Dynamic server assignment in an extended machine-repair model

EURANDOM PREPRINT SERIES
2012-020
November 22, 2012

Dynamic server assignment in an extended machine-repair model

J.L. Dorsman, S. Bhulai, M. Vlasiou

ISSN 1389-2355

Dynamic server assignment in an extended machine-repair model

J.L. Dorsman [*] [†] S. Bhulai [‡] [†] M. Vlasiou [*] [†]

j.l.dorsman@tue.nl s.bhulai@vu.nl m.vlasiou@tue.nl

November 22, 2012

Abstract

We consider an extension of the classical machine-repair problem. The machines, apart from receiving service from a single repairman, now also supply service themselves to queues of products. The extended model can be viewed as a two-layered queueing network, in which the queues of products in the first layer are generally correlated, due to the fact that the machines have to share the repairman's capacity in the second layer. We are concerned with the dynamic control problem how the repairman should allocate his capacity to the machines at any point in time so that the long-term average (weighted) sum of the queue lengths of the first-layer queues is minimised. Since the optimal policy for the repairman cannot be found analytically due to the correlations in the queue lengths, we propose a near-optimal policy. We do this by combining intuition and results from queueing theory with techniques from Markov decision theory. Specifically, we study the relative value functions for several policies for which the model can be decomposed in less complicated subsystems, and we combine the results with the classical one-step policy improvement algorithm. The resulting policy is easy to apply, scalable in the number of machines and is shown to be highly accurate over a wide range of parameter settings.

1 Introduction

In this paper, we study a queueing model that consists of two layers. The first layer of the model contains two queues of products, see Figure 1. Each of these queues is served by its own machine. At any point in time, a machine is subject to failure, irrespective of the number of products in each of the queues. When a failure occurs, the service of a product in progress is interrupted. Upon failure, the machine temporarily stops fulfilling its server role in the first layer and becomes a customer in the second layer of the model. The second layer of the model consists of a single repairman capable of repairing failed machines. When the repair of a machine in the second layer has finished, the machine assumes its server role in the first layer again.

This model has wide applicability. Evidently, it has immediate applications in manufacturing systems. However, this model is also of interest in less obvious application areas, such as telecommunication systems. For instance, this model occurs naturally in the modelling of middleware technology, where multi-threaded application servers compete for access to shared object code. Access of threads to the object code is typically handled by a portable object adapter (POA) that serialises the threads trying to access a shared object. During the complete duration of the serialisation and execution time a thread typically remains blocked, and upon finalising the execution, the thread is de-activated and ready to process pending requests [12]. In this setting, the application servers and the

Funded in the framework of the STAR-project "Multilayered queueing systems" by the Netherlands Organization for Scientific Research (NWO). The research of M. Vlasiou is also partly supported by an NWO individual grant through project 632.003.002.

[*] EURANDOM and Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

[†] Probability and Stochastic Networks, Centrum Wiskunde & Informatica (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

[‡] Department of Mathematics, VU University Amsterdam, De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

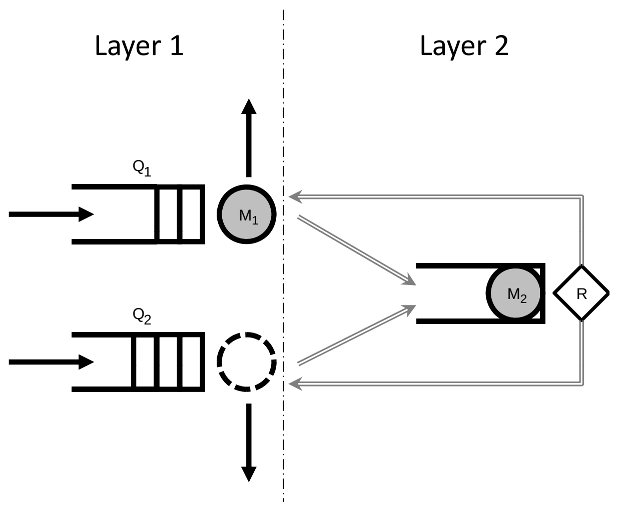

Figure 1: The two-layered model under consideration.

shared object code are analogous to the machines and the repairman. Likewise, several service systems can be modelled in this way.

This model is an example of a *layered queueing network*. In such networks, there exist servers which, while executing a service, may request a lower-layer service and wait for it to be completed. Layered queueing networks occur naturally in information and e-commerce systems, grid systems, and real-time systems such as telecom switches, see [7] and references therein for an overview. A layered queueing network is characterised by simultaneous or separate phases where entities are no longer classified in the traditional role of 'servers' or 'customers', but they may also have a dual role of being either a server to upper-layer entities or a customer to lower-layer entities. Think for example of a peer-to-peer network, where users are both customers when downloading a file, but also servers to users who download their files. In our model, the machines are the entities fulfilling the dual role. They act as a server to the products in the first layer, but may interrupt service to request a second-layer service from the repairman.

The model we consider is also motivated by an extension of the classical machine-repair problem. The machine-repair problem, also known as the *computer terminal model* (cf. [1]) or as the *time sharing system* (cf. [13, Section 4.11]), is a well-studied problem in the literature. In the machine-repair problem, there is a number of machines working in parallel, and one repairman. As soon as a machine fails, it joins a repair queue in order to be repaired by the repairman. It is one of the key models to describe problems with a finite input population. An extensive survey of the machine-repair problem can be found in [11]. We extend this model in two different directions. First, we allow different machines to have mutually different failure or repair rates. As observed in [9], this leads to technical complications. Second, we assume that each of the machines processes a stream of products, which leads to an additional layer in the model.

An important feature of the classical machine-repair problem and the two-layered model under consideration is the fact that the machines have to share the repair facilities offered by the repairman. When both of the machines are down, this may result in the repair of each of the machines occuring at a much slower rate or in one machine having to wait for the other machine to be repaired before its own repair can be started. Because of this, significant positive correlations may arise in the downtimes of the machines. As a consequence of the dual roles of the machines, this leads to correlations between the lengths of the queues of products in the first layer. Due to these correlations, it is extremely hard to obtain expressions for the distribution or even the mean of these queue lengths. Even when machines are repaired in the same order as they break down, the queue lengths are notoriously difficult to analyse (see e.g. [5] or [6]).

We are concerned with the problem how the repairman should allocate his capacity dynamically to the machines at any point in time, given complete information on the lengths of the queues of products and the state of the machines at all times. This must be done in such a way that the long-term average (weighted) sum of the queue lengths of the first-layer queues is minimised. The identification of the repairman's optimal policy is very hard, due to the general intractability of the analysis on the queue lengths because of their interdependence. When formulating this problem as a Markov decision problem, one may be able to obtain the optimal policy numerically for a specific set of parameter settings by truncating the state space. However, these numerical methods lack transparency and provide little insight into the effects of the model's parameters. Moreover, due to the multi-dimensionality of the model, the computation time needed to obtain reliable and accurate results may be infeasibly long.

In this paper, we aim to derive a near-optimal policy for the repairman. A novel point in our technique is that we use insights and results from queueing theory in order to overcome difficulties that appear in this Markov decision problem. We apply the classical approximation method of one-step policy improvement. This method requires a relative value function of an initial policy, which can be obtained analytically by solving the Poisson equations known from standard theory on Markov decision processes. The result is then used in one step of the policy iteration algorithm from Markov decision theory to obtain an improved policy. Although the relative value function of the improved policy is usually hard to compute, the improved policy itself is known explicitly as a function of the state and the parameters of the model. Moreover, it is known that the improved policy performs better in terms of costs than the initial policy, so that it may be used as an approximation for the optimal policy. The intrinsic idea of one-step policy improvement goes back to Norman [15]. Since then, this method has been succesfully applied to derive nearly optimal state-dependent policies in a variety of applications, such as the control of traffic lights [10], production planning [19] and the routing of telephone calls in a network or call center [3, 16, 18]. In our case, we use both a state-independent policy and a priority policy as input for the one-step policy improvement method. Computing the relative value function for the priority policy seems to be intractable. However, we use insights from queueing theory in order to describe the asymptotic behaviour of the relative value function, which we use as input for the one-step policy improvement method. Based on the resulting improved policies, we propose a near-optimal policy for the repairman.

The paper is organised as follows. Section 2 gives a mathematical description of the control problem and introduces the notation required. Although the optimal policy for this control problem cannot be obtained explicitly, several of its structural properties can be derived. As we will see in Section 3, the optimal policy makes the repairman work at full capacity whenever there is at least one machine down, and behaves like a threshold policy. Subsequently, we focus on finding a policy which generally performs nearly as well as the optimal policy. As input for the one-step policy improvement algorithm, we study two policies in Section 4, for which the system decomposes into multiple subsystems, so that the system becomes easier to evaluate. The first of these policies, which we will call the static policy, is state-independent, and always reserves a certain pre-determined fraction of repair capacity to each machine, regardless of the state of the machines. Therefore, the machines behave independently of each other under this policy, which allows us to derive an exact expression for the relative value function. The second policy that we study in Section 4 is the priority policy, in which the repairman always prioritises the repair of a specific machine over the other when both machines are down. Under this policy, the repairman assigns his full capacity to the high-priority machine when it is down, irrespective of the state of the low-priority machine. This makes the system easier to analyse. Nevertheless, it is hard to obtain the relative value function for this policy exactly, but we are able to identify most of its behaviour. Although analytic results on the relative value functions of these policies are of independent interest, we use these results in Section 5 in combination with the one-step policy improvement algorithm. This ultimately results in a well-performing, nearly optimal policy, which is given in terms of a few simple decision rules. The resulting policy turns out to be scalable in the number of machines and corresponding first-layer queues in the model, so that the policy can be readily extended to allow for a number of machines larger than two. Finally, extensive numerical results in Section 6 show that the proposed policy is highly accurate over a wide range of parameter settings. We also identify the key factors determining the performance of the near-optimal policy.

2 Model description and problem formulation

The layered model consists of two machines M_1 and M_2 and a single repairman R, see Figure 1. Each machine M_i serves its own first-layer queue Q_i of products. The products arrive at Q_i according to a Poisson process with rate λ_i. The service requirements of the products in Q_i are exponentially distributed with rate μ_i. The service of these products may be interrupted, since the machines are prone to failure. After an exponentially (σ_i) distributed uptime or lifetime, denoted by U_i, a machine M_i will break down, and the service of Q_i inevitably stops. When this happens, the service of a product in progress is aborted and will be resumed once the machine is operational again. To be returned to an operational state, the machine M_i needs to be repaired by the repairman. Whenever the repairman allocates his full repair capacity to M_i, its repair takes an exponential (ν_i) distributed time. However, the machines share the capacity of the repairman. At any moment in time, the repairman is able to decide how to divide his total repair capacity over the machines. More specifically, it can choose the fractions of capacity q_1 and q_2 that are allocated to the repair of M_1 and M_2 respectively, so that the machines are getting repaired at rate $q_1\nu_1$ and $q_2\nu_2$ respectively. We naturally have that $0 \le q_1 + q_2 \le 1$, and that $q_i = 0$ whenever M_i is operational. The objective of the repairman is to allocate his repair capacity dynamically in such a way that the average long-term weighted number of products in the system is minimised.

In order to describe this dynamic optimisation problem mathematically, one does not only need to keep track of the queues of products, but also of the conditions of the machines. To this end, we define the state space of the system as $\mathcal{S} = \mathbb{N}^2 \times \{0,1\}^2$. Each possible state corresponds to an element $s = (x_1, x_2, w_1, w_2)$ in $\mathcal{S}$, where x_1 and x_2 denote the number of products in Q_1 and Q_2 respectively. The variable w_1 and w_2 represents whether M_1 and M_2 are in an operational (1) or in a failed state (0). The repairman bases his decision on the information $\boldsymbol{s}$, and therefore, any time the state changes can be regarded as a decision epoch. At these epochs, the repairman takes an action $\boldsymbol{a} = (q_1, q_2)$ out of the state-dependent action space $\mathcal{A}_{\boldsymbol{s}} = \{(q_1,q_2) : q_1 \in [0, 1-w_1], q_2 \in [0, 1-w_2], q_1 + q_2 \le 1\}$, where q_i denotes the fraction of capacity assigned to M_i, $i = 1, 2$. The terms $1 - w_1$ and $1 - w_2$ included in the description of the action set enforce the fact that the repairman can only repair a machine if it is down. Now that the states and actions are defined, we introduce the cost structure of the model. The objective is modelled by the cost function $c(\boldsymbol{s}, \boldsymbol{a}) = c_1x_1 + c_2x_2$, where c_1 and c_2 are positive real-valued weights. Thus, when the system is in state $\boldsymbol{s}$, the weighted number of customers present in the system equals $c(\boldsymbol{s}, \cdot)$, regardless of the action $\boldsymbol{a}$ taken by the repairman.

