A Product-form Network for Systems with Job Stealing Policies

DILETTA OLLIARO, DAIS, Università Ca' Foscari di Venezia, Mestre Venezia, Italy
GIULIANO CASALE, Department of Computing, Imperial College London, London, United Kingdom
ANDREA MARIN, DAIS, Università Ca' Foscari di Venezia, Mestre Venezia, Italy
SABINA ROSSI, DAIS, Università Ca' Foscari di Venezia, Mestre Venezia, Italy

In queueing networks, product-form solutions are of fundamental importance to efficiently compute performance metrics in complex models of computer systems. The product-form property entails that the steady-state probabilities of the joint stochastic process underlying the network can be expressed as the normalized product of functions that only depend on the local state of the components. In many relevant cases, product-forms are the only way to perform exact quantitative analyses of large systems.

In this work, we introduce a novel class of product-form queueing networks where servers are always busy. Applications include model of systems where successive refinements on jobs improve the processes quality but are not strictly required to obtain a result. To this aim, we define a job movement policy that admits instantaneous migrations of jobs from non-empty waiting buffers to empty ones. Thus, the resulting routing scheme is state-dependent. This class of networks maximizes the system throughput.

This model can be implemented with arbitrary topology, including feedback, and both in an open and closed setting. As far as closed systems are concerned, we give a convolution algorithm and the corresponding mean value analysis to compute expected performance indices for closed models.

CCS Concepts: • **Mathematics of computing** → **Markov processes**; **Queueing theory**; • **Networks** → *Network performance modeling*; • **General and reference** → *Performance*; • **Computing methodologies** → **Model development and analysis;**

Additional Key Words and Phrases: Queueing theory, product-form solutions, reversed compound agent theorem, fetching policy

ACM Reference Format:
Diletta Olliaro, Giuliano Casale, Andrea Marin, and Sabina Rossi. 2024. A Product-form Network for Systems with Job Stealing Policies. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 9, 2, Article 6 (March 2024), 26 pages.
https://doi.org/10.1145/3643845

1 INTRODUCTION

Product-form models and, specifically, product-form queueing networks, have played a central role for the community of researchers and practitioners involved in performance evaluation of

This work is partially supported by the Project PRIN 2020 "Nirvana - Noninterference and Reversibility Analysis in Private Blockchains" - N. 20202FCJMH.
Authors' addresses: D. Olliaro, A. Marin, and S. Rossi, DAIS, Università Ca' Foscari di Venezia, via Torino, 155, Mestre Venezia 30172, Italy; e-mails: diletta.olliaro@unive.it, marin@unive.i, sabina.rossi@unive.it; G. Casale, Department of Computing, Imperial College London, Exhibition Rd, South Kensington, London SW7 2BX, United Kingdom; e-mail: g.casale@imperial.ac.uk.

ACM 2376-3639/2024/03-ART6
https://doi.org/10.1145/3643845

computer and telecommunication systems. Product-forms allow decomposing a model into simple components each of which is parameterised in such a way that its stationary solution in isolation can be efficiently obtained. The steady-steady state probabilities of the joint model are then obtained as the normalized product of these isolated solutions on all the ergodic joint states. In this way, the state space explosion problem can be tackled efficiently.

In this work, we present a novel product-form stochastic network model in which servers are never idle as they *steal* jobs from other stations when the queues of the latter are empty. To illustrate an application example, consider a distributed system for image processing that applies several filters in cascade to the input. Let the system be composed by $K > 1$ application servers; each filter is applied to the image processing job by a different server, $S_1, \ldots, S_K$. If each job is processed sequentially, we can model the system as a tandem of queues where arrivals occur at S_1, with rate λ, and completions at S_K. Let the expected service time at server S_i be μ_i^{-1}. The underlying queueing model of such a system is represented in Figure 1 and corresponds to a traditional Jackson network. Accordingly, the maximum system throughput is bounded by $\mu_{\min} \triangleq \min_i\{\mu_i\}$.

For a system of this kind we can maximize throughput to $\mu_{\max} \triangleq \max_i\{\mu_i\}$ as follows: suppose we order the servers by increasing speed, that is, $\mu_i \leq \mu_{i+1}$ for all $i = 1, \ldots, K-1$. Upon finishing its work on a job, server S_i sends the job to server S_{i+1} as usual. However, if S_i does not have any job in its queue, it steals a job from server S_{i-1} and, if this is also empty, from S_{i-2} and so forth. If also server S_1 is empty, we take a job from the outside. Under this scheme, some of the jobs have not passed through all the filtering stages and hence the image quality may be degraded in return for better performance since the throughput of S_K is maximized to μ_K. In order to further underline the difference with a Jackson network we will call networks behaving according to the proposed job stealing policy *fetching networks* and the underlying queueing model for a tandem topology is represented in Figure 2.

To give the reader an idea, consider a traditional Jackson queueing network as the one in Figure 1, but comprising four tandem stations each characterized by specific service rates: $\mu_1 = 3\ j/s, \mu_2 = 4\ j/s, \mu_3 = 5\ j/s,$ and $\mu_4 = 6\ j/s$. In such a configuration, the system's maximum throughput is determined by the *slowest ship of the convoy*, meaning the station with the slowest service rate. In this particular network, Station 1 can serve a maximum of 3 jobs per second, establishing it as the bottleneck for the entire system. Consequently, the overall network throughput cannot exceed this rate, despite the subsequent stations having higher individual capacities. Now, let us consider the same tandem stations organized within a *fetching network* as the one of Figure 2. In this context, an interesting dynamic emerges where, even though Station 1 is constrained to a maximum service rate of $3\ j/s$, faster stations can steal jobs from the other (slower) stations. This implies that the maximum achievable throughput for the system is no longer dictated by the slowest station; but rather, by the fastest one. Accordingly, in this scenario, the maximum achievable throughput for the system is governed by Station 4 which can serve 6 job per second. Moreover, in the standard tandem system, as the throughput approaches $\mu_{\min}$, the queue length at the bottleneck tends to infinity, while in our system, we will show that we can reach $\mu_{\max}$ and maintain finite queue lengths at all queues. Although alternative strategies can be employed to efficiently reuse resources and minimize response times, it is crucial to highlight that the mechanism of *taking a job from the external world* plays a pivotal role in increasing the expected number of job completions per unit of time. It is also worth noting that, later in this work, in order to formalize these dynamics and prove the product-form result that underlies this model, we consider the case of one station that is never empty (i.e., a station from which it is always possible to steal a job), rather than characterizing an exogenous arrival process.

