I^u - D^u)^+] + c_l^u E[(D^u - I^u)^+] \right]$$

$$+ \sum\nolimits_{j=i}^{F-1} \left[\left(\mathbf{P}(\mathbf{m})^{(q^m)}\right)_{ij} (c_s^m + c_m q^m + c_h^m E[(I^m + q^m - D^m)^+] + c_l^m E[(D^m - q^m - I^m)^+] \right.$$

$$\left. + \gamma\, E\left[V(j, (I^1 - D^1)^+, \ldots, (I^m + q^m - D^m)^+, \ldots, \left(I^{|M|} - D^{|M|}\right)^+) \right] \right) \Big]$$

$$+ \sum\nolimits_{k=1}^{q^m} \left[P\left\{T_F(m)^{(i)} = k\right\} (c_s^m + c_m k + c_h^m E[(I^m + k - D^m)^+] \right.$$

$$\left. + c_l^m E[(D^m - k - I^m)^+] + \gamma E \left[V(F, (I^1 - D^1)^+, \ldots, (I^m + k - D^m)^+, \ldots, \left(I^{|M|} - D^{|M|}\right)^+) \right] \right) \Big] \Bigg] \Bigg\}$$

where the subscripts n have been suppressed to make the notation more concise and because the problem is stationary. The constraint $q^m = C\rho_m Y^m \quad \forall m \in M$, ensures that either production is done at full capacity for product m or item m is not produced; and the constraint $q^m + I^m \le I^m_{max} \quad \forall m \in M$, excludes "producing item m" from the action space if $q^m + I^m$ would exceed I^m_{max}. Under the constraint $\sum_{m\in M} Y^m +$

$R+\mathbf{1}_{X=F} \leq 1$, there are four possibilities: (1) a particular product can be produced, (2) preventive maintenance is conducted, (3) corrective maintenance is done if the equipment is at the failure level indicating $\mathbf{1}_{X=F} = 1$, (4) the system is kept idle.

Appendix E

For subproblem $m \in M$, the optimal state-action values $Q(s_m, a_m)$, with $s_m = (X = i, I^m) \in S_m$ and action $a_m \in A_m$, satisfy

$$Q_m(i, I^m, a_m) = EC_m(i, I^m, a_m) + \gamma E\left[V^m\left(i', I^{m'}\right), | i, I^m, a_m\right],$$

$$a_m \in \begin{cases} \{stay\ idle, do\ prv.maint.\} \ \ if \quad C\rho_m + I^m > I^m_{max}\ and\ i < F \\ \{stay\ idle, produce\ q^m = C\rho_m, do\ prv.maint.(R = 1)\}\ if \quad C\rho_m + I^m \leq I^m_{max}\ and\ i < F \\ \{do\ cor.maint.\}\ if\ i = F \end{cases}$$

where $V^m\left(i', I^{m'}\right) = \min_{a'_m} Q_m\left(i', I^{m'}, a'_m\right)$, is the optimal value function, and $EC_m(i, I^m, a_m)$ is the one-period expected cost given by

$$EC_m(i, I^m, a_m) = c_c \mathbf{1}_{X=F} + Rc_p + (1 - Y^m)\ (c^m_h E[(I^m - D^m)^+]$$

$$+c^m_l E[(D^m - I^m)^+]) + Y^m \Bigg(P\left\{T_F(m)^{(i)} > q^m\right\} (c^m_s + c_m q^m + c^m_h E[(I^m + q^m - D^m)^+]$$

$$+c^m_l E[(D^m - q^m - I^m)^+])$$

$$+\sum_{k=1}^{q^m} \Bigg[P\left\{T_F(m)^{(i)} = k\right\} (c^m_s + c_m k + c^m_h E[(I^m + k - D^m)^+]$$

$$+c^m_l E[(D^m - k - I^m)^+]) \Bigg] \Bigg),$$

which satisfies $Y^m + R+\mathbf{1}_{X=F} \leq 1$. $V^m(X = i, I^m)$ is the total minimum expected cost of the subproblem $m \in M$ and for state $s_m = (X = i, I^m)$. It can be expressed as

$$V^m(X=i,I^m)=\min_{\substack{q^m=C\rho_m Y^m\\ Y^m+R+\mathbf{1}_{X=F}\le 1\\ q^m+I^m\le I^m_{max}}}\Big\{c_c\mathbf{1}_{X=F}+Rc_p+(R+\mathbf{1}_{X=F})(c_h^m E[(I^m-D^m)^+]$$