With this description, the control problem can be fully described as a Markov decision problem. To this end we uniformise the system (see e.g. [17, Section 11.5]), i.e., we add dummy transitions (from a state to itself) such that the outgoing rate of every state equals a constant parameter γ, the uniformisation parameter. We choose $\gamma = \lambda_1 + \lambda_2 + \mu_1 + \mu_2 + \sigma_1 + \sigma_2 + \nu_1 + \nu_2$ and we assume that $\gamma = 1$ without loss of generality, since we can always achieve this by scaling the model parameters. Note that this assumption has the intuitive benefit that rates can be considered to be transition probabilities, since the outgoing rates of each state sum up to one. Thus, for $i = 1, 2$, any action $\boldsymbol{a} \in \mathcal{A}_{\boldsymbol{s}}$ and state $\boldsymbol{s} \in \mathcal{S}$, the transition probabilities P are given by

$$\begin{array}{lll}
P_{\boldsymbol{a}}(\boldsymbol{s}, \boldsymbol{s}+\boldsymbol{e}_i) & = \lambda_i, & \text{(product arrivals)} \\
P_{\boldsymbol{a}}(\boldsymbol{s}, \boldsymbol{s}-\boldsymbol{e}_i) & = \mu_i w_i \mathbb{1}_{\{x_i>0\}}, & \text{(product services)} \\
P_{\boldsymbol{a}}(\boldsymbol{s}, \boldsymbol{s}-\boldsymbol{e}_{i+2}) & = \sigma_i w_i, & \text{(machine breakdowns)} \\
P_{\boldsymbol{a}}(\boldsymbol{s}, \boldsymbol{s}+\boldsymbol{e}_{i+2}) & = q_i \nu_i, & \text{(machine repairs)} \\
P_{\boldsymbol{a}}(\boldsymbol{s}, \boldsymbol{s}) & = 1 - \lambda_i - w_i(\mu_i \mathbb{1}_{\{x_i>0\}} + \sigma_i) - q_i\nu_i, & \text{(uniformisation)}
\end{array}$$

where $\mathbb{1}_{\{E\}}$ denotes the indicator function on the event E and $\boldsymbol{e}_i$ represents the unit vector of which the i-th entry equals one. All other transition probabilities are zero. The tuple $(\mathcal{S}, \{\mathcal{A}_{\boldsymbol{s}} : \boldsymbol{s} \in \mathcal{S}\}, P, c)$ now fully defines the Markov decision problem at hand.

Define a deterministic policy π^* as a function from $\mathcal{S}$ to $\mathcal{A}$, i.e., $\pi^*(\boldsymbol{s}) \in \mathcal{A}_{\boldsymbol{s}}$ for all $\boldsymbol{s} \in \mathcal{S}$, and let $\{\boldsymbol{X}^*(t) : t \ge 0\}$ be its corresponding Markov process taking values in $\mathcal{S}$, which describes the state of the system over time when the repairman adheres to policy π^*. Furthermore, let

$$u^*(\boldsymbol{s}, t) = \mathbb{E}[\int_{z=0}^{t} c(\boldsymbol{X}^*(z), \pi^*(\boldsymbol{X}^*(z)))\, dz | \boldsymbol{X}^*(0) = \boldsymbol{s}]$$

denote the total expected costs up to time t when the system starts in state $\boldsymbol{s}$ under policy π^*. Observe that for any stable policy, the Markov process has a single recurrent class, so that the average costs per time unit $g^* = \lim_{t\to\infty} \frac{u^*(\boldsymbol{s},t)}{t}$ is independent of the initial state $\boldsymbol{s}$. This number may also be interpreted as the long-term average sum of queue lengths under policy π, weighted by the constants c_1 and c_2. Any policy π^* can be characterised through its relative value function $V^*(\boldsymbol{s})$. This function is a real-valued function defined on the state space $\mathcal{S}$ given by

$$V^*(\boldsymbol{s}) = \lim_{t\to\infty} (u^*(\boldsymbol{s},t) - u^*(\boldsymbol{s}_{\text{ref}},t)),$$

and represents the asymptotic difference in expected total costs accrued when starting the process in state $\boldsymbol{s}$ instead of some reference state $\boldsymbol{s}_{\text{ref}}$. The optimal policy π^{opt} with relative value function V^{opt} minimises the long-term average weighted sum of queue lengths, thus $g^{opt} = \min_{\pi^*} g^*$. The long-term optimal actions are a solution of the Bellman optimality equations $g^{opt} + V^{opt}(\boldsymbol{s}) = \min_{\boldsymbol{a}\in\mathcal{A}_s}\{c(\boldsymbol{s},\boldsymbol{a}) + \sum_{\boldsymbol{t}\in\mathcal{S}} P_{\boldsymbol{a}}(\boldsymbol{s},\boldsymbol{t})V^{opt}(\boldsymbol{t})\}$ for all $\boldsymbol{s} \in \mathcal{S}$. For our problem, this constitutes

$$g^{opt} + V^{opt}(x_1,x_2,w_1,w_2) = H^{opt}(x_1,x_2,w_1,w_2) + K^{opt}(x_1,x_2,w_1,w_2)$$

for every $(x_1,x_2,w_1,w_2) \in \mathcal{S}$, where H^{opt} and K^{opt} are defined in the following way. For an arbitrary policy π^* with relative value function V^*, the function H^* is given by

$$\begin{aligned} H^*(x_1,x_2,w_1,w_2) =& c_1x_1 + c_2x_2 \\ &+ \lambda_1 V^*(x_1+1,x_2,w_1,w_2) + \lambda_2 V^*(x_1,x_2+1,w_1,w_2) \\ &+ \mu_1 w_1 V^*((x_1-1)^+,x_2,1,w_2) + \mu_2 w_2 V^*(x_1,(x_2-1)^+,w_1,1) \\ &+ \sigma_1 w_1 V^*(x_1,x_2,0,w_2) + \sigma_2 w_2 V^*(x_1,x_2,w_1,0) \\ &+ \Big(1 - \sum_{i=1}^{2}\Big(\lambda_i + w_i(\mu_i+\sigma_i)\Big)\Big)V^*(x_1,x_2,w_1,w_2), \end{aligned} \tag{1}$$

and it models the costs and the action-independent events of product arrivals, product service completions, machine breakdowns and dummy transitions respectively. The function K^* given by

$$\begin{aligned} K^*(x_1,x_2,w_1,w_2) = \min_{(q_1,q_2)\in\mathcal{A}_{(x_1,x_2,w_1,w_2)}} &\{q_1\nu_1(V^*(x_1,x_2,1,w_2) - V^*(x_1,x_2,0,w_2)) \\ &+ q_2\nu_2(V^*(x_1,x_2,w_1,1) - V^*(x_1,x_2,w_1,0))\} \end{aligned} \tag{2}$$

models the optimal state-specific decisions how to allocate the repair capacity over the machines and includes corrections for the uniformisation term.

As already mentioned in Section 1, these equations are exceptionally hard to solve analytically. Alternatively, the optimal actions can also be obtained numerically by recursively defining $V^{n+1}(\boldsymbol{s}) = H^n(\boldsymbol{s}) + K^n(\boldsymbol{s})$ for an arbitrary function V^0. For $n \to \infty$, the minimising actions converge to the optimal ones (see [17, Chapter 8] for existence and convergence properties). We use this procedure called *value iteration* or *successive approximation* for our numerical experiments in Section 6.

3 Structural properties of the optimal policy

As mentioned before, it is hard to give a complete, explicit characterisation of the optimal policy for the problem sketched in Section 2. Therefore, we derive a near-optimal policy later in Section 5. Nevertheless, several important structural properties of the optimal policy can be obtained. It turns out that the optimal policy is work-conserving, always dictates the repairman to work on one machine only and can be classified as a threshold policy. In this section, we inspect these properties more closely.

3.1 Work-conserving property

The optimal policy is work-conserving, which means the repairman always repairs at full capacity whenever a machine is not operational, i.e. $q_1 + q_2 = 1 - w_1 w_2$. Intuitively, this makes sense, as there are no costs involved for the repairman's service. On the other hand, having less repair capacity go unused has decreasing effects on the long-term weighted number of products in the system. There is no trade-off present, and therefore the repair capacity should be used exhaustively whenever there is a machine in need of repair.

This property can be proved mathematically. Note that the minimisers of the right-hand side of (2) represent the optimal actions. From this, it follows that the optimal action satisfies $q_1 + q_2 = 1 - w_1 w_2$ for every state $\boldsymbol{s} \in \mathcal{S}$ (i.e., the optimal policy satisfies the work-conserving property), if both $V^{opt}(x_1, x_2, 0, w_2) - V^{opt}(x_1, x_2, 1, w_2)$ and $V^{opt}(x_1, x_2, w_1, 0) - V^{opt}(x_1, x_2, w_1, 1)$ are non-negative for all $(x_1, x_2, w_1, w_2) \in \mathcal{S}$. The next proposition proves the latter condition. For the sake of reduction of the proof's complexity, it also concerns the trivial fact that under the optimal policy, the system incurs higher costs whenever the number of products in the system is increasing (i.e., $V^{opt}(x_1 + 1, x_2, w_1, w_2) - V^{opt}(x_1, x_2, w_1, w_2)$ and $V^{opt}(x_1, x_2 + 1, w_1, w_2) - V^{opt}(x_1, x_2, w_1, w_2)$ are non-negative).

Proposition 3.1. *The relative value function $V^{opt}(\boldsymbol{s})$ corresponding to the optimal policy satisfies the following properties for all $\boldsymbol{s} \in \mathcal{S}$:*

1. $V^*(x_1, x_2, 0, w_2) - V^*(x_1, x_2, 1, w_2) \geq 0$ *and* $V^*(x_1, x_2, w_1, 0) - V^*(x_1, x_2, w_1, 1) \geq 0$,
2. $V^*(x_1 + 1, x_2, w_1, w_2) - V^*(x_1, x_2, w_1, w_2) \geq 0$ *and* $V^*(x_1, x_2 + 1, w_1, w_2) - V^*(x_1, x_2, w_1, w_2) \geq 0$.

Proof. The proof is based on induction and the guaranteed convergence of the value iteration algorithm. We arbitrarily pick a function $V^0(\boldsymbol{s}) = 0$ for all $\boldsymbol{s} \in \mathcal{S}$. Obviously, this function satisfies properties 1 and 2. We show that these properties are preserved when performing one step of the value iteration algorithm. In mathematical terms, we show for any $n \in \mathbb{N}$ that the function V^{n+1} defined by $V^{n+1}(\boldsymbol{s}) = H^n(\boldsymbol{s}) + K^n(\boldsymbol{s})$ also satisfies the properties if V^n does. Because of the guaranteed convergence, V^{opt} then satisfies properties 1 and 2 by induction. For an extensive discussion of this technique to prove structural properties of relative value functions, see [14].

The induction step is performed as follows. We assume that properties 1 and 2 hold for V^n (the induction assumption). We now show that property 1 holds for V^{n+1}. Observe that by interchanging the indices of the model parameters, one obtains another instance of the same model, since the structure of the model is symmetric. Therefore, the left-hand side of property 1 implies the right-hand side. To prove the first part of property 1, we expand $V^{n+1}(x_1, x_2, 0, w_2) - V^{n+1}(x_1, x_2, 1, w_2)$ into V^n:

$$\begin{aligned}&V^{n+1}(x_1, x_2, 0, w_2) - V^{n+1}(x_1, x_2, 1, w_2)\\ &\quad = H^n(x_1, x_2, 0, w_2) - H^n(x_1, x_2, 1, w_2) + K^n(x_1, x_2, 0, w_2) - K^n(x_1, x_2, 1, w_2).\end{aligned} \tag{3}$$

By rearranging the terms arising from (1) and applying the induction assumption, we have that

$$\begin{aligned}&H^n(x_1, x_2, 0, w_2) - H^n(x_1, x_2, 1, w_2)\\ &\quad \geq (1 - \lambda_1 - \lambda_2 - \sigma_1 - w_2(\mu_2 + \sigma_2))(V^n(x_1, x_2, 0, w_2) - V^n(x_1, x_2, 1, w_2)).\end{aligned} \tag{4}$$

Furthermore, since $V^n(x_1, x_2, w_1, 1) - V^n(x_1, x_2, w_1, 0)$ and $V^n(x_1, x_2, 1, w_2) - V^n(x_1, x_2, 0, w_2)$ are both non-positive numbers, we can limit the set of possible minimising actions in K^n (see (2)) to $\{(q_1, q_2) : q_1 \in \{0, 1 - w_1\}, q_2 \in \{0, 1 - w_2\}, q_1 + q_2 = 1 - w_1 w_2\}$. By this and (2), we obtain

$$\begin{aligned}&K^n(x_1, x_2, 0, w_2) - K^n(x_1, x_2, 1, w_2)\\ &\quad = \min\{\nu_1(V^n(x_1, x_2, 1, w_2) - V^n(x_1, x_2, 0, w_2)),\\ &\qquad\qquad (1 - w_2)\nu_2(V^n(x_1, x_2, 0, 1) - V^n(x_1, x_2, 0, 0))\}\\ &\quad\quad - (1 - w_2)(V^n(x_1, x_2, 1, 1) - V^n(x_1, x_2, 1, 0)),\end{aligned} \tag{5}$$

where the second equality holds because of the induction assumption. Let E_1 denote the event that the last minimum is only minimised by its first argument and let E_2 be its complementary event. As a conclusion, we find by combining (3)–(5) that

$$\begin{aligned}
&V^{n+1}(x_1,x_2,0,w_2) - V^{n+1}(x_1,x_2,1,w_2)\\
&\quad\geq (1-\lambda_1-\lambda_2-\sigma_1-w_2(\mu_2+\sigma_2)-\mathbb{1}_{\{E_1\}}\nu_1-\mathbb{1}_{\{E_2\}}(1-w_2)\nu_2)\\
&\qquad\qquad\qquad (V^n(x_1,x_2,0,w_2)-V^n(x_1,x_2,1,w_2))\\
&\qquad+\mathbb{1}_{\{E_1\}}(1-w_2)(V^n(x_1,x_2,1,0)-V^n(x_1,x_2,1,1))\\
&\qquad+\mathbb{1}_{\{E_2\}}(1-w_2)(V^n(x_1,x_2,0,1)-V^n(x_1,x_2,1,1))\\
&\quad\geq 0.
\end{aligned}$$

The last inequality holds by applying the induction assumption on each term of the expression in front of it and observing, for the first term, that $(1-\lambda_1-\lambda_2-\sigma_1-w_2(\mu_2+\sigma_2)-\mathbb{1}_{\{E_1\}}\nu_1-\mathbb{1}_{\{E_2\}}(1-w_2)\nu_2)$ is non-negative due to uniformisation.