The idea of letting some jobs skip some phases of service is not new and has been first introduced by Pittel in [22] for finite capacity queues. This has been widely used both for studying computer and telecommunication systems and in operations research (see, e.g., [21, 26, 27]).

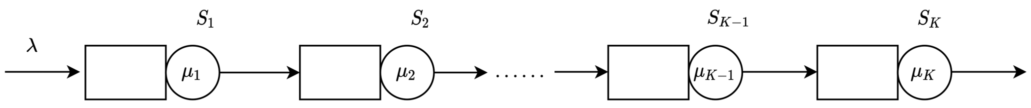

Fig. 1. A tandem Jackson network.

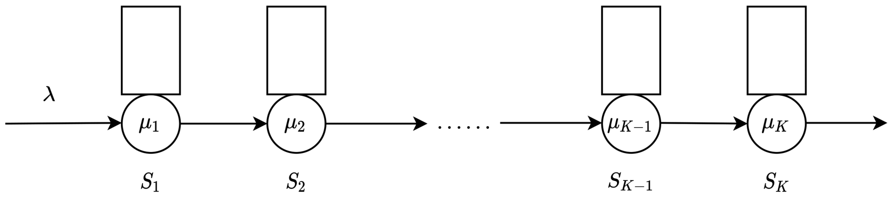

Fig. 2. A tandem fetching network.

However, these models express policies for job movements that describe where that job has to be placed. Conversely, to the best of our knowledge, this is the first policy that defines where the job has to be fetched from.

The stochastic networks that we analyze admit arbitrary routing that may involve cycles and they can be either open or closed. Notably, the movement of jobs is governed by probabilistic policies that are state dependent. In fact, whenever a station needs to steal a job from another station, the latter is chosen probabilistically, but conditional on having at least one job in the buffer and no non-empty queues before it.

Another important observation is that the stochastic network is neither reversible nor quasi-reversible according to Kelly's notion of quasi reversibility [14] (see Sections 2 and 3.5). Moreover, the standard argument for proving product-form based on the substitution of the expression in the **global balance equations** (**GBE**) is here made complicated by the presence of absorption probabilities in the definition of the underlying CTMC.

Our proof overcomes this issue using the multi-way extension of the **Reversed Compound Agent Theorem** (**RCAT**) presented in [12]. The intrinsic compositionality of RCAT allows us to present the result and its proof in an elegant way and without resorting to heavy algebraic derivations. Nevertheless, for some specific examples, we are going to show how the GBE are satisfied by the proposed solution to emphasise the difficulties of the traditional approach.

Summing up, the contributions of this article are the following:

— We introduce a novel class of models and their product-form solution for the performance evaluation of systems with probabilistic routing and job stealing. The proposed model is defined both for open and closed systems.
— We propose a convolution algorithm and the corresponding **mean value analysis** (**MVA**)[23] to study closed systems and derive average performance indices. In general, throughput is shown to be different from the one obtained by Buzen's convolution [2] in closed Gordon-Newell networks.

The article is structured as follows: Section 2 gives an overview of the related work. In Section 3, we present in detail the open model and its underlying CTMC and prove the main product-form result. Section 4 generalizes the results to closed models. In Section 5, the relevant average stationary performance indices are defined and we show how to derive them for open and closed systems in equilibrium. In Section 6, we further describe how to use MVA to efficiently retrieve performance indices in closed networks. Finally, Section 7 gives some concluding remarks and suggestions for future work.

2 RELATED WORK

The literature on product-form models is very rich. In this section, we analyse the contributions that, to the best of our knowledge, are the most closely related to ours.

Since Kelly's work [14], product-form models have been studied in a modular way thanks to the notion of quasi-reversibility. However, all the models discussed in [14] have pairwise synchronisations, that is, an event depends on and can change the state of at most two components of the stochastic network. More complicated types of synchronisations in product-form have been firstly proposed by Gelenbe in the context of G-networks as shown in [7], where an event can change the state of many G-queues of the system instantaneously. This type of synchronization has been generalized in [12] with an extension of RCAT [10] and discussed in the context of G-networks in [16].

In our model, while we still have the case in which at most two stations change their states at a transition event, these are chosen according to the global state of the network. To the best of our knowledge, the closest behaviour is proposed by Pittel [22] in networks of finite capacity queues with independent probabilistic routing. In this model, once a job arrives at a saturated station, it instantaneously skips to the next one according to the routing matrix. Therefore, a job skips the service at a station as it happens in our model, although the motivations of this behaviour are quite different: we aim at maximising the system throughput in infinite capacity stations, while Pittel's policy is introduced to handle finite capacity queues. Since its introduction, skipping policy for finite capacity queues has found numerous applications beyond the analysis of computer systems (see, e.g., [28] and the references therein). Further insights on the relation between the model we propose here and Pittel's queueing networks are given in [20].

To give another example of a situation where our proposed model may apply, let us consider a cluster where the system workload consists of a series of mandatory and optional tasks to be processed, such as the real-time and multimedia systems as described in [19]. This system can be modeled as a group of stations, denoted as S_i for $i = 0, \dots, K$, where each station is composed of a warehouse with its corresponding server. A job execution consists of a mandatory task and an optional task. Warehouse 0, denoted as W_0, is responsible for generating and processing optional tasks, while all other stations are equipped with more advanced processing features (e.g., more memory, more compute capacity) dedicated to the execution of mandatory tasks. When a station S_i with $i > 0$ becomes idle, it retrieves either a mandatory task from one of its peers or an optional task on W_0 terminates and a mandatory task commences in its place at station i, based on the routing probabilities.