$$+c_l^m E[(D^m-I^m)^+]+\gamma E[V^m(1,(I^m-D^m)^+)\,])$$

$$+(1-Y^m-R-\mathbf{1}_{X=F})(c_h^m E[(I^m-D^m)^+]+$$

$$+c_l^m E[(D^m-I^m)^+]+\gamma E[V^m(i,(I^m-D^m)^+)\,])$$

$$+Y^m\Bigg(\sum_{j=i}^{F-1}\Bigg[\left(\mathbf{P}(\mathbf{m})^{(q^m)}\right)_{ij}(c_s^m+c_m q^m+c_h^m E[(I^m+q^m-D^m)^+]$$

$$+c_l^m E[(D^m-q^m-I^m)^+]+\gamma E[V^m(j,(I^m+q^m-D^m)^+])\Bigg]$$

$$+\sum_{k=1}^{q^m}\Bigg[P\{T_F(m)^{(i)}\ =k\}\,(c_s^m+c_m k+c_h^m E[(I^m+k-D^m)^+]$$

$$+c_l^m E[(D^m-k-I^m)^+]+\gamma E\ [V^m(F,(I^m+k-D^m)^+])\Bigg]\Bigg)\Big\}.$$

Appendix F

Table F1. Step size parameters for IQL and QL

Number of Products	IQL		QL	
	b_0	b	b_0	b
2	1	1	1	5
3	0.1	50	1	5
4	0.1	50	1	5

Table F2. Algorithmic parameters for QLA1, QLA2 and QLA3

Number of Products	QLA1			QLA2			QLA3		
	ϵ	b_0	b	ϵ	b_0	b	ϵ	b_0	b
4	0.3	10^{-3}	10^5	0.3	0.02	500	0.2	1	5
10	0.3	10^{-3}	10^5	0.3	0.02	10^3	0.2	0.1	50

Supplementary Material. One step probability transition matrices of degradation

$\boldsymbol{P1}$ (one step probability transition matrix of the products with a production rate of 1) =

0.483942	0.289925	0.110157	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0015165	0.000872064	0.000504072	0.000292602	0.000170453	0.0000995962	0.0000583458	0.0000342577	0.0000201543	0.0000118779	7.01116×10^{-6}	0.0000101757
0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.00151649	0.000872064	0.000504072	0.000292602	0.000170453	0.0000995962	0.0000583458	0.0000342577	0.0000201543	0.0000118779	0.0000171869
0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.00151649	0.000872064	0.000504072	0.000292602	0.000170453	0.0000995962	0.0000583458	0.0000342577	0.0000201543	0.0000290648
0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.00151649	0.000872064	0.000504072	0.000292602	0.000170453	0.0000995962	0.0000583458	0.0000342577	0.0000492191
0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.00151649	0.000872064	0.000504072	0.000292602	0.000170453	0.0000995962	0.0000583458	0.0000834768
0.	0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0015165	0.000872064	0.000504072	0.000292602	0.000170453	0.0000995962	0.000141823
0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0015165	0.000872064	0.000504072	0.000292602	0.000170453	0.000241419
0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0015165	0.000872064	0.000504072	0.000292602	0.000411872
0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0015165	0.000872064	0.000504072	0.000704474
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0015165	0.000872064	0.00120855
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0015165	0.00208061
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110156	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00265404	0.0035971
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110157	0.0536081	0.027966	0.0151169	0.00835161	0.00468284	0.00625115
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110157	0.0536081	0.027966	0.0151169	0.00835161	0.010934
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110157	0.0536081	0.027966	0.0151169	0.0192856
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110157	0.0536081	0.027966	0.0344025
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110157	0.0536081	0.0623686
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.110157	0.115977
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.289925	0.226133
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.483942	0.516058
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.

$\boldsymbol{P2}$ (one step probability transition matrix of the products with a production rate of 2) =

0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000052286	0.0000299855	0.0000172636	9.97294×10^{-6}	5.77841×10^{-6}	3.35689×10^{-6}	1.95472×10^{-6}	2.75457×10^{-6}
0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000052286	0.0000299855	0.0000172636	9.97294×10^{-6}	5.77841×10^{-6}	3.35689×10^{-6}	4.70929×10^{-6}
0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000052286	0.0000299855	0.0000172636	9.97294×10^{-6}	5.77841×10^{-6}	8.06618×10^{-6}
0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591321	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000052286	0.0000299855	0.0000172636	9.97294×10^{-6}	0.0000138446
0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591321	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000052286	0.0000299855	0.0000172636	0.0000238175
0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591321	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000052286	0.0000299855	0.0000410811
0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591321	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000052286	0.0000710666
0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.0000915857	0.000123353
0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.000161282	0.000214938
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.000510493	0.000285831	0.00037622
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.000510493	0.000662051
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.000920533	0.00117254
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00168035	0.00209308
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00311724	0.00377343
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.00591322	0.00689067
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0115922	0.0128039
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0239806	0.0243961
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.0551627	0.0483766
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.191869	0.103539
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.704591	0.295409
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.