This proves that property 1 holds. By similar techniques of expanding V^{n+1} into V^n and termwise elimination, one can also show that V^{n+1} satisfies property 2 under the induction assumption, which completes the proof. $\square$

By proving that V^{opt} satisfies property 1 as stated in Proposition 3.1, we have established that the optimal policy is work-conserving, implying that $q_1+q_2=1-w_1w_2$ at all times. We finish this section by pointing out that the optimal policy always dictates the repairman to focus all his attention on one machine. That is, at all times, the optimal action reads $(q_1,q_2)=(1-w_1,0)$ or $(q_1,q_2)=(0,1-w_2)$. This is easily derived from (2) in combination with property 1 in Proposition 3.1. Even when there are states for which $w_1w_2=0$ and $\nu_1(V^*(x_1,x_2,1,w_2)-V^*(x_1,x_2,0,w_2))=\nu_2(V^*(x_1,x_2,w_1,1)-V^*(x_1,x_2,w_1,0))$, the actions $(q_1,q_2)=(1-w_1,0)$ and $(q_1,q_2)=(0,1-w_2)$ will be optimal (although they are uniquely optimal), so that there are always optimal policies that concentrate all repair capacity on one machine. Therefore, K^{opt} can be simplified to

$$\begin{aligned}
K^{opt}(x_1,x_2,w_1,w_2)=\min_{(q_1,q_2)\in\{(1-w_1,0),(0,1-w_2)\}}&\{q_1\nu_1(V^{opt}(x_1,x_2,1,w_2)-V^{opt}(x_1,x_2,0,w_2))\\
&+q_2\nu_2(V^{opt}(x_1,x_2,w_1,1)-V^{opt}(x_1,x_2,w_1,0))\}.
\end{aligned}\tag{6}$$

This is a welcome simplification when one wants to evaluate the optimal policy numerically, since now the minimum-operator only involves two arguments.

3.2 Threshold policy

Now that we know that the optimal policy is work-conserving and always dictates the repairman to focus his attention on a single machine, the question arises which machine this should be. In the event both machines are down, this question is hard to answer explicitly, since the relative value function V^{opt} pertaining to the optimal policy defies an exact analysis. However, by inspection of numerical results, one can derive a partial answer.

To this end, we numerically examine the model with the settings $c_1=c_2=\mu_2=\sigma_1=\nu_1$ = 1.0, $\lambda_1=0.1$, $\lambda_2=0.2$ and $\mu_1=\sigma_2=\nu_2$ = 0.5. By using the simplified version (6) of K^{opt} in the value iteration algorithm, we numerically obtain the optimal actions for the states $(x_1,x_2,0,0)$, $x_1\in\{0,\ldots,50\}$, $x_2\in\{0,\ldots,100\}$. Figure 2 shows the optimal actions in the form of a scatter plot. Given that both machines are down, a marked point (x_1,x_2) in the scatter plot indicates that it is optimal for the repairman to repair M_2. If a certain point (x_1,x_2) is not marked, then the optimal action is to repair M_1 at full capacity.

Judging by Figure 2, it seems that the optimal policy falls in the class of threshold policies. That is, if the optimal action for the state $(x_1,x_2,0,0)$ is to repair M_1 at full capacity, then this is also the optimal action for the states $(x_1+k,x_2,0,0)$, $k\in\mathbb{N}$. Meanwhile, if it would be optimal to repair M_2 when the system is in the state $(x_1,x_2,0,0)$, then the optimal policy also prescribes to repair M_2 if there are fewer products waiting in Q_1, i.e., in the states $(x_1-k,x_2,0,0)$, $k\in\{0,\ldots,x_1\}$. Thus, for any number x_2, the number of products in Q_1 from which the optimal policy starts taking the decision to repair M_1 can be seen as a threshold. Similar effects and definitions

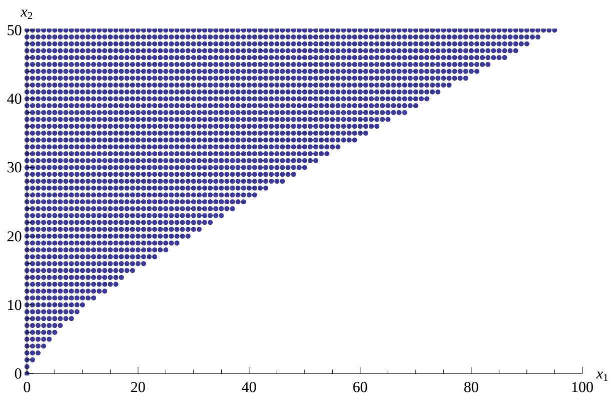

Figure 2: The optimal actions for the model instance studied in Section 3.2.

apply for varying number of products in Q_2. The figure clearly exposes a curve that marks the thresholds. At first glance, this threshold curve may seem linear. However, especially near the origin, this is not quite true.

One can reason intuitively that for any instance of the model, the optimal policy is a threshold policy. This is easily understood by the notion that an increasing number of products in Q_1 makes it more attractive for the repairman to repair M_1. Then, if it was already optimal to repair M_1, this obviously will not change. Similar notions exist for a decreasing number of products in Q_1 and varying numbers of products in Q_2. Although the threshold effects are easily understood, they are hard to prove mathematically. A possible approach to this would be to show that the difference between the arguments in (6) is increasing in x_1 using the same techniques as used in the proof of Proposition 3.1. However, this turns out to be very challenging.

4 Relative value functions

Recall that for any policy π^*, we defined V^* and g^* to be its corresponding relative value function and long-run expected weighted number of products in the system respectively. The main reason why one cannot obtain the optimal policy π^{opt} other than through numerical means, is because its corresponding relative value function V^{opt} does not allow for an exact analysis. As an intermediate step, we therefore study the relative value functions of two other policies, for which explicit expressions can be obtained. We first examine the static policy in Section 4.1, where each machine is assigned a fixed part of the repair capacity regardless of the system's state. Then, we observe the priority policy in Section 4.2, which dictates the repairman to prioritise a specific machine (the high-priority machine), in case both machines are not operational; i.e., in such a case, all repair capacity is given to the high-priority machine. In Section 5, these two policies and their relative value functions act as a basis for the one-step policy improvement to obtain nearly optimal heuristic policies.

4.1 Static policy

As the name of the static policy suggests, the actions taken under this policy do not depend on the state the system is in. Under the static policy, the repairman always has a fraction $p \in (0, 1)$ of his repair capacity reserved for the repair of M_1, regardless of whether M_1 (or M_2) is down or not. Likewise, the remaining fraction $(1-p)$ is reserved for M_2. Therefore, repair on M_1 at rate $p\nu_1$ starts instantly the moment it breaks down, and the same holds for M_2 at rate $(1-p)\nu_2$. Thus, under this policy, the repairman always takes the action $(p(1-w_1), (1-p)(1-w_2))$. In the sequel, we will refer to p as the splitting parameter. It is evident that this policy is not optimal, since the repairman does not use his repair capacity exhaustively when exactly one of the two machines is down, i.e., the static policy does not satisfy the work-conserving property studied in Section 3.1. However, when the splitting parameter is chosen well, this policy is not totally unreasonable either. When analysing this policy, we assume that the system is stable when adhering to it. That is, for each queue the rate of arriving products is smaller than the rate at which the corresponding machine is capable of serving products:

$$\lambda_1 < \mu_1 \frac{p\nu_1}{\sigma_1 + p\nu_1} \text{ and } \lambda_2 < \mu_2 \frac{(1-p)\nu_2}{\sigma_2 + (1-p)\nu_2}, \tag{7}$$

where the two fractions denote the fractions of time M_1 and M_2 are operational respectively.

Observe that the capacity that M_1 receives from the repairman is now completely independent of that received by M_2 at any given time, and vice versa. Analysis of the relative value function of the static policy is tractable, since the machines do not compete for repair resources anymore under this policy, making the queue lengths in each of the queues uncorrelated. In a way, it is as if each machine has its own repairman now, repairing at rate $p\nu_1$ and $(1-p)\nu_2$ respectively. Therefore, the system can be decomposed into two components, which do not interact. Each of these components can be modelled as a single-server queue of M/M/1 type with server vacations occuring independently of the amount of work present in the queue. Because of this decomposition, the relative value function $V^{sta}(x_1, x_2, w_1, w_2)$ of the total system can be seen as the weighted sum of the relative value functions $V_1^{com}(x_1, w_1)$ and $V_2^{com}(x_2, w_2)$ corresponding to the two components. As a result, the long-term average cost g^{sta} is also a weighted sum of the average costs g_1^{com} and g_2^{com}:

$$g^{sta} = c_1 g_1^{com} + c_2 g_2^{com} \text{ and } V^{sta}(x_1, x_2, w_1, w_2) = c_1 V_1^{com}(x_1, w_1) + c_2 V_2^{com}(x_2, w_2). \tag{8}$$

To derive g_1^{com}, g_2^{com}, $V_1^{com}(x_1, w_1)$ and $V_2^{com}(x_2, w_2)$, we focus on the relative value function corresponding to one component in Section 4.1.1. We then finalise the analysis on V^{sta} in Section 4.1.2.

4.1.1 Relative value function for the components

We now derive the relative value function of one component of the model under the static policy, and omit all indices of the parameters. Thus, we regard a single-server queue of M/M/1 type, in which products arrive at rate λ and are processed at rate μ if the machine is up. Independently of this process, the server takes a vacation after an exponentially (σ) distributed amount of time, even when there is a product in service. The service of the product is then interrupted and resumed once the server ends its vacation. A vacation takes an exponentially (ν) distributed amount of time, after which the server will process products again until the next vacation. This system can be interpreted as a Markov reward chain with states $(x, w) \in \mathcal{S}^{com}$ representing the products present in the system and the state of the server being in a vacation ($w = 0$) or not ($w = 1$), where $\mathcal{S}^{com} = \mathbb{N} \times \{0, 1\}$ is its state space. The system is said to accrue costs state-dependently at rate $c(x, w) = x$ per time unit. After uniformisation at rate one, the transition probabilities $P^{com}(\boldsymbol{s}, \boldsymbol{t})$ from a state $\boldsymbol{s} \in \mathcal{S}^{com}$ to a state $\boldsymbol{t} \in \mathcal{S}^{com}$ are given by

$$\begin{aligned}
&P^{com}((x,w),(x+1,w)) = \lambda, && P^{com}((x,w),(x-1,w)) = \mu w \mathbb{1}_{\{x>0\}},\\
&P^{com}((x,1),(x,0)) = \sigma, && P^{com}((x,0),(x,1)) = \nu,\\
&\text{and} && P^{com}((x,w),(x,w)) = (1 - \lambda - w(\mu \mathbb{1}_{\{x>0\}} + \sigma) + \nu(1-w)),
\end{aligned}$$

while all other transition probabilities are zero. By the above description, the Poisson equations for this Markov reward chain with long-term average costs per time unit g^{com} and relative value function $V^{com}(x, w)$ are given by

$$\begin{aligned}
g^{com} + V^{com}(x,w) =& x + \lambda V^{com}(x+1,w) + \mu w V^{com}((x-1)^+, w)\\
&+ \sigma w V^{com}(x,0) + \nu(1-w) V^{com}(x,1)\\
&+ (1 - \lambda - w(\mu+\sigma) - \nu(1-w)) V^{com}(x,w)
\end{aligned} \tag{9}$$

for all $(x, w) \in \mathbb{N} \times \{0, 1\}$.

To solve these equations, we first observe that the completion time for a product from the moment its service is started until it leaves the system consists of an exponentially (μ) distributed amount of actual service time and possibly some interruption time due to server vacations. When interruption takes place, the number of interruptions is geometrically ($\frac{\mu}{\mu+\sigma}$) distributed, due to the Markovian nature of the model. Combined with the fact that every interruption takes an exponential (ν) amount of time, this means that the total interruption time, given that it is positive, is exponentially ($\frac{\mu\nu}{\mu+\sigma}$) distributed. Thus, the completion time consists of an exponential (ν) repair phase and also, with a probability $\frac{\sigma}{\mu+\sigma}$ that there is at least one interruption, an exponential ($\frac{\mu\nu}{\mu+\sigma}$) interruption phase. The above implies that the distribution of the completion time falls in the class of Coxian distributions with order 2. Due to this observation, the average costs per time unit g^{com} incurred by a component can be calculated by the use of standard queueing theory, see Remark 4.2. However, we are also interested in the relative value function of the component. If the server would only start a vacation if there is at least one product in the queue, the component could in principle be modelled as an M/Cox(2)/1 queue, by incorporating the interruption times into the service times (that is, by replacing the service times with the completion times). For the M/Cox(2)/1 queue, it is known that the relative value function can be expressed as a second-order polynomial in the queue length (cf. [2]). However, in our case, a server may also start a vacation during an idle period, so that products arriving at an empty system may not be served instantly. Nevertheless, it is reasonable to conjecture that the relative value function V^{com} is of a second-order polynomial type too.