As a further example, we can think of a communication line connecting nodes, typically traversed in a predetermined sequence to optimize packet transmission. Each node in the system is collecting data, and upon reaching a specified timeout, it transmits the accumulated data to a neighbor node. If the timeout expires and a node has no data to transmit, it requests transmission from a preceding node, continuing this process recursively until a node with pending data is identified. This process, however, comes at the expense of increased energy consumption, as the transmission cost rises with the distance the fetching request must travel to receive the transmission.

The model we propose also has a practical application as in the paper of Gates and Westcott [5, 29]. In these works, the authors devise an interesting stochastic process whose aim is to optimise the costs associated with the management of the system handling wheel replacement and restoration on the trains of Queensland Railways. To obtain the expression of the product-form equilibrium distribution, they introduce the theory of dynamically reversed processes, and show that, under a proper state renaming function, the process satisfies the conditions for an exact analysis. In practice, Gates and Wescott's model is a particular instance of our result (see Section 3.6) where the topology is just a tandem of stations. Furthermore, resorting to dynamically reversed argument is not modular since the entire stochastic process must be considered in a monolithic way and requires a guess on the state renaming function which is, in general, not trivial. The originality of the model in the field of product-forms relies also on the fact that its underlying stochastic process is neither reversible nor quasi-reversible.

Recently, many works have studied job stealing policies both with exact [17] and approximate or mean-field based techniques (see, e.g., [4, 15, 25]). From the modelling point of view, all mentioned works study strategies aimed at load balancing and not at throughput maximization, so they do not rely on the idea of skipping a service stage, but to move a job to a functionally equivalent server with lower workload. With respect to [4, 15, 25], our product-form result does not require the assumption of many servers. The result proved in [17] is a product-form in which jobs are stolen in batches of truncated geometric sizes with the aim of making the load factor of stations balanced. Therefore, the routing policy is totally different from the one shown here.

3 THE OPEN MODEL

In this section, we formally introduce the model and study its solution for the open case. Section 4 generalizes the results to closed systems.

3.1 Informal Model Description

We consider K warehouses with infinite capacity, indexed $1, \ldots, K$, each one associated with a service room where jobs are processed. In addition, a warehouse denoted by 0, that is always full is present. We prefer to use the terms warehouse instead of queue for clarity: when we refer to an *empty warehouse*, we mean that the station buffer is empty but there is still a job in service, as in this model service rooms are never idle. In contrast, the notion of *empty queue* often refers to the situation in which the server is idle.

The state of the network is a vector $\mathbf{n} = (n_1, \ldots, n_K) \in \mathcal{S}_K$ describing the occupancy of the warehouses, where $\mathcal{S}_K = \mathbb{N}^K$ is the state space with $K \in \mathbb{N}, K \geq 1$. Let us denote a $K-$dimension vector of zeros with a 1 in position i by $\mathbf{e}_i$, where the value of K will be clear from the context. Finally, we say station i is a *provider* of station j if the probability of going from i to j is greater than 0. Each warehouse $i > 0$ is associated with a server that behaves as follows:

— If warehouse i is not empty (recall that this is always the case for server 0), then, after an exponentially distributed time with rate μ_i, it sends to warehouse $j > 0$ the job it is serving with probability p_{ij} and then it picks another job from warehouse i. Therefore, the state changes from $\mathbf{n}$ to $\mathbf{n} - \mathbf{e}_i + \mathbf{e}_j$. With probability $p_{i0} = 1 - \sum_{j=1}^{K} p_{ij}$ the server removes a job from its warehouse and disposes it. In this case, the state changes from $\mathbf{n}$ to $\mathbf{n} - \mathbf{e}_i$.

— If warehouse i is empty, at service completion, the server contacts one of its providers and fetches a job from its warehouse. The provider is chosen probabilistically: server $k \geq 0$ is chosen with probability:

$$q_{ik} = \frac{\mu_k p_{ki}}{\sum_{h=0}^{K} \mu_h p_{hi}}, \tag{1}$$

which states that the provider is chosen with a probability that is proportional to the intensity of the stream directed to the empty warehouse that experiences a service completion. If the contacted warehouse is empty, then its server instantaneously fetches a job from one of its providers applying the same policy. The process terminates when a non-empty provider is found (possibly provider 0). If $k > 0$ is the provider with non-empty warehouse, then the state changes from $\mathbf{n}$ to $\mathbf{n} - \mathbf{e}_k + \mathbf{e}_j$. If $k = 0$, then the state changes form $\mathbf{n}$ to $\mathbf{n} + \mathbf{e}_j$. In both cases, the destination warehouse j is chosen, as before, according to the routing probabilities.

Also in this case, if $\sum_{j=1}^{K} p_{ij} < 1$, then the job departing from server i may be disposed with probability $p_{i0} = 1 - \sum_{j=1}^{K} p_{ij}$. Notice that it may be the case that after an exponential time with rate μ_i there is no state change because the job served by server i is disposed and the new one is taken from warehouse 0.

Server 0 fetches jobs from its always full warehouse and, upon a service completion, sends a job to one of the other K warehouses with rate μ_0. The destination warehouse is chosen probabilistically according to vector $\mathbf{P}_0 = (p_{01}, \dots, p_{0K})$, with $\sum_{k=1}^{K} p_{0K} = 1$, p_{0k} being the probability of choosing warehouse k. The scheduling discipline is First-Come First-Served.

3.2 The Provider Choice Probability

In this subsection, we study the probability $\alpha_{ij}(\mathbf{n})$ that server i, upon terminating the work on a job, fetches a job from warehouse j, with $j = 0, \dots, K$ given the state of the system is $\mathbf{n}$. We introduce an intermediate vanishing state $\mathbf{m}$ with the purpose of formally specifying the state-dependent routing. Suppose that the completed job at warehouse i migrates to warehouse k, and let $\mathbf{m} = \mathbf{n} + \mathbf{e}_k$. If the job leaves the system, then $\mathbf{m} = \mathbf{n}$. Then, we have:

$$\alpha_{ij}(\mathbf{m}) = \begin{cases} 1 & \text{if } j = i \wedge m_i > 0\,, \\ \sum_{\ell=0}^{K} q_{i\ell}\alpha_{\ell j}(\mathbf{m}) & \text{if } m_i = 0\,, \\ 0 & \text{if } j = i \wedge m_i = 0\,. \end{cases}$$

where q_{ik} has been defined in Equation (1) and $m_0 > 0$. In other words, $\alpha_{ij}(\mathbf{m})$ corresponds to the probability of absorption in state j starting from state i in a discrete time Markov chain where the absorbing states correspond to the non-empty warehouses and the transition probabilities outgoing the non-absorbing states are given by Equation (1). Recall that $\mathbf{m}$ is chosen assuming that the output of warehouse i goes to warehouse k or leaves the system, and this happens with probability p_{ik} or p_{i0}, respectively.

3.3 Underlying Continuous Time Markov Chain

The stochastic process underlying the network presented in Section 3.1 is a CTMC since all the arrival and service times are independent and exponentially distributed random variables and the state of the process $\mathbf{n} \in \mathcal{S}_K$ is sufficient to describe the system dynamics. Let $X(t)$ be this Markov process, then its transition rates are:

— From state $\mathbf{n}$ to state $\mathbf{n}+\mathbf{e}_i-\mathbf{e}_j$, with rate $\sum_{k=1}^{K} \mu_k p_{ki} \alpha_{kj}(\mathbf{m})$, $1 \le i, j \le K$ and $\mathbf{m} = \mathbf{n}+\mathbf{e}_i$. In this case, we have a reduction of items in warehouse j and an increment of those in i whenever one of the servers k that is a provider of i ($p_{ki} > 0$) fetches an item from warehouse j, which occurs with probability $\alpha_{kj}(\mathbf{m})$. Notice that, in the case of a topology including feedback, it may happen that $i = j$ and hence no state change actually occurs.

— From state $\mathbf{n}$ to state $\mathbf{n} - \mathbf{e}_j$ with rate $\sum_{k=1}^{K} \mu_k p_{k0} \alpha_{kj}(\mathbf{n})$, $1 \le j \le K$. This case is similar to the previous one, but the item that leaves station k is disposed. Henceforth, we use p_{k0} to denote $1 - \sum_{h=1}^{K} p_{kh}$.

— From state $\mathbf{n}$ to state $\mathbf{n} + \mathbf{e}_i$ with rate $\sum_{k=0}^{K} \mu_k p_{ki} \alpha_{k0}(\mathbf{m})$, $1 \le i \le K$ and $\mathbf{m} = \mathbf{n} + \mathbf{e}_i$. In this case, the number of items of warehouse i increases, but the item is fetched from warehouse 0 which, being always full, is not represented in the process state.

Observation 1 (Irreducibility of X(t)). *$X(t)$ is irreducible if the routing matrix, considering the outside as warehouse 0, is irreducible.*

This observation is not surprising and it is analogous to the condition of irreducibility for Jackson's networks. Henceforth, we assume the irreducibility of $X(t)$.

Theorem 1. *Let $X(t)$ be the irreducible CTMC underlying a fetching queueing network, then its invariant measure has product-form:*

$$\pi(\mathbf{n}) \propto \prod_{i=1}^{K} \rho_i^{n_i}, \tag{2}$$

where:

$$\rho_i = \frac{\sum_{k=0}^{K} \mu_k p_{ki}}{\mu_i p_{i0} + \sum_{k=1}^{K} x_{ik}}, \tag{3}$$

and x_{ik} is the solution the linear system of rate equations:

$$x_{ik} = \left(\sum_{j=1}^{K} x_{kj} + \mu_k p_{k0} \right) \frac{\mu_i p_{ik}}{\sum_{j=0}^{K} \mu_j p_{jk}}. \tag{4}$$

Proof. In Appendix A, we provide a brief review of the RCAT and its terminology that is inherited from the **stochastic process algebra (SPA)** domain.

The proof consists of two steps. First, we give the modular representation of the network using the synchronisations used by RCAT. Then, we show that the general single component satisfies the three conditions of the theorem and hence we have the product-form solution of Equation (2). We choose a graphical description of the component, rather than a process algebraic one, to make the proof easier to read. Notice that, similarly to birth and death processes, the only possible transitions are between adjacent states, and we call birth transitions those from state n to state $n+1$ and death transitions those from state $n+1$ to n, with $n \geq 0$. From now on we will refer to *active* and *passive* actions according to PEPA [13] notation. An active action is one that has a specific exponential rate and is initiated by a component within the system. It informally represents a proactive behaviour where a component actively engages in an event. Conversely, passive actions are not actively triggered but are typically executed concurrently and/or as a consequence of another action. To identify passive actions, they are denoted by the symbol $\top$ as their action rate, indicating that the latter depends on another action. These two concepts are fundamental in PEPA for modelling and analyzing the performance of concurrent systems.

We start by detailing the process of Figure 3 to show that it is an exact representation of the behaviour of each station described so far.

— Label a_{ji} describes arrivals at warehouse i, it is designed as an active action type and it is performed with rate $\mu_j p_{ji}$. Since this action describes a new arrival, it will increase by one the state of the process. Notice that label a_{ji} will describe all the possible arrivals at the warehouse meaning that as far as $p_{ji} > 0$, the corresponding label a_{ji} will exist.