If this conjecture holds, substituting $V^{com}(x,0) = \alpha_1 x^2 + \alpha_2 x + \alpha_3$ and $V^{com}(x,1) = \beta_1 x^2 + \beta_2 x + \beta_3$ in (9) should lead to a consistent system of equations and give a solution for the coefficients. After substitution, we find the equations

$$\begin{aligned}
g^{com} + \alpha_3 &= \lambda(\alpha_1+\alpha_2) + (1-\nu)\alpha_3 + \nu\beta_3,\\
g^{com} + \beta_3 &= \sigma\alpha_3 + \lambda(\beta_1+\beta_2) + (1-\sigma)\beta_3,\\
g^{com} + \alpha_1 x^2 + \alpha_2 x + \alpha_3 &= ((1-\nu)\alpha_1 + \nu\beta_1)x^2 + (1 + 2\lambda\alpha_1 + (1-\nu)\alpha_2 + \nu\beta_2)x\\
&\quad + \lambda(\alpha_1+\alpha_2) + (1-\nu)\alpha_3 + \nu\beta_3,\\
g^{com} + \beta_1 x^2 + \beta_2 x + \beta_3 &= (\sigma\alpha_1 + (1-\sigma)\beta_1)x^2 + (1 + \sigma\alpha_2 + 2(\lambda-\mu)\beta_1 + (1-\sigma)\beta_2)x\\
&\quad + \sigma\alpha_3 + (\lambda+\mu)\beta_1 + (\lambda-\mu)\beta_2 + (1-\sigma)\beta_3
\end{aligned}$$

for all $x \in \mathbb{N}_+$. One can easily verify that the system of equations is indeed consistent. By solving for the coefficients, a solution for g^{com} and V^{com} up to a constant can be found. The constant can be chosen arbitrarily, e.g. by assuming that $V^{com}(0,1) = 0$, but is of no importance. In principle, there may exist other solutions to (9) that do not behave like a second-order polynomial in x. In fact, when the state space is not finite, as is the case in our model, it is known that there are many pairs of g and V that satisfy the Poisson equations (9) (see e.g. [4]). There is only one stable pair satisfying $V(0,1) = 0$ that is the correct stable solution, however, and we refer to this as the unique solution. Showing that a solution to (9) is the unique solution involves the construction of a weighted norm so that the Markov chain is geometrically recurrent with respect to that norm. This weighted norm imposes extra conditions on the solution to the Poisson equations, so that the unique solution can be identified. The next lemma summarises the solution resulting from the set of equations above, and states that this is also the unique solution.

Lemma 4.1. *For a stable component instance, the long-term average number of products g^{com} and the relative value function V^{com} are given by*

$$g^{com} = \frac{\lambda((\sigma+\nu)^2 + \mu\sigma)}{(\sigma+\nu)(\mu\nu - \lambda(\sigma+\nu))}, V^{com}(x,0) = \alpha_1 x^2 + \alpha_2 x + \alpha_3 \text{ and } V^{com}(x,1) = \alpha_1 x^2 + \alpha_1 x, \quad (10)$$

for $x \in \mathbb{N}$, where

$$\alpha_1 = \frac{\sigma+\nu}{2(\mu\nu - \lambda(\sigma+\nu))}, \alpha_2 = \frac{2\mu+\sigma+\nu}{2(\mu\nu - \lambda(\sigma+\nu))} \text{ and } \alpha_3 = \frac{\lambda\mu}{(\mu\nu - \lambda(\sigma+\nu))(\sigma+\nu)},$$

when taking $V^{com}(0,1) = 0$ as a reference value.

Proof. One simply verifies by substitution that the solution given in (10) satisfies $V^{com}(0,1) = 0$ and the Poisson equations in (9). It is left to show that the above solution is the unique solution. To this end, we use [4, Theorem 6]. Suppose that there exists a finite subset of states M and a weight function $u : \mathcal{S}^{com} \to \{0,1\}$, such that the Markov chain, which satisfies the stability and aperiodicity conditions needed for the theorem to hold, is u-geometrically recurrent, i.e.,

$$R_{M,u}(x,w) := \sum_{(x',w')\notin M} \frac{P^{com}((x,w),(x',w'))u(x',w')}{u(x,w)} < 1$$

for all $(x, w) \in \mathcal{S}$ and

$$||c||_u = \sup_{\boldsymbol{s} \in \mathcal{S}^{com}} \frac{|c(\boldsymbol{s})|}{u(\boldsymbol{s})} < \infty.$$

Then, this theorem implies that a pair (g, V) satisfying the Poisson equations (9) is the unique solution when

$$||V||_u = \sup_{\boldsymbol{s} \in \mathcal{S}^{com}} \frac{|V(\boldsymbol{s})|}{u(\boldsymbol{s})} < \infty.$$

To invoke this theorem, we set $M = \{(0,0), (0,1)\}$ and $u(x, w) = (1+\delta)^x (1-\epsilon)^w$, with

$$\delta \in \Big(0, \frac{\mu + \nu + \sigma - \sqrt{(\lambda - \mu - \nu - \sigma)^2 + 4\,(\lambda\nu - \mu\nu + \lambda\sigma)}}{2\lambda} - \frac{1}{2}\Big)$$

and

$$\epsilon \in \Big(\frac{\lambda}{\nu}\delta, \frac{\delta\mu - \lambda\delta(1+\delta)}{\delta\mu - \lambda\delta(1+\delta) + \sigma(1+\delta)}\Big).$$

Then, we have that

$$R_{M,u}(x, w) = \lambda(1+\delta) + w(\mu \mathbb{1}_{\{x>1\}} \frac{1}{1+\delta} + \sigma \frac{1}{1-\epsilon}) + \nu(1-w)(1-\epsilon) + (1 - \lambda - w(\mathbb{1}_{\{x>1\}}\mu + \sigma) - (1-w)\nu).$$

For all $x \in \mathbb{N}$, the lower bound on ϵ ensures that $R_{M,u}(x, 0) < 1$, while the upper bound guarantees that $R_{M,u}(x, 1) < 1$. The upper bound of δ is derived by equating the two bounds of ϵ, and thus warrants that the lower bound of ϵ does not exceed the upper bound of ϵ. In its turn, the stability condition $\lambda < \mu\frac{\nu}{\sigma+\nu}$ (see (7)) guarantees that the upper bound of δ is positive. Observe that for the assessment of the validity of the conditions $||c||_u < \infty$ and $||V^{com}||_u < \infty$, the value of w does not play an essential role, as it can only influence the value of $u(x, w)$ up to a finite factor $(1 - \epsilon)$ for any $x \in \mathbb{N}$. We clearly have that the cost function $c(x, w) = x$ satisfies $||c||_u < \infty$, since it is linear in x, and the weight function u is exponential in x. Likewise, the function V^{com} as given in (10), satisfies $||V^{com}||_u < \infty$, since it behaves as a quadratic polynomial as opposed to exponential in x. Hence, by [4, Theorem 6] the solution given by (10) is the unique solution to the Poisson equations. $\square$

This concludes the derivation of the relative value function for a component with parameters λ, μ, σ and ν.

Remark 4.1. For $\sigma = 0$ and $w = 1$, the component model degenerates to a regular M/M/1 queue. As expected, g^{com} and $V^{com}(x, 1)$ then simplify to the well-known expressions $g^{M/M/1} = \frac{\lambda}{\mu - \lambda}$ and $V^{M/M/1}(x) = \frac{1}{2(\mu-\lambda)} x(x+1)$. For the general case, we may rewrite $V^{com}(x, 1) = \frac{1}{2(\mu\frac{\nu}{\sigma+\nu} - \lambda)} x(x+1)$. Observe that $\mu\frac{\nu}{\sigma+\nu}$ is the maximum rate at which the server is able to process products in the long term. When interpreting this as an effective service rate, we may conclude that the structure of the relative value function V^{com} is similar to that of the regular M/M/1 queue.

Remark 4.2. As observed above, the component can be modelled alternatively as a single-server vacation queue with the Coxian completion time C of a product regarded as the service time and with server vacations occuring exclusively when the queue is empty. As a result, the average costs per time unit, or rather, the average queue length g^{com} (including any possible product in service) can also be obtained by applying the Fuhrmann-Cooper decomposition (cf. [8]):

$$g^{com} = \mathbb{E}[V] + \mathbb{E}[L_{\text{M/Cox/1}}],$$

where $\mathbb{E}[V]$ represents the expected queue length when observed during a server vacation (at the start of which there are no products in the queue) and $\mathbb{E}[L_{\text{M/Cox/1}}] = \lambda\mathbb{E}[C] + \frac{\lambda\mathbb{E}[C^2]}{2(1-\lambda\mathbb{E}[C])}$ is the well-known expectation of the queue length $L_{\text{M/Cox/1}}$ of a similar single-server queueing system with Poisson (λ) arrivals and the completion times as service times, but excluding any server vacations. The term $\mathbb{E}[V]$ equals the probability $\frac{\sigma}{\sigma+\nu}$ that a product arriving in an empty system finds the server in a vacation, times the mean number of Poisson (λ) arrivals during a residual exponentially (ν) distributed vacation time, and thus amounts to $\frac{\lambda\sigma}{\nu(\sigma+\nu)}$. The moments $\mathbb{E}[C]$ and $\mathbb{E}[C^2]$ can be determined by studying the relation between the completion time and the service requirement of a product. When substituting these expressions, we obtain g^{com} as given in Lemma 4.1.

4.1.2 Resulting expression for V^{sta}

We now turn back to the relative value function of the model as described in Section 2 under the static policy with parameter p. As mentioned before, this model consists of two components with rates $\lambda_1, \mu_1, \sigma_1, p\nu_1$ and $\lambda_2, \mu_2, \sigma_2, (1-p)\nu_2$ respectively. Now that we have found an expression for the relative value functions pertaining to one such component, we readily obtain an expression for the relative value function for the complete system. Combining (8) with Lemma 4.1 results in the following theorem.

Theorem 4.2. *Given that the stability conditions in* (7) *are satisfied, the long-term average costs g_p^{sta} and the relative value function $V_p^{sta}(x_1, x_2, w_1, w_2)$ corresponding to the static policy with parameter p are given by*

$$g_p^{sta} = c_1 \frac{\lambda_1((\sigma_1 + p\nu_1)^2 + \mu_1\sigma_1)}{(\sigma + p\nu_1)(\mu_1 p\nu_1 - \lambda_1(\sigma_1 + p\nu_1))} + c_2 \frac{\lambda_2((\sigma_2 + (1-p)\nu_2)^2 + \mu_2\sigma_2)}{(\sigma_2 + (1-p)\nu_2)(\mu_2(1-p)\nu_2 - \lambda_2(\sigma_2 + (1-p)\nu_2))}$$

and

$$\begin{aligned} V_p^{sta}(x_1, x_2, w_1, w_2) =& \alpha_{1,1}c_1x_1^2 + c_1(\alpha_{2,1}(1-w_1) + \alpha_{1,1}w_1)x_1 + \alpha_{3,1}c_1(1-w_1) \\ &+ \alpha_{1,2}c_2x_2^2 + c_2(\alpha_{2,2}(1-w_2) + \alpha_{1,2}w_2)x_2 + \alpha_{3,2}c_2(1-w_2) \end{aligned}$$

for all $(x_1, x_2, w_1, w_2) \in \mathcal{S}$, where

$$\begin{aligned} \alpha_{1,1} &= \frac{\sigma_1 + p\nu_1}{2\mu_1 p\nu_1 - \lambda_1(\sigma_1 + p\nu_1)}, \alpha_{1,2} = \frac{\sigma_2 + (1-p)\nu_2}{2\mu_2(1-p)\nu_2 - \lambda_2(\sigma_2 + (1-p)\nu_2)}, \\ \alpha_{2,1} &= \frac{2\mu_1 + \sigma_1 + p\nu_1}{2\mu_1 p\nu_1 - \lambda_1(\sigma_1 + p\nu_1)}, \alpha_{2,2} = \frac{2\mu_2 + \sigma_2 + (1-p)\nu_2}{2\mu_2(1-p)\nu_2 - \lambda_2(\sigma_2 + (1-p)\nu_2)}, \\ \alpha_{3,1} &= \frac{\lambda_1\mu_1}{(\mu_1 p\nu_1 - \lambda_1(\sigma_1 + p\nu_1))(\sigma_1 + p\nu_1)}, \textit{ and} \\ \alpha_{3,2} &= \frac{\lambda_2\mu_2}{(\mu_2(1-p)\nu_2 - \lambda_2(\sigma_2 + (1-p)\nu_2))(\sigma_2 + (1-p)\nu_2)}. \end{aligned}$$

4.2 Priority policy

In the previous section, we have derived the relative value function for the static policy explicitly. In Section 5, this policy will act as an initial policy for the one-step policy improvement algorithm to obtain a well-performing heuristic policy. However, for certain instances of the model, there may be no static policy available for which stability holds, whereas the optimal policy does result in stable queues. Then, one-step policy improvement based on the static policy is not feasible, since the initial policy for this procedure must result in a stable system. In these cases, a priority policy may still result in stability and thus be suitable as an initial policy, so that a heuristic policy can still be obtained. For this reason, we study the relative value function V^{prio} of the priority policy in the current section.