— Label a_{ih} describes fetching actions from warehouse i, it is designed as a passive action type with unknown rate $\top_{ih}$. Once the components are studied in isolation, these unknown rates are then instantiated with x_{ih}, corresponding to the constant reversed rate of the synchronising action type. Intuitively, this action type may synchronize with an action describing server i fetching a new job from its own warehouse, upon job completion, or with a fetching action, always upon job completion, from a server with an associated empty warehouse for which warehouse i is a provider. As before, for every $p_{ih} > 0$ an a_{ih} label will exists. As proved in [10], the reversed rate x_{ih} is computed as the sum of the rates of the death transitions in the isolated component W_h, distributed amongst the birth transitions in proportion to their forward transition rates.

— Label a_{i0} describes departures from the system, a certain process underlying a warehouse i will have transitions labelled with this particular action type if and only if $p_{i0} > 0$, that is, if from station i a job can be disposed from the system.

— As far as the loop on state 0 is concerned, we have what follows: label 1. describes a fetching action from warehouse j, as a matter of fact the action type is labelled a_{ji}, and this action will happen at the rate of the possible completion from the system multiplied by the probability of choosing exactly warehouse j as provider. The self-loop transition is needed to describe that the fetched job will go directly in service, consequently the warehouse will remain in the same state. Label 2. also describes a fetching action exactly as before. Intuitively, W_i is receiving a fetching request from W_h which it is not able to fulfill and hence it instantaneously propagates the request to provider W_j with probability q_{ij}. Notice that W_j will recursively behave analogously to W_i. This semantics is denoted by $\longrightarrow$ as in [11]. Studying the isolated components requires to instantiate the unknown rate $\top_{ih}$ with x_{ih} as described before.

We prove that the process underlying component W_i, depicted in Figure 3, fulfills the three conditions of the RCAT:

(i) If a synchronising type is passive in a component then all states of that component must have an outgoing transition labelled with that action type; this is true for component W_i as a_{ih} is the only passive synchronising type and we have an outgoing transition labelled a_{ih} for every state of the component.

(ii) If a synchronising type is active in a component then all the states of that component must have an incoming transition with that type; this holds for component W_i as a_{ji} is the only active synchronising action type and we have an incoming transition labelled a_{ji} for every state of the component as in state 0 we have a self-loop labelled directly with a_{ji} if there is a departure from the system or with a label describing the propagation of a passive action type as active action type a_{ji}.

(iii) All transitions sharing the same active synchronising type must have the same reversed rate. In station W_i the transitions sharing the same active synchronising type are all the birth transitions and label 2. of the self-loop on state 0. So, for birth transitions we have the following reversed rate

$$x_{ji} = \left(\mu_i p_{i0} + \sum_{h=1}^{K} x_{ih}\right) \cdot \frac{\mu_j p_{ji}}{\sum_{k=0}^{K} \mu_k p_{ki}} ,$$

that is, the sum of the rate with which a job possibly departs from the system plus the sum of the rates with which a job departs from station i to any other possible station. This sum is then multiplied for the probability of choosing exactly provider j. Now, we analyse the reversed rate, x_{ih}^R, of the propagating transition of the self-loop on state 0 and we obtain

$$\begin{aligned} x_{ih}^R &= \mu_i p_{i0} q_{ij} + \sum_{h=1}^{K} x_{ih} \cdot q_{ij} \\ &= \left(\mu_i p_{i0} + \sum_{h=1}^{K} x_{ih}\right) \cdot q_{ij} , \end{aligned}$$

meaning that the reversed rate of x_{ih} is given by the sum of the rate with which a job possibly leaves the system multiplied by the probability of choosing a particular provider j with the sum of all the possible rates of action types describing the departure from station i to any other possible station multiplied for the probability of choosing a particular provider. Notice

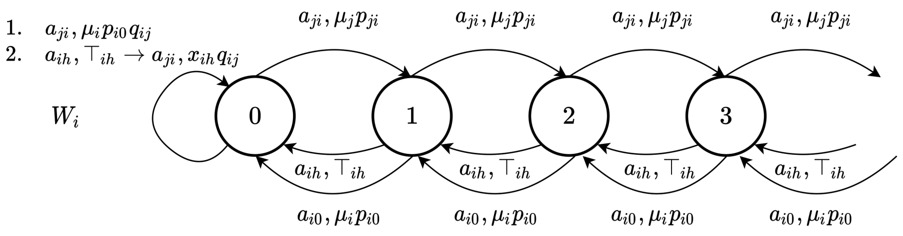
1. $a_{ji}, \mu_i p_{i0} q_{ij}$
2. $a_{ih}, \top_{ih} \to a_{ji}, x_{ih} q_{ij}$
W_i
0
1
2
3
$a_{ji}, \mu_j p_{ji}$
$a_{ih}, \top_{ih}$
$a_{i0}, \mu_i p_{i0}$

Fig. 3. Generalization of the processes underlying a generic warehouse W_i.

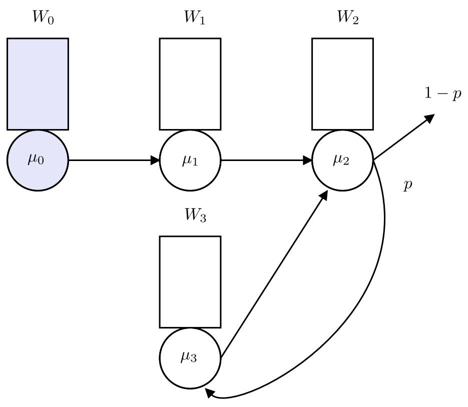

Fig. 4. Example of stochastic network. Warehouses are denoted by labels $W_0, \ldots, W_3$, recall that W_0 is a special warehouse which always contains at least one job waiting to be served.

that $q_{ij} = \dfrac{\mu_j p_{ji}}{\sum_{k=0}^{K} \mu_k p_{ki}}$, consequently we have

$$x_{ih}^R = \left(\mu_i p_{i0} + \sum_{k=1}^{K} x_{ik}\right) \cdot \frac{\mu_j p_{ji}}{\sum_{k=0}^{K} \mu_k p_{ki}} = x_{ji} \,.$$

Therefore, the three conditions of the RCAT are satisfied and accordingly we are able to compute the product-form. □

Corollary 1. *$X(t)$ is ergodic if and only if it is irreducible and $\rho_i < 1$ for all $i = 1, \ldots, K$.*

Corollary 2. *If $X(t)$ is ergodic, then its steady-state distribution is:*

$$\pi(\mathbf{n}) = \prod_{i=1}^{K} (1 - \rho_i)\rho_i^{n_i} \,, \tag{5}$$

3.4 Example

Let us consider the network of Figure 4. To emphasize the different behaviour of this fetching network compared to Jackson's networks, we represent the warehouse on the top of the corresponding server, rather than on its left. Observe that, when Server 2 releases the job in the service room, it may either dispose it with probability $1 - p$ or send it to Server 3. If its warehouse W_2 is empty, it sends a request either to Server 1 or 3. This probabilistic choice is done in a very natural way according to the speed of the servers, that is, W_1 server is chosen with probability $q_{21} = \mu_1/(\mu_1 + \mu_3)$ and W_2 server with probability $q_{23} = \mu_3/(\mu_1 + \mu_3)$. It is worth noticing that if

Server 2 moves a job to W_3 and W_2 is empty, the same job may be re-taken by Server 2 because of the loop between these two stations.

According to the specification we have just given, for this example, we have the following parameters:

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & p \\ 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{P}_0 = [1\ 0\ 0]$$

In Figure 4, we show the network configuration of this particular study case, then in Figure 5, we give the graphical representation of the processes, divided in isolated components, underlying the warehouses of the network. Notice that the only variables x_{ik} whose value is different from 0 are those associated with $p_{ik} > 0$. The system of rate equations is:

$$\begin{cases} x_{12} = (x_{23} + \mu_2(1-p))\dfrac{\mu_1}{\mu_1 + \mu_3} \\ x_{23} = x_{32} \\ x_{32} = (x_{23} + \mu_2(1-p))\dfrac{\mu_3}{\mu_1 + \mu_3} \end{cases}$$

whose solution is:

$$x_{12} = (1-p)\mu_2\,, \quad x_{23} = x_{32} = \frac{(1-p)\mu_3\mu_2}{\mu_1}\,.$$

Starting from the representation in Figure 5, we check if the structural and rate conditions of the RCAT are fulfilled.

For structural conditions it is required that if a synchronizing type is active in a component then all the states of that component must have an incoming transition with that type and if it is passive then every state should have an outgoing transition labelled with that action type. We can state that structural conditions are satisfied as follows:

- a_{01} is passive in W_0 and we have an outgoing arc for every state labelled a_{01};
- a_{01} is active in W_0 and we have an incoming arc for every state labelled a_{01}, thanks to the propagating transition of the self-loop in state 0;
- a_{12} is passive in W_1 and we have an outgoing arc for every state labelled a_{12};
- a_{12} and a_{32} are active in W_2 and we have for each state an incoming arc labelled a_{12} and another one labelled a_{32};
- a_{23} is passive in W_2 and we have for each state an outgoing arc labelled a_{23};
- a_{23} is active in W_3 and we have for each state an incoming arc labelled a_{23}, thanks to the propagating transition of the self-loop in state 0;
- a_{32} is passive in W_3 and we have for each state an outgoing arc labelled a_{32}.

As far as rate condition is concerned, RCAT requires that all transitions sharing the same active synchronizing type must have the same reversed rate. For component W_1 we have active action type a_{01} whose reversed rate is x_{12} for every birth transition in this component, this must be equal to the reversed rate of the propagating transition on the self-loop on state 0 but the reversed rate of a self-loop is the rate itself that is, x_{12} in this case. Consequently, the condition is satisfied for component W_1. The exact same reasoning can be applied to component W_3 as active action type a_{23} has reversed rate x_{32} for every birth transition and this equals the reversed rate of the propagating transition on the self-loop in state 0. Accordingly, also in this component the rate condition is satisfied.

For component W_2 the reasoning is slightly more complex: we must check that the reversed rates of active action types labelled a_{12} and a_{32} are always the same, for all birth transitions this is true as in each one we have:

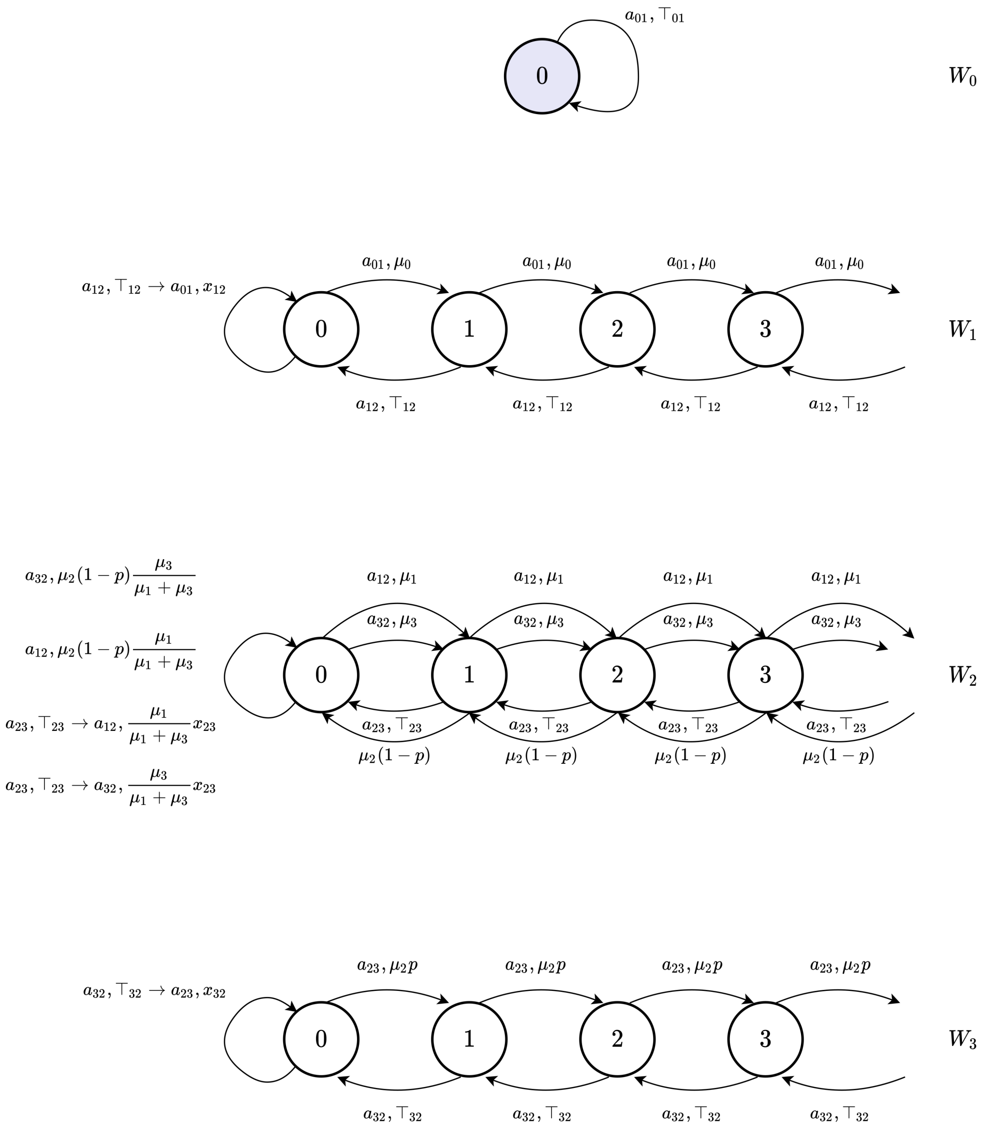
$a_{01}, \top_{01}$
W_0
$a_{12}, \top_{12} \to a_{01}, x_{12}$
a_{01}, μ_0
$a_{12}, \top_{12}$
W_1
$a_{32}, \mu_2(1-p)\frac{\mu_3}{\mu_1+\mu_3}$
$a_{12}, \mu_2(1-p)\frac{\mu_1}{\mu_1+\mu_3}$
$a_{23}, \top_{23} \to a_{12}, \frac{\mu_1}{\mu_1+\mu_3}x_{23}$
$a_{23}, \top_{23} \to a_{32}, \frac{\mu_3}{\mu_1+\mu_3}x_{23}$
a_{12}, μ_1
a_{32}, μ_3
$a_{23}, \top_{23}$
$\mu_2(1-p)$
W_2
$a_{32}, \top_{32} \to a_{23}, x_{32}$
$a_{23}, \mu_2 p$
$a_{32}, \top_{32}$
W_3
0
1
2
3

Fig. 5. Graphical representation of the processes underlying the warehouses of Figure 4.

$$x_{12} = (\mu_2(1-p) + x_{23})\frac{\mu_1}{\mu_1 + \mu_3},$$
$$x_{32} = (\mu_2(1-p) + x_{23})\frac{\mu_3}{\mu_1 + \mu_3}.$$

These must be equal to the reversed rates of the propagating transitions in the self-loop, this is proved by summing the rates of the labels sharing the same action type, consequently we have:

$$x_{12} = \frac{\mu_1}{\mu_1 + \mu_3}x_{23} + \mu_2(1-p)\frac{\mu_1}{\mu_1 + \mu_3} = (\mu_2(1-p) + x_{23})\frac{\mu_1}{\mu_1 + \mu_3},$$
$$x_{32} = \frac{\mu_3}{\mu_1 + \mu_3}x_{23} + \mu_2(1-p)\frac{\mu_3}{\mu_1 + \mu_3} = (\mu_2(1-p) + x_{23})\frac{\mu_3}{\mu_1 + \mu_3},$$

respectively. Accordingly we can state that also in component W_3 the rate condition is fulfilled and with this we have proved all the RCAT conditions holds; consequently, we can obtain ρ_i with Equation (3):

$$\rho_1 = \frac{\mu_0}{(1-p)\mu_2}\,, \quad \rho_2 = \frac{\mu_1}{(1-p)\mu_2}\,, \quad \rho_3 = \frac{p\mu_1}{(1-p)\mu_3}\,.$$

As we can see from the expression of ρ_1 above, it may happen that the ρ_i of a station does not depend directly from the corresponding service rate μ_i, as its effect vanishes. Assuming the stability $\rho_i < 1$ for $i = 1, 2, 3$ we have the steady-state distribution:

$$\pi(n_1, n_2, n_3) = \prod_{i=1}^{3} (1 - \rho_i)\rho_i^{n_i}\,.$$

3.5 Comparison with Reversibility and Kelly's Quasi-reversibility

Most product-forms are related to stochastic reversibility of the Markov chain underlying the model. As far as our model is concerned, it is easy to show a counter-example proving that this is not the case. Consider to find the example system of Figure 4 in state $(1, 1, 1)$. With rate μ_1 the state moves to $(0, 2, 1)$. However, there exists no transition bringing the system from state $(0, 2, 1)$ back to state $(1, 1, 1)$ which is a necessary (but not sufficient) condition for stochastic reversibility.