Under the priority policy π^{prio}, the repairman always prioritises a specific machine, which we will call the high-priority machine. This means that in case both machines are down, the repairman allocates his full capacity to the high-priority machine. If there is only one machine unoperational, the repairman dedicates his capacity to the broken machine, regardless of whether it is the high-priority machine. In case all machines are operational, the repairman obviously remains idle. Without loss of generality, we assume in this section that M_1 is the high-priority machine. Thus, the repairman always takes the action $((1-w_1), w_1(1-w_2))$. In the remainder of this section, we also assume the system to be stable. That is, for each queue the rate at which products arrive is smaller than the effective service rate of its machine under the priority policy:

$$\lambda_1 < \mu_1 \frac{\nu_1}{\sigma_1 + \nu_1} \text{ and } \lambda_2 < \mu_2^{eff}, \tag{11}$$

where μ_2^{eff} refers to the effective service rate of M_2. The right-hand side of the first inequality represents the effective service rate of the high-priority machine M_1, and consists of the actual service rate μ_1 times the fraction of time M_1 is operational under the priority policy. The effective service rate of M_2 analogously satisfies

$$\mu_2^{\mathit{eff}} = \mu_2 \frac{\frac{1}{\sigma_2}}{\frac{1}{\sigma_2} + \frac{z}{\nu_1} + \mathbb{E}[R_2]}. \tag{12}$$

The expression $\frac{z}{\nu_1} + \mathbb{E}[R_2]$ in the right-hand side represents the expected downtime of M_2. The constant z refers to the probability that M_2 observes the repairman busy on M_1 when it breaks down, so that $\frac{z}{\nu_1}$ represents the expected time M_2 has to wait after its breakdown until the start of its repair as a result of an M_1 failure. The probability z is computed by the fixed-point equation

$$z = \frac{\sigma_1}{\sigma_1 + \sigma_2}\Big(\frac{\sigma_2}{\nu_1 + \sigma_2} + \frac{\nu_1}{\nu_1 + \sigma_2} z\Big) \quad \Rightarrow \quad z = \frac{\sigma_1}{\sigma_1 + \sigma_2 + \nu_1}. \tag{13}$$

Likewise, $\mathbb{E}[R_2]$ represents the expected time from the moment the repairman starts repair on M_2 until its finish, and is computed by the fixed-point equation

$$\mathbb{E}[R_2] = \frac{1}{\sigma_1 + \nu_2} + \frac{\sigma_1}{\sigma_1 + \nu_2}\Big(\frac{1}{\nu_1} + \mathbb{E}[R_2]\Big) \quad \Rightarrow \quad \mathbb{E}[R_2] = \frac{1}{\nu_2} + \frac{\sigma_1}{\nu_1 \nu_2}.$$

Deriving an expression for the relative value function V^{prio} of the priority policy is hard. Before, in the case of the static policy, the model could be decomposed into several components which exhibit no interdependence. This allowed us to obtain an explicit expression for V^{sta}. In contrast, a similar decomposition under the current policy does lead to *interacting* components. The first component, which contains the high-priority machine M_1 and its corresponding queue, acts independently of any other component, since M_1 is not affected by M_2 when accessing repair resources. However, M_2 is affected by M_1. This interference causes the second component, which contains the other machine and its queue of products, to become dependent on the events occurring in the first component. Therefore, there exist correlations, which make an explicit analysis of V^{prio} intractable. Nevertheless, we are still able to derive certain characteristics of the relative value function.

When decomposing the model in the same way as was done in Section 4.1, we have, similar to (8), that the long-term average costs g^{prio} per time unit and the relative value function V^{prio} pertaining to the priority policy can be written as

$$g^{prio} = c_1 g^{prc} + c_2 g^{nprc} \text{ and } V^{prio}(x_1, x_2, w_1, w_2) = c_1 V^{prc}(x_1, w_1) + c_2 V^{nprc}(x_2, w_1, w_2), \tag{14}$$

where g^{prc} and $V^{prc}(x_1, w_1)$ are the long-term average costs and the relative value function pertaining to the first component, which we will also call the priority component. Similarly, g^{nprc} and $V^{nprc}(x_2, w_1, w_2)$ denote the long-term average costs and the relative value function of the second component, which we will also refer to as the non-priority component. In both of these subsystems, the products present are each assumed to incur costs at rate one. Note that the function $V^{nprc}(x_2, w_1, w_2)$ of the second component now includes w_1 as an argument, since the costs accrued in the second component are now dependent on the state of M_1 in the first component. In Section 4.2.1, we obtain an expression for V^{prc}. While V^{nprc} defies an explicit analysis due to the aforementioned dependence, we make several conjectures on its form in Section 4.2.2. In Section 5, it will turn out that these conjectures still allow us to use π^{prio} as an initial policy for the one-step improvement algorithm.

4.2.1 Relative value function for the priority component

In the priority component, the machine M_1 faces no competition in accessing repair facilities. If M_1 breaks down, the repairman immediately starts repairing M_1 at rate ν_1. Thus, from the point of view of M_1, it is as if M_1 has its own dedicated repairman. Therefore, the priority component behaves completely similar to a component of the static policy studied in Section 4.1.1, but now with $\lambda_1, \mu_1, \sigma_1$ and ν_1 as product arrival, product service, machine breakdown and machine repair rates. As a result, we obtain by Lemma 4.1 that, when products in the queue incur costs at rate one, the long-term average costs g^{prc} and the relative value function V^{prc} are given by

$$g^{prc} = \frac{\lambda_1((\sigma_1 + \nu_1)^2 + \mu_1 \sigma_1)}{(\sigma_1 + \nu_1)(\mu_1 \nu_1 - \lambda_1(\sigma_1 + \nu_1))}, V^{prc}(x_1, 0) = \upsilon_1 x_1^2 + \upsilon_2 x_1 + \upsilon_3 \text{ and } V^{prc}(x_1, 1) = \upsilon_1 x_1^2 + \upsilon_1 x_1, \tag{15}$$

for $x_1 \in \mathbb{N}$, where

$$\upsilon_1 = \frac{\sigma_1 + \nu_1}{2(\mu_1\nu_1 - \lambda_1(\sigma_1 + \nu_1))}, \upsilon_2 = \frac{2\mu_1 + \sigma_1 + \nu_1}{2(\mu_1\nu_1 - \lambda_1(\sigma_1 + \nu_1))} \text{ and } \upsilon_3 = \frac{\lambda_1\mu_1}{(\mu_1\nu_1 - \lambda_1(\sigma_1 + \nu_1))(\sigma_1 + \nu_1)},$$

when taking $V^{prc}(0, 1) = 0$ as a reference value.

4.2.2 Relative value function for the non-priority component

As mentioned earlier, the relative value function V^{nprc} of the non-priority component defies an explicit analysis, due to its dependence on the priority component. In this section, we conjecture that V^{nprc} asymptotically behaves like a second-order polynomial in x_2 as $x_2 \to \infty$. Building on this, we also make certain conjectures on the first-order and second-order coefficients of this polynomial. The plausibility of the conjectures and claims presented in this section were verified by numerical results. For the sake of brevity, we omit the presentation of this numerical study. We also present an approximation for the long-term expected costs g^{nprc}.

In the non-priority component, products arrive at rate λ_2, and are served at rate μ_2 by M_2 when it is operational. Independently of this, M_2 breaks down at rate σ_2 when it is operational. In case M_2 is down, it gets repaired at rate ν_2 if M_1 is operational, and at rate zero otherwise. Obviously, if M_1 is operational, it breaks down at rate σ_1, and gets repaired at rate ν_1 if it is down. The resulting system can again be formulated as a Markov reward chain with states $(x_2, w_1, w_2) \in \mathcal{S}^{nprc}$, representing the number of products in the component (x_2), and the indicator variables corresponding to each of the machine's operational states (w_1, w_2), where $S^{nprc} \in \mathbb{N} \times \{0, 1\}^2$ is its state space. This chain is said to accrue costs state-dependently at rate $c(x_2, w_1, w_2) = x_2$. After uniformisation at rate 1, the transition probabilities $P^{nprc}(\boldsymbol{s}, \boldsymbol{t})$ from a state $\boldsymbol{s} \in \mathcal{S}^{nprc}$ to a state $\boldsymbol{t} \in \mathcal{S}^{nprc}$ are given by

$$\begin{aligned}
&P^{nprc}((x_2, w_1, w_2), (x_2 + 1, w_1, w_2)) = \lambda_2, \quad && P^{nprc}((x_2, w_1, w_2), (x_2 - 1, w_1, w_2)) = \mu_2 w_2 \mathbb{1}_{\{x_2>0\}},\\
&P^{nprc}((x_2, 1, w_2), (x_2, 0, w_2)) = \sigma_1, && P^{nprc}((x_2, w_1, 1), (x_2, w_1, 0)) = \sigma_2,\\
&P^{nprc}((x_2, 0, w_2), (x_2, 1, w_2)) = \nu_1, && P^{nprc}((x_2, 1, 0), (x_2, 1, 1)) = \nu_2, \text{ and}\\
&P^{nprc}((x_2, w_1, w_2), (x_2, w_1, w_2)) = (1 - \lambda_2 - \sigma_1 w_1 - w_2(\mu_2 \mathbb{1}_{\{x_2>0\}} + \sigma_2) - \nu_1(1 - w_1) - \nu_2 w_1(1 - w_2)),
\end{aligned}$$

while all other transition probabilities are zero. For this Markov reward chain, the Poisson equations are given by

$$\begin{aligned}
g^{nprc} &+ V^{nprc}(x_2, w_1, w_2)\\
&= x_2 + \lambda_2 V^{nprc}(x_2 + 1, w_1, w_1) + \mu_2 w_2 V^{nprc}((x_2 - 1)^+, w_1, 1)\\
&\quad + \sigma_1 w_1 V^{nprc}(x_2, 0, w_2) + \sigma_2 w_2 V^{nprc}(x_2, w_1, 0)\\
&\quad + \nu_1(1 - w_1) V^{nprc}(x_2, 1, w_2) + \nu_2 w_1(1 - w_2) V^{nprc}(x_2, 1, 1)\\
&\quad + (1 - \lambda_2 - \sigma_1 w_1 - w_2(\mu_2 + \sigma_2) - \nu_1(1 - w_1) - \nu_2 w_1(1 - w_2)) V^{nprc}(x_2, w_1, w_2). \quad (16)
\end{aligned}$$

Conjecture 4.3. *Assume that the stability conditions in* (11) *are satisfied. Then, the relative value function* $V^{nprc}(x_2, w_1, w_2)$ *of the non-priority component asymptotically behaves as a second-order polynomial in* x_2 *with second-order coefficient* $\phi_1 = \frac{1}{2}(\mu_2^{\mathit{eff}} - \lambda_2)^{-1}$ *as* $x_2 \to \infty$ *for each* $w_1, w_2 \in \{0, 1\}$*, where* μ_2^{eff} *represents the effective service rate of* M_2 *as given in* (12).

Argument. Recall that $V^{nprc}(x_2 + 1, w_1, w_2) - V^{nprc}(x_2, w_1, w_2)$ represents the long-term difference in total expected costs accrued in the non-priority component when starting the system in state $(x_2 + 1, w_1, w_2)$ instead of (x_2, w_1, w_2). Since every customer generates costs at rate one per time unit, it is easily seen by a sample path comparison argument that this difference asymptotically (for $x_2 \to \infty$) amounts to the expected time it takes for the queue to empty when the system is started in the state $(x_2 + 1, w_1, w_2)$. For small values of x_2, this difference may depend slightly on w_1, since the event of M_1 being down at the start of the process may have a relative significant impact on the time to empty the queue, as the first repair on M_2 is likely to take longer than usual. However, as x_2 becomes larger, the time needed for the queue to empty becomes larger too, so that the process describing the conditions of the machines is more likely to have advanced towards an equilibrium in the meantime. As a result, the initial value of w_1 does not have a relatively significant impact on the difference (i.e., the time for the queue to empty) for larger x_2-values. In fact, the extra delay in the time to empty imposed by an initial failure of M_1 is expected to converge to a constant as x_2 increases. Based on these observations, we expect that asymptotically, the value w_1 will only appear in front of x_2-terms (and not higher powers) in $V^{nprc}(x_2, w_1, w_2)$. This asymptotic linear effect is studied in Conjecture 4.4. We also expect that V^{nprc} starts to exhibit this asymptotic behaviour very quickly as x_2 increases, since the process describing the conditions of the machines regenerates every time M_2 is repaired, and thus moves to an equilibrium rather quickly.

Now that we have identified the contribution of w_1, we study the behaviour of V^{nprc} in the direction of x_2 that is not explained by w_1. When ignoring the interaction with the priority queue (thus ignoring w_1), the queue of products in the non-priority component may be interpreted as an M/PH/1 queue, by incorporating the service interruptions (consisting of M_1 and M_2 repairs) into the service times of the products. Thus, queueing-theoretic intuition suggests that the relative value function for our problem may behave similarly to that of the M/PH/1 queue, particularly if the degree of interdependence between the queue lengths of Q_1 and Q_2 is not very high. It is known that the relative value function of such a queue is a quadratic polynomial (see e.g. [2]), so that asymptotically, V^{nprc} is likely to behave as a quadratic polynomial too. The second-order coefficient of the relative value function of the M/PH/1 queue satisfies the form $\frac{1}{2}(\mu_2^{\mathit{eff}} - \lambda_2)^{-1}$, where λ_2 is the arrival rate and μ_2^{eff} is the effective service rate; i.e., the long-term rate at which the products leave the system. As observed in Remark 4.1, the second-order coefficient α_1 of the static component in Lemma 4.1 is also of this form, independent of the value of w_2. Therefore, it is reasonable to assume that the second-order coefficient of V^{nprc} also satisfies this form, independent of the values for w_1, w_2. The involved effective service rate of M_2, μ_2^{eff}, is given in (12). By combining all arguments above, the conjecture follows. □

Note that the first-order coefficient of the polynomial, unlike the second-order coefficient, is expected to be dependent on w_1 as mentioned in the argument of Conjecture 4.3, but also on w_2, in line with the results on the components of the static policy. The first-order coefficient is studied in the next conjecture.

Conjecture 4.4. *Suppose that Conjecture 4.3 holds true, so that asymptotically*

$$\begin{aligned} &V^{nprc}(x_2,0,0) = \phi_1 x_2^2 + \phi_2 x_2 + \phi_3, && V^{nprc}(x_2,1,0) = \phi_1 x_2^2 + \psi_2 x_2 + \psi_3, \\ &V^{nprc}(x_2,0,1) = \phi_1 x_2^2 + \chi_2 x_2 + \chi_3, \text{ and } && V^{nprc}(x_2,1,1) = \phi_1 x_2^2 + \omega_2 x_2 + \omega_3 \end{aligned} \tag{17}$$

as $x_2 \to \infty$. Then, $\psi_2 = \phi_2 - \Delta_{1,0}$, $\chi_2 = \phi_2 - \Delta_{0,1}$, $\omega_2 = \phi_2 - \Delta_{1,1}$, where

$$\begin{aligned} \Delta_{0,1} &= \frac{\mu_2(\nu_1+\sigma_1)(\nu_1+\nu_2+\sigma_1+\sigma_2)}{\mu_2\nu_1\nu_2(\nu_1+\sigma_1+\sigma_2) - \lambda_2(\nu_1+\sigma_1)(\nu_1(\nu_2+\sigma_2)+\sigma_2(\nu_2+\sigma_1+\sigma_2))}, \\ \Delta_{1,0} &= \frac{\mu_2\nu_2(\nu_1+\sigma_1+\sigma_2)}{\mu_2\nu_1\nu_2(\nu_1+\sigma_1+\sigma_2) - \lambda_2(\nu_1+\sigma_1)(\nu_1(\nu_2+\sigma_2)+\sigma_2(\nu_2+\sigma_1+\sigma_2))} \quad \textit{and} \\ \Delta_{1,1} &= \frac{\mu_2(\nu_1+\nu_2+\sigma_1)(\nu_1+\sigma_1+\sigma_2)}{\mu_2\nu_1\nu_2(\nu_1+\sigma_1+\sigma_2) - \lambda_2(\nu_1+\sigma_1)(\nu_1(\nu_2+\sigma_2)+\sigma_2(\nu_2+\sigma_1+\sigma_2))}. \end{aligned}$$

Argument. The relative value function V^{nprc} is expected to satisfy the Poisson equations (16), also asymptotically for $x_2 \to \infty$. When substituting (17) into (16) for $x_2 > 0$, the constraints on ϕ_2, χ_2, ψ_2 and ω_2 mentioned above are necessary for the first-order terms in x_2 on both sides of the equations to be equal. □

Remark 4.3. As costs in the non-priority component are generated primarily by having customers in the queue, we expect the values of ϕ_3, χ_3, ψ_3 and ω_3 in (17) to be of very moderate significance compared to the second-order and first-order coefficients. As mentioned before, we also expect that V^{nprc} starts to exhibit its asymptotic behaviour very quickly as x_2 increases. Although we have not found an explicit solution for the first-order coefficients ϕ_2, χ_2, ψ_2 and ω_2, we can therefore still obtain accurate approximations for expressions such as $V^{prio}(x_1, x_2, 1, 0) - V^{prio}(x_1, x_2, 0, 0)$ and $V^{prio}(x_1, x_2, 0, 1) - V^{prio}(x_1, x_2, 0, 0)$ based on the information we have obtained. In particular, we have by combining the results in (14), (15), Conjecture 4.3 and Conjecture 4.4 that

$$\begin{aligned} &V^{prio}(x_1,x_2,1,0) - V^{prio}(x_1,x_2,0,0) \approx c_1((\upsilon_1-\upsilon_2)x_1 - \upsilon_3) - c_2\Delta_{1,0}x_2 \\ &V^{prio}(x_1,x_2,0,1) - V^{prio}(x_1,x_2,0,0) \approx -c_2\Delta_{0,1}x_2 \end{aligned} \tag{18}$$

with the parameters υ_1, υ_2, υ_3, $\Delta_{1,0}$ and $\Delta_{0,1}$ as defined in Section 4.2.1 and Conjecture 4.4 respectively. These two accurate approximations allow us to apply the one-step policy improvement algorithm based on the priority policy in Section 5.1.2.

In the two conjectures above, the long-term expected costs per time unit g^{nprc} was not regarded. However, in deciding which of the two possible priority policies (i.e., the policies corresponding to M_1 and M_2 being the high-priority machine) to use ultimately as an initial policy for the one-step policy improvement algorithm, an expression for this quantity is needed. Therefore, we end this section with an approximation for the expected costs per time unit g^{nprc}, again assuming that M_1 is the machine having priority.

Approximation 4.5. *An accurate approximation for the long-term expected costs per time unit g^{nprc} is given by*

$$g^{nprc} \approx \lambda_2 \mathbb{E}[\Gamma_{app}] + \frac{\lambda_2^2 \mathbb{E}[\Gamma_{app}^2]}{2(1-\lambda_2\mathbb{E}[\Gamma_{app}])} + \lambda_2 \frac{\sigma_2 \mathbb{E}[D^2]}{2(1+\sigma_2 ED)} \tag{19}$$

where

$$\mathbb{E}[\Gamma_{app}^i] = (-1)^i \frac{d^i}{ds^i}\Big(\frac{\mu_2}{\mu_2+s+\sigma_2(1-\widetilde{D}(s))}\Big)\Big|_{s=0}, \qquad \mathbb{E}[D^i] = (-1)^i \frac{d^i}{ds^i}\widetilde{D}(s)\Big|_{s=0}$$

and

$$\widetilde{D}(s) = \Big((1-z) + z\frac{\nu_1}{\nu_1+s}\Big)\frac{\nu_2}{\nu_2+s+\sigma_1(1-\frac{\nu_1}{\nu_1+s})},$$

with z as defined in (13).

Justification. This approximation is obtained by ignoring the interaction between the two components. Inspired by Remark 4.2, we approximate g^{nprc} by studying the queue length in an M/G/1 queue with server vacations, with completion times denoted by Γ instead of service times, which incorporate the time lost due to service interruptions as a result of a breakdown of M_2 during service. The server vacations, which start each time the queue becomes empty, include the down-periods of M_2 following a breakdown occuring when the queue is empty. Let $\widetilde{D}(s) = \mathbb{E}[e^{-sD}]$ be the Laplace-Stieltjes transform (LST) of the duration D of an M_2 down-period. This period D consists of an exponential (ν_2) repair time R_2 with LST $\widetilde{R}_2(s) = \frac{\nu_2}{\nu_2+s}$, and a Poisson ($\sigma_1 R_2$) number of interruptions N, each caused by a breakdown of M_1. Since M_1 has priority, these interruptions take an exponential (ν_1) repair time R_1 with LST $\widetilde{R}_1(s) = \frac{\nu_1}{\nu_1+s}$. Finally, when M_2 breaks down, it will have to wait for an M_1-repair to finish before repair on M_2 can start with probability z as defined in (13). Since the repair time of M_1 is memoryless, this waiting time also has the LST $\widetilde{R}_1(s)$. Thus, we have that

$$\begin{aligned}\widetilde{D}(s) =& \big((1-z)+z\widetilde{R}_1(s)\big)\int_{t=0}^{\infty} e^{-st}\Big(\sum_{n=0}^{\infty}\widetilde{R}_1^n(s)\mathbb{P}(N=n)\Big)\nu_2 e^{-\nu_2 t}dt\\ =& \big((1-z)+z\widetilde{R}_1(s)\big)\int_{t=0}^{\infty} e^{-st}\Big(\sum_{n=0}^{\infty}e^{-\sigma_1 t}\frac{(\sigma_1 t\widetilde{R}_1(s))^n}{n!}\Big)\nu_2 e^{-\nu_2 t}dt\\ =& \big((1-z)+z\widetilde{R}_1(s)\big)\widetilde{R}_2(s+\sigma_1(1-\widetilde{R}_1(s)))\\ =& \Big((1-z)+z\frac{\nu_1}{\nu_1+s}\Big)\frac{\nu_2}{\nu_2+s+\sigma_1(1-\frac{\nu_1}{\nu_1+s})}.\end{aligned}$$

The completion time Γ of a product, of which the distribution is represented by its LST $\widetilde{\Gamma}(s)$, consists of an exponentially (μ_2) distributed service time B_2 with LST $\widetilde{B}_2(s) = \frac{\mu_2}{\mu_2+s}$, and a Poisson ($\sigma_2 B_2$) number of interruptions, each caused by a breakdown of M_2. Due to the interaction between the components, we know that at the start of the first completion time after a vacation period, both machines are operational, biasing the duration of the first breakdown of M_2 after that time. When ignoring this effect of interaction, and assuming that each breakdown has a duration distributed according to D, we obtain similarly to the computations above that

$$\widetilde{\Gamma}(s) \approx \widetilde{B}_2(s+\sigma_2(1-\widetilde{D}(s))). \tag{20}$$

An application of the Fuhrmann-Cooper decomposition similar to Remark 4.2 suggests that

$$g^{nprc} \approx \mathbb{E}[L_{\text{M/G/1}}] + \mathbb{E}[L_{\text{vac}}],$$

where $\mathbb{E}[L_{\text{M/G/1}}] = \lambda\mathbb{E}[\Gamma] + \frac{\lambda_2\mathbb{E}[\Gamma^2]}{2(1-\lambda_2\mathbb{E}[\Gamma])}$ is the mean queue length of the number of products in an M/G/1 queue with Poisson (λ_2) arrivals and service times distributed according to the completion times Γ. Approximations for the moments of Γ follow by differentation of (20) with respect to s. The term $\mathbb{E}[L_{\text{vac}}]$ represents the expected queue length observed when the server is on a vacation, which is initiated any time the queue empties. This vacation lasts until the time of the next product arrival in case M_2 is operational at that point or otherwise, until the first time a repair of M_2 completes after that time. When conditioning on the former case, the expected queue length as observed by a vacation is obviously zero. When conditioning on the latter, this expectation resolves to the expected number of Poisson (λ_2) arrivals during the past part of a down-period D. The duration of this past part has expectation $\frac{\mathbb{E}[D^2]}{2\mathbb{E}[D]}$, of which the moments of D can be computed by differentation of $\widetilde{D}(s)$ with respect to s. The probability of the latter event occurring is hard to compute, as the durations of the vacation periods are weakly interdependent, due to the interaction between the components. By ignoring this dependence, we have by a renewal argument that this probability is well approximated by $\frac{\mathbb{E}[D]}{\frac{1}{\sigma_2}+\mathbb{E}[D]}$, where $\frac{1}{\sigma_2}$ is the expected duration of an up-period of M_2. As a result, $\mathbb{E}[L_{\text{vac}}] \approx \lambda_2\frac{\mathbb{E}[D^2]}{2\mathbb{E}[D]}\frac{\mathbb{E}[D]}{\frac{1}{\sigma_2}+\mathbb{E}[D]} = \lambda_2\frac{\sigma_2\mathbb{E}[D^2]}{2(1+\sigma_2\mathbb{E}[D])}$. By combining the results above, we obtain an approximation for g^{nprc} as summarised in (19). Note that the application of the Fuhrmann-Cooper decomposition requires that the completion times are mutually independent. However, in our case, this requirement is not met, again due to the interaction between the components. For example, a very long completion time may imply that the last actual service period of M_2 has been longer than usual. This in its turn implies that M_2 has been in operation for some time, so that if a M_2-breakdown occurs in the next completion time, it is likelier than usual that M_1 is also down at that point. This obviously has effects on the completion time. Due to this interdependence, the application of the Fuhrmann-Cooper decomposition also results in a computation error. However, all computation errors made share the same source, namely the interaction between the components, and in particular the role of M_1. As we already saw in Conjecture 4.3, the influence of M_1 on the relative value function is limited, especially for states with a large number of products in the queue. Therefore, we expect this approximation to be accurate, especially for the purpose of deciding which of the two priority policies available performs best. A numerical study, omitted for the sake of brevity, verifies the plausibility of this statement. $\square$

5 Derivation of a near-optimal policy

Based on the explicit expressions for the relative value functions of the static policy and the priority policy as obtained in the previous section, we derive a nearly optimal dynamic policy. We do so by applying the one-step policy improvement method on both the static policy and the priority policy in Section 5.1. The resulting improved policies π^{oss} and π^{osp} can then be used to construct a nearly optimal policy, as discussed in Section 5.2.

5.1 One-step policy improvement

One-step policy improvement is an approximation method that is distilled from the policy iteration algorithm in Markov decision theory. In policy iteration, one starts with an arbitrary policy π^{init} for which the relative value function V^{init} is known. Next, using these values, an improved policy π^{imp} can be obtained by performing a policy improvement step:

$$\pi^{imp}(\boldsymbol{s}) = \underset{\boldsymbol{a}\in\mathcal{A}_{\boldsymbol{s}}}{\arg\min}\{\sum_{\boldsymbol{s}'\in\mathcal{S}} P_{\boldsymbol{a}}(\boldsymbol{s},\boldsymbol{s}')V^{init}(\boldsymbol{s}')\}, \tag{21}$$

i.e., the minimising action of $K^{init}(\boldsymbol{s})$ as defined in (2). If $\pi^{imp} = \pi^{init}$, the optimal policy has been found. Otherwise, the procedure can be repeated with the improved policy by setting $\pi^{init} := \pi^{imp}$, generating a sequence converging to the optimal policy. However, as the relative value function of the improved policy may not be known explicitly, subsequent iterations may have to be executed numerically. To avoid this problem, the one-step policy improvement consists of executing the policy improvement step only once. In this case, the algorithm starts with a policy for which an expression for the relative value function is known. The resulting policy is then explicit and can act as a basis for approximation of the optimal policy. We now derive two one-step improved policies based on the results of the static policy and the priority policy as obtained in Section 4.

5.1.1 One-step improvement based on the static policy

In Section 4.1, we have found the relative value function V^{sta} for the class of static policies, in which each policy corresponds to a splitting parameter $p \in (0,1)$. As an initial policy for the one-step policy improvement, we take the policy which already performs best within this class with respect to the weighted number of products in the system. Thus, we take as an initial policy the static policy with splitting parameter

$$p^{oss} = \min_{p} \{ g_p^{sta} : p \in \mathcal{P} \}, \tag{22}$$

with g_p^{sta} as defined in Theorem 4.2 and where $\mathcal{P} \subset (0,1)$ is the set of splitting parameters which satisfy the stability conditions in (7). Then, by performing one step of policy improvement as given in (21), we obtain

$$\begin{aligned} \pi^{oss}(x_1,x_2,w_1,w_2) = \operatorname*{arg\,min}_{(q_1,q_2)\in\mathcal{A}_{(x_1,x_2,w_1,w_2)}} & \{ q_1\nu_1 (V_{p^{oss}}^{sta}(x_1,x_2,1,w_2) - V_{p^{oss}}^{sta}(x_1,x_2,0,w_2)) \\ & + q_2\nu_2 (V_{p^{oss}}^{sta}(x_1,x_2,w_1,1) - V_{p^{oss}}^{sta}(x_1,x_2,w_1,0)) \}. \end{aligned} \tag{23}$$

It is easily seen that $V_{p^{oss}}^{sta}(x_1,x_2,1,w_2) - V_{p^{oss}}^{sta}(x_1,x_2,0,w_2)$ and $V_{p^{oss}}^{sta}(x_1,x_2,w_1,1) - V_{p^{oss}}^{sta}(x_1,x_2,w_1,0)$ are non-positive for any $(x_1,x_2,w_1,w_2) \in \mathcal{S}$ by observing that $\alpha_{2,i} \geq \alpha_{1,i}$ and $\alpha_{3,i} \geq 0$, $i = 1,2$. This means that π^{oss} satisfies the properties mentioned in Section 3.1. Therefore, we can simplify (23) to

$$\begin{aligned} \pi^{oss}(x_1,x_2,w_1,w_2) = \operatorname*{arg\,min}_{(q_1,q_2)\in\{(1-w_1,0),(0,1-w_2)\}} & \{ q_1\nu_1 (V_{p^{oss}}^{sta}(x_1,x_2,1,w_2) - V_{p^{oss}}^{sta}(x_1,x_2,0,w_2)) \\ & + q_2\nu_2 (V_{p^{oss}}^{sta}(x_1,x_2,w_1,1) - V_{p^{oss}}^{sta}(x_1,x_2,w_1,0)) \}. \end{aligned}$$

Substituting $V_{p^{oss}}^{sta}$ as obtained in Theorem 4.2 in this expression yields the following one-step improved policy:

$$\pi^{oss}(x_1,x_2,w_1,w_2) = \begin{cases} (0,0) & \text{if } w_1 = w_2 = 1, \\ (1,0) & \text{if } w_1 = 1 - w_2 = 0, \text{ or if } w_1 w_2 = 0 \text{ and} \\ & c_1\nu_1((\alpha_{1,1} - \alpha_{2,1})x_1 - \alpha_{3,1}) \leq c_2\nu_2((\alpha_{1,2} - \alpha_{2,2})x_2 - \alpha_{3,2}), \\ (0,1) & \text{otherwise} \end{cases} \tag{24}$$

for $(x_1,x_2,w_1,w_2) \in \mathcal{S}$, where expressions for the α-coefficients are obtained by substituting the value for p in the expressions given in Theorem 4.2 by its optimised counterpart p^{oss}. Thus, whenever both machines are not operational, the repairman repairs the machine M_i for which $c_i\nu_i((\alpha_{1,i} - \alpha_{2,i})x_i - \alpha_{3,i})$ is smallest, when adhering to π^{oss}.

Remark 5.1. If $\mathcal{P}$ is empty, there is no static policy available which results into a system with stable queues. In such circumstances, the static policy cannot be used as an initial policy for the one-step improvement approach. The priority policy as studied in Section 4.2, may however still result in a stable system. If this is the case, the priority policy may act as an initial policy for the one-step policy improvement method, as explained in the next section.

Remark 5.2. Whenever $\mathcal{P}$ is not empty, the optimal splitting parameter p^{oss} is guaranteed to exist. As g_p^{sta} is a continuous function in p for $p \in \mathcal{P}$, the optimal splitting parameter p^{oss} is then a root of $\frac{d}{dp} g_p^{sta}$ in the domain $\mathcal{P}$. This derivative, which forms a sixth-order polynomial in p, defies the possibility of deriving an explicit expression for p^{oss}. For implementation purposes, however, this poses no significant problems, as such roots can be found numerically up to arbitrary precision with virtually no computation time needed.

5.1.2 One-step improvement based on the priority policy

Although an explicit expression for the relative value function V^{prio} is not tractable, we have identified enough of its characteristics in Section 4.2 to allow the use of a priority policy as an initial policy. In particular, we will use the approximations obtained in Remark 4.3 in this section to apply the one-step policy improvement algorithm on a priority policy. As was the case in the previous section, we first decide which policy within the class of priority policies to take as an initial policy, by checking which policy performs best with respect to g^{prio}, the weighted number of products in the system. In this case, however, there are only two priority policies available, each prioritizing their own machine. By combining (14), (15) and (19), we already have an approximation of g^{prio} for the priority policy with M_1 as the high-priority machine. A similar approximation for the priority policy where M_2 is the high-priority machine, is easily obtained by interchanging indices. Consequently, we select the priority policy with the smallest approximated value for g^{prio} as an initial policy for the one-step policy improvement. The fact we base this decision on approximations rather than exact values poses no problems, since the approximations are accurate and share the same source of approximation error. Moreover, we only check which of the approximated values is smallest, so that any reasonable approximation error is not likely to alter the outcome.

Now that the initial policy is selected, we perform the improvement step. In the sequel, we will show how to compute the one-step improved policy based on the priority policy where M_1 is the high-priority machine. For the one-step improved policy based on the other priority policy, one again simply interchanges the indices of the model parameters of the following computation. The improvement step as given in (21) implies, after performing the same simplification as in the case of the static policy, that

$$\begin{aligned}\pi^{osp}(x_1, x_2, w_1, w_2) = \underset{(q_1,q_2)\in\{(1-w_1,0),(0,1-w_2)\}}{\arg\min} &\{q_1\nu_1(V^{prio}(x_1, x_2, 1, w_2) - V^{prio}(x_1, x_2, 0, w_2)) \\ &+ q_2\nu_2(V^{prio}(x_1, x_2, w_1, 1) - V^{prio}(x_1, x_2, w_1, 0))\}. \quad (25)\end{aligned}$$

The simplification is justified by the fact that $V^{prio}(x_1, x_2, 1, w_2) - V^{prio}(x_1, x_2, 0, w_2)$ and $V^{prio}(x_1, x_2, w_1, 1) - V^{prio}(x_1, x_2, w_1, 0)$ are obviously non-positive, since also under the priority policy it is always beneficial for the system to have a machine operational. Due to this, it is clear that $\pi^{osp}(x_1, x_2, w_1, w_2)$ in (25) resolves to $((1 - w_1), (1 - w_2))$ in case $w_1 = w_2 = 1$, $w_1 = 1 - w_2 = 1$ or $1 - w_1 = w_2 = 1$. For the case $w_1 = w_2 = 0$, expressions for $V^{prio}(x_1, x_2, 1, 0) - V^{prio}(x_1, x_2, 0, 0)$ and $V^{prio}(x_1, x_2, 0, 1) - V^{prio}(x_1, x_2, 0, 0)$, are however not available. Due to their general intractability, we use the approximations for these differences as derived in (18) instead. By plugging these approximations into (25) in case $w_1 = w_2 = 0$, we obtain, with a slight abuse of notation, that

$$\pi^{osp}(x_1, x_2, w_1, w_2) = \begin{cases} (1, 0) & \text{if } w_1 = 1 - w_2 = 0, \text{ or if } w_1 w_2 = 0 \text{ and} \\ & \nu_1(c_1((\upsilon_1 - \upsilon_2)x_1 - \upsilon_3) - c_2\Delta_{1,0}x_2) \le -c_2\Delta_{0,1}\nu_2 x_2, \\ (0, 1) & \text{otherwise,} \end{cases} \quad (26)$$

where the parameters υ_1, υ_2, υ_3, $\Delta_{1,0}$ and $\Delta_{0,1}$ are as defined in Section 4.2.1 and Conjecture 4.4 respectively.

Remark 5.3. We have based π^{osp} on approximations for the relative value function V^{prio}, rather than exact expressions. Nevertheless, we have already argued in Section 4.2.2 that these approximations are accurate. Moreover, the argmin-operator in (25) only checks which of the two arguments is smallest. Therefore, the improvement step is very robust against approximation errors, especially since both arguments share the same source of approximation error. Numerical results confirm that the policy π^{osp} based on approximations thus differs little to not at all from the policy which we would have obtained if explicit expressions for V^{prio} had been available.

5.2 Resulting near-optimal policy

In the previous section, we have constructed improved policies based on the static policy and the priority policy respectively. However, the question remains which of these policies should be followed by the repairman, given a particular case of the model. In this section, we suggest a near-optimal policy, which chooses either of the two policies based on the model parameters. We do so by comparing the two improved policies as given in (24) and (26) as well as their initial policies.

First, we observe that the two improved policies satisfy the structural properties of the optimal policy. Both π^{oss} and π^{osp} always instruct the repairman to work at full capacity whenever at least one of the machines is down, and therefore satisfy the work-conserving property as derived in Section 3.1. Furthermore, when both of the machines are down, the two improved policies base the action on threshold curves (or, in this case, threshold lines), so that they both also satisfy the properties discussed in Section 3.2. As both of the improved policies satisfy the required properties, we base the decision on which of the improved policies to follow on their respective initial policies.

In terms of feasibility, the static policy and the priority policy complement each other. For any model instance, one can construct an improved static policy, if there exists a static policy that results into stable queues, i.e., there exists a value $p \in (0, 1)$ such that (7) holds. Similarly, an improved priority policy can be constructed if either (11) holds, or its counterpart corresponding to taking M_2 as the high-priority machine. There are cases of the model for which the construction of an improved static policy is possible, whereas the construction of an improved priority policy is not possible. There are also cases for which the reverse holds true. In these cases, it is clear which of the two improved policies to take as a near-optimal policy. However, when both of the approaches are adequate, we base our decision on other characteristics of the initial policies.

A static policy would be optimal in a similar model, where the repairman has no information to base his decision on (i.e., it has no knowledge about the queue lengths and the state of the machines). It is not optimal in the current model, as it is not work-conserving; i.e., the server works only at partial capacity when exactly one of the machines is down. However, this problem does not arise in the improved version of the static policy. A priority policy is intuitively easily seen to be optimal in another similar model, where the arrival streams of products are such that the queues of the products are never exhausted. The possibility of having a machine in an operational but idle state then disappears, so that the optimal policy always gives priority to one machine over the other, due to faster service of products, a slower breakdown, faster repair times, or a higher cost rate. We therefore expect the priority policy (and thus also its improved version) to work particularly well in our model when the model parameters are skewed in the favour of repair of a certain machine and where the queues of products are particularly heavily loaded. The performance of the improved static policy, however, is not expected to be as sensitive to the amount of workload, since the static policy balances the repair fractions based on, among other things, the load offered to each of the queues.

Based on the observations above, we suggest a near-optimal policy that prescribes to follow the improved static policy as derived in Section 5.1.1, if there is a static policy available that results into a stable system. If this is not the case, the queues of products are relatively heavily loaded, so that an improved priority policy with the appropriate machine selected as high-priority machine should be followed. More concretely, the near-optimal policy is given by the following scheme:

1. As defined earlier, let $\mathcal{P} \subset (0, 1)$ be the set of values for the splitting parameter p for which the static policy results into a stable system, i.e., the values for p which satisfy the stability conditions in (7). If this set is non-empty, the near-optimal policy is given by π^{oss} as expressed in (24), i.e., the one-step improved version of the static policy with splitting parameter p^{oss} as defined in (22). If $\mathcal{P}$ is empty, go to step 2.
2. Check whether the priority policy with M_1 being the high-priority machine is stable by seeing whether its stability conditions in (11) hold. Equivalently, check whether the priority policy with M_2 as the high-priority machine is stable, by checking whether its stability conditions, the equivalent of (11) obtained by interchanging indices, holds. If both policies are stable, go to step 3. If exactly one out of these policies is stable, then mark this one as the selected initial policy, and go to step 4. If neither are stable, there is no suggested policy available, as all possible initial policies result into unstable queues.
3. Approximate the average costs per time unit of the priority policy with M_1 as the high-priority machine by combining (14), (15) and Approximation 4.5. Do the same for the priority policy with M_2 as the high-priority machine, by evaluating the expressions with the parameter indices interchanged. Mark the policy with the lowest approximated average costs as the selected initial policy, and go to step 4.
4. In case the policy with M_1 as a high-priority machine is marked as the selected initial policy, then the near-optimal policy is given by (26). In the opposite case, the near-optimal policy is also given by (26), but again with the indices of the model parameters interchanged.

We end this section with several remarks concerning the obtained nearly optimal policy.

Remark 5.4. In theory, it is possible that there is no static policy or priority policy available which results into stable queues, whereas 'stable policies' exist in general. Then, the approach given in this section is not adequate to solve the assignment problem considered. In such a case, the set of stable policies is usually very small, so that finding a stable policy is already hard, considerably complicating the search for a near-optimal one. However, in general, this is not likely to happen, as the stability conditions (7) and (11) combined for all possible static and priority policies cover a wide range of parameter settings. As we will see in Section 6, the derived policy may even be applied in model instances where the policy of repairing the machines in a first-come-first-served order does not keep both queues stable.

Remark 5.5. Many optimisation approaches in Markov decision theory suffer from the curse of dimensionality. When dimensions are added to the state space, e.g. by adding more machines to the problem, the size of the state space increases considerably, so that numerical computation techniques break down due to time and resource constraints. Note, however, that the approach presented in this paper generally scales well in the number of machines and the corresponding queues of products. The one-step improved policy based on the static policy is easily modified to allow for models with $N > 2$ machines, since a decomposition of the system in the fashion of (8) can then be done into N components. After finding a vector of splitting parameters $(p_1^{oss}, p_2^{oss}, \ldots, p_N^{oss})$, the execution of the one-step improvement algorithm will then still result into a simple decision rule similar to (24). Likewise, the priority policy may be used to derive near-optimal policies in a model with larger dimensions. The current approximation for the relative value function V^{prio} in the case of $N = 2$ already accounts for the components containing the two most prioritised machines in a model with $N > 2$ machines, as the repair capacity assigned to a machine is not affected by the breakdown of a machine with lower priority. When approximations for the relative value function pertaining to lower prioritised components can be found, a nearly optimal policy follows similarly to the case of $N = 2$.

6 Numerical study

In this section, we numerically assess the performance of the near-optimal policy obtained in Section 5 with respect to the optimal policy. We do this by comparing the average costs per time unit of both policies applied to a large number of model instances. We will see that the near-optimal policy performs very well over a wide range of parameter settings. Moreover, we observe several parameter effects.

The complete test bed of instances that are analysed contains all 1944 possible combinations of the parameter values listed in Table 1. This table lists multiple values for the cost weights of having products in Q_1 and Q_2 (c_1 and c_2), the service rates at which M_1 and M_2 serve products when operational (μ_1 and μ_2, their breakdown rates (σ_1 and σ_2) as well as their repair rates (ν_1 and ν_2). Finally, the product arrival rates λ_1 and λ_2 are specified by the values of the parameters ρ_1^{FCFS} and ρ_2^{FCFS} given in the table, where ρ_i^{FCFS} represents the workload offered to M_i, if the repairman would repair the machines in a first-come-first-serve (FCFS) manner. More specifically, the arrival rates are taken such that the values of the workload

$$\rho_i{}^{FCFS} = \frac{\lambda_i}{\mu_i} \frac{1}{\theta_i^{FCFS}}$$

would coincide with those given in Table 1, if the repairman were to follow a FCFS policy. In this expression, θ_i^{FCFS} represents the fraction of time that M_i is operational under a FCFS policy, and can be obtained by modelling the second layer of the model (cf. Figure 1) as a continuous-time Markov chain with a finite state space. The values for ρ_i^{FCFS}, σ_i and ν_i are varied in the order of magnitude through the values a_i^{ρ}, a_i^{σ} and a_i^{ν} as specified in the table and in the imbalance through the values b_j^{ρ}, b_j^{σ} and b_j^{ν}. For example, the workload values $(\rho_1^{FCFS}, \rho_2^{FCFS})$ run from $(0.25 \cdot \frac{2}{3}, 0.25 \cdot \frac{4}{3}) = (\frac{1}{6}, \frac{1}{3})$, being small and putting the majority of the load on the second queue, to $(0.75 \cdot \frac{4}{3}, 0.75 \cdot \frac{2}{3}) = (1, 0.5)$, being large and putting the majority on the first queue. Observe that in the latter case, ρ_1^{FCFS} takes the value of one. Thus, we also consider cases where not all of the queues would be stable, if the repairman would repair the machines in a FCFS fashion.

For the systems corresponding to each of the parameter combinations in Table 1, it turns out that there is always at least one static policy or priority policy available as an initial policy, so that the near-optimal policy is feasible. We numerically compute the average costs $g^{n\text{-}opt}$ incurred per time unit by the system if the repairman would follow the near-optimal policy as suggested in Section 5.2, as well as the average costs g^{opt} incurred per time unit if the repairman would follow the optimal policy. We do this by using the value iteration algorithm (see e.g. [17]).

Parameter	Considered parameter values
c_1	$\{0.25, 0.75\}$
c_2	$\{1\}$
$(\rho_1^{FCFS}, \rho_2^{FCFS})$	$a_i^\rho \cdot b_j^\rho \qquad \forall i, j$ where $\boldsymbol{a}^\rho = \{0.25, 0.5, 0.75\}$ and $\boldsymbol{b}^\rho = \{(\frac{2}{3}, \frac{4}{3}), (1, 1), (\frac{4}{3}, \frac{2}{3})\}$
(μ_1, μ_2)	$\{(0.75, 1.25), (1.25, 0.75), (1., 1.)\}$
(σ_1, σ_2)	$a_i^\sigma \cdot b_j^\sigma \qquad \forall i, j$ where $\boldsymbol{a}^\sigma = \{0.1, 1\}$ and $\boldsymbol{b}^\sigma = \{(\frac{1}{2}, \frac{3}{2}), (1, 1), (\frac{3}{2}, \frac{1}{2})\}$
(ν_1, ν_2)	$a_i^\nu \cdot b_j^\nu \qquad \forall i, j$ where $\boldsymbol{a}^\nu = \{0.1, 1\}$ and $\boldsymbol{b}^\nu = \{(\frac{1}{2}, \frac{3}{2}), (1, 1), (\frac{3}{2}, \frac{1}{2})\}$

Table 1: Parameter values of the test bed.

	0-0.1%	0.1-1%	1-10%	10-25%
% of rel. differences	38.37%	28.50%	32.66%	0.46%

Table 2: Relative differences between $g^{n\text{-}opt}$ and g^{opt} categorised in bins.

Subsequently, we compute the relative difference between these approximations, i.e.

$$\Delta := 100\% \times \frac{g^{n\text{-}opt} - g^{opt}}{g^{opt}}.$$

Obviously, Δ cannot take negative values, and the closer the value of Δ is to zero, the better the near-optimal policy performs. In Table 2 the resulting 1944 relative differences are summarised. We note that the vast majority of relative differences do not exceed 10%, and more than half of the cases constitute a difference lower than 1%. These results show that the near-optimal policy works very well for typical systems. The worst performance of the near-optimal policy encountered in the test bed is the exceptional case with parameters $(c_1, c_2) =$ (0.25, 1), $(\rho_1^{FCFS}, \rho_2^{FCFS}) =$ (1, 0.5), $(\mu_1, \mu_2) =$ (0.75, 1.25), $(\sigma_1, \sigma_2) =$ (0.15, 0.05) and $(\nu_1, \nu_2) =$ (0.05, 0.15), as well as the results $g^{opt} = 26.37$, $g^{n\text{-}opt} = 32.70$ and consequently $\Delta = 24.05\%$. For this instance, any static policy, as well as the FCFS policy, results into unstable queues, due to the high workload. Moreover, this instance is characterised by highly asymmetric model parameters, but in such a way that neither of the machines would be a clear candidate for the role of the high-priority machine in the priority policy.

To observe any further parameter effects, we also give the mean relative difference categorised in some of the variables in Table 3. Based on these results, four factors determining the quality of the near-optimal policy can be identified:

- Table 3(a) suggests that the closer the value of c_1 is to the value of c_2, the better the quality of the near-optimal policy becomes. A similar effect can be observed in Table 3(c) with the values ρ_1^{FCFS} and ρ_2^{FCFS}. These effects suggest that the level of asymmetry in the parameters plays a role in the effectiveness of the near-optimal policy. Intuitively, this makes sense, as the optimal policy gets easier to predict when the system becomes more symmetric. For example, in the case of a completely symmetric model (i.e., $\lambda_1 = \lambda_2$, $\mu_1 = \mu_2$, etc.), the threshold curve of the optimal policy is easily seen to be the line $x_1 = x_2$ by a switching argument. In that case, the improved static policy also attains this curve, which suggests that the near-optimal policy is optimal in symmetric systems if the workloads are not too high.
- Judging by Table 3(b), the performance of the near-optimal policy with respect to the optimal policy becomes worse when the workload of products offered to the queues increases. This can be explained by the fact that in

(a)

c_1	0.25	0.75
Mean rel. diff.	1.36%	1.16%

(b)

a_i^ρ	0.25	0.5	0.75
Mean rel. diff.	0.60%	1.01%	2.16%

(c)

b_j^ρ	$(\frac{2}{3}, \frac{4}{3})$	(1, 1)	$(\frac{4}{3}, \frac{2}{3})$
Mean rel. diff.	1.35%	1.17%	1.26%

(d)

(μ_1, μ_2)	(0.75, 1.25)	(1,1)	(1.25, 0.75)
Mean rel. diff.	1.36%	1.24%	1.17%

(e)

a_i^σ	0.1	1
Mean rel. diff.	0.48%	2.04%

(f)

a_i^ν	0.1	1
Mean rel. diff.	2.10%	0.41%

(g)

b_j^σ	$(\frac{1}{2}, \frac{3}{2})$	(1,1)	$(\frac{3}{2}, \frac{1}{2})$
Mean rel. diff.	1.02%	1.24%	1.51%

(h)

b_j^ν	$(\frac{1}{2}, \frac{3}{2})$	(1,1)	$(\frac{3}{2}, \frac{1}{2})$
Mean rel. diff.	1.60%	1.17%	1.00%

Table 3: Mean relative difference categorised in each of the parameters as specified in Table 1.

case of a smaller workload, products on average encounter less waiting products in their respective queue, and therefore are less influenced by the downtimes of their machines, which occured before their arrival. In its turn, this means that the sojourn time of products in the system is less sensitive to any sub-optimal decisions taken in the past, improving the accuracy of the near-optimal policy. In the extreme case where the workload offered to each queue equals zero (i.e., there are no products arriving), any policy is optimal, as the system does not incur any costs in that case.

- From Tables 3(e) and 3(f) it is apparent that the quality of the near-optimal policy is influenced by the values of a_i^σ and a_i^ν. This can be mainly explained by the fact that these values determine the level of competition between the machines for access to the repairman. When breakdowns do not occur often and repairs are done quickly, the event of having both machines down is exceptional, so that any sub-optimality of the policy used is expected to have a relatively little impact on the average costs.
- Tables 3(d), 3(g) and 3(h) seem to contradict with the first factor mentioned, as the near-optimal policy seems to perform better when M_2 has more 'favourable' characteristics with respect to M_1; i.e., the fast product services, slow breakdowns and fast repairs make it lucrative to repair M_2 at the expense of additional downtime for M_1. However, note that this effect occurs because the cost weights are already taken in favour of the repair of M_2 in every instance of the test bed. When the workloads are such that the static policy becomes infeasible, a priority policy with M_2 as the high-priority machine will then already be close to optimal. Therefore, its improved version also works particularly well. However, if, as opposed to the cost weights, the rates of product services, breakdowns and repairs are in favour of M_1, a priority policy works less well, since there is no clear candidate for the high-priority machine anymore. This leaves room for sub-optimality of the improved priority policy.

Acknowledgements

The authors wish to thank Onno Boxma for valuable comments on earlier drafts of the present paper.

References

[1] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[2] S. Bhulai. On the value function of the M/Cox(r)/1 queue. *Journal of Applied Probability*, 43:363–376, 2006.

[3] S. Bhulai. Dynamic routing policies for multiskill call centers. *Probability in the Engineering and Informational Sciences*, 23:101–119, 2009.

[4] S. Bhulai and F. M. Spieksma. On the uniqueness of solutions to the Poisson equations for average cost Markov chains with unbounded cost functions. *Mathematical Methods of Operations Research*, 58:221–236, 2003.

[5] J.L. Dorsman, R.D. Van der Mei, and M. Vlasiou. Analysis of a two-layered network with correlated queues by means of the power-series algorithm. Technical Report 2012-05, Eurandom Preprint Series, 2012.

[6] J.L. Dorsman, M. Vlasiou, and O.J. Boxma. Marginal queue length approximations for a two-layered network with correlated queues. Technical Report 2011-43, Eurandom Preprint Series, 2011.

[7] G. Franks, T. Al-Omari, M. Woodside, O. Das, and S. Derisavi. Enhanced modeling and solution of layered queuing networks. *IEEE Transactions on Software Engineering*, 35:148–161, 2009.

[8] S.W. Fuhrmann and R.B. Cooper. Stochastic decompositions in the M/G/1 queue with generalized vacations. *Operations Research*, 33:1117–1129, 1985.

[9] D. Gross and J.F. Ince. The machine repair problem with heterogeneous populations. *Operations Research*, 29:532–549, 1981.

[10] R. Haijema and J. Van der Wal. An MDP decomposition approach for traffic control at isolated signalized intersections. *Probability in the Engineering and Informational Sciences*, 22:587–602, 2008.

[11] L. Haque and M.J. Armstrong. A survey of the machine interference problem. *European Journal of Operational Research*, 179:469–482, 2007.

[12] M. Harkema, B.M.M. Gijsen, R.D. Van der Mei, and Y. Hoekstra. Middleware performance: A quantitative modeling approach. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Communication Systems (SPECTS)*, pages 733–742, 2004.

[13] L. Kleinrock. *Queueing Systems, Volume II: Computer Applications*. Wiley, New York, 1976.

[14] G.M. Koole. Monotonicity in Markov reward and decision chains: Theory and applications. *Foundations and Trends in Stochastic Systems*, 1:1–76, 2006.

[15] J.M. Norman. *Heuristic Procedures in Dynamic Programming*. Manchester University Press, 1972.

[16] T.J. Ott and K.R. Krishnan. Separable routing: A scheme for state-dependent routing of circuit switched telephone traffic. *Annals of Operations Research*, 35:43–68, 1992.

[17] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons Inc., 1994.

[18] S. A. E. Sassen, H.C. Tijms, and R.D. Nobel. A heuristic rule for routing customers to parallel servers. *Statistica Neerlandica*, 51:107–121, 1997.

[19] J. Wijngaard. Decomposition for dynamic programming in production and inventory control. *Engineering and Process Economics*, 4:385–388, 1979.