For what concerns the comparison with Kelly's quasi-reversibility, we observe that the type of state-dependent routing characterizing our model is different from the one assumed in quasi-reversible queueing networks. In fact, in the latter case, every transition depends on the state of at most two components. Let us consider the tandem topology shown in Figure 2, and assume that warehouse k is empty. With rate μ_k a job is stolen from a warehouse $k' < k$. In this example, k' is the largest index of the non-empty warehouses in $\{0, 1, \ldots, k-1\}$ hence, the transition depends on all the states of the warehouses with indices lower than k.

3.6 Comparison with the Gates-Westcott Model

In [5, 29], the authors devised a particular Markov process for modelling the process of wheel replacement and restoration on the trains of Queensland Railways. In this framework, each different train requires all its wheels to be of a particular size i and all trains should be operable at any time. Because of continuous use, train wheels get worn; when the wear becomes excessive the wheel is removed and replaced from stock, whereas worn wheels get re-profiled by grinding and go into stock so that they can be used again in trains that need wheels with a smaller diameter. Once a wheel becomes too small, it is discarded. Notice that re-profiling leads to wheels of various sizes both on trains and in stock. The aim of [5, 29] was to optimize the management of the system given the associated costs. In those works, the stock occupation level at time t is a random process $\boldsymbol{X}(t) = (x_2(t), \ldots, x_M(t))$, where $x_i(t)$, $(i = 2, \ldots, M)$ is the number of size i wheels in stock at time $t > 0$. Notice that $i = 1$ denotes wheels of type 1, which is the type used to represent new wheels. Moreover, the authors assume to have an unlimited supply of new wheels, which are not held in stock and this is the reason why the random process describing the stock occupation level is indexed starting from $i = 2$. Changes of state are caused by the arrival of a train with worn wheels. The strategy, followed in [5, 29], to obtain the equilibrium distribution, if any, of $\boldsymbol{X}(t)$ was to dynamically reverse the process and to apply a generalization of the classical Kolmogorov cycle. Notice, however, that the derivation of this product-form is rather complex as it resorts to detailed balance conditions and Kolmogorov's criterion. Moreover, transposing this model in the queueing network context, the system encompasses only feed-forward tandem configurations, as the one of Figure 2.

In the Discussion Section of [5], the authors provide insights into potential model variants to be investigated, which may be more realistic in real-world applications. The first variant implies a fixed number of wheels K as the model population. The latter can be encompassed in the closed system framework we are going to define in Section 4. In addition, the authors suggest an alternative fetching mechanism, enabling one to fetch from different sources according to different probabilities. This suggestion aligns with our extension of the model to accommodate various network topologies that can also include branches, confluences, and cycles. It is worth mentioning that the approach used for the solution of the model of our contribution and [5] are very different. Indeed, when we allow arbitrary routing matrices, we can ensure the product-form solution only by imposing the fetching probability on the base of the service rates of the providers. This constraint in the model becomes relatively easy to derive thanks to the modular analysis provided here but has probably represented a major difficulty for the monolithic one of [5]. In conclusion, with respect to [5], in our model arbitrary topologies are allowed, both open and closed, and the routing can admit feedback.

4 CLOSED SYSTEMS

The proposed model is also applicable in the context of closed systems. Let the system consists of $N + K$ jobs and K stations, with $N > 0, K > 0$. With respect to the model studied until now, we have the following differences: jobs disposal from the system is not allowed and the number of jobs is constant. In addition, in a closed setting there is no warehouse that is always full, as it was the case of W_0 in the open setting. Moreover, with respect to traditional Gordon-Newell closed networks, we assume that every station always has a job in service, consequently the system state composed of K zeros actually describes a system where every station has a job in service and no job is waiting to be served. Accordingly, in this case the only state changes accepted are:

— $\mathbf{n} - \mathbf{e}_i + \mathbf{e}_j$, if warehouse i is not-empty and if server i sends the served job to warehouse j;
— $\mathbf{n} - \mathbf{e}_k + \mathbf{e}_j$ if warehouse i is empty and its server sends the served job to warehouse j and retrieves another job to serve from warehouse k according to the same provider choice policy we have described in Section 3.1.

As we already know, in closed networks the queue length distribution of any station i does not correspond to term ρ_i as for open networks, but it is derived from the joint queue length distribution $\pi(\mathbf{n})$ given by the following formula

$$\pi(\mathbf{n}) = \frac{1}{G(N,K)} \prod_{i=1}^{K} g_i(n_i)\,. \tag{6}$$

This requires the computation of functions g_i for each station and also of the normalizing constant G that guarantees $\sum_{\mathbf{n}\in\mathcal{S}} \pi(\mathbf{n}) = 1$, where $\mathcal{S}$ is the state space of the system. In our case, we have:

$$g_i(n_i) = \rho_i^{n_i}\,, \tag{7}$$

where the term ρ_i is defined as in Equation (3) and the normalising constant can be computed by the convolution algorithm [2]. We consider a closed network with N jobs and K stations, for such a system the normalizing constant using convolution algorithm is computed as follows. Let $\mathcal{K}_1, \mathcal{K}_2$ be a non-trivial partition of the set of stations ($\mathcal{K}$) then we have:

$$G(N,\mathcal{K}) = \sum_{n=0}^{N} G(N-n,\mathcal{K}_1)G(n,\mathcal{K}_2)\,.$$

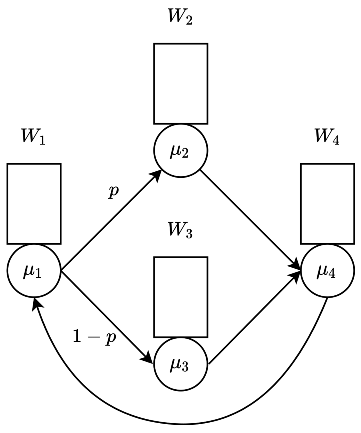

Fig. 6. Example of closed stochastic network. Warehouses are denoted by labels $W_1, \ldots, W_4$.

Using this expression, one may add to the model one station at the time. Now, let $\pi_i(n)$ be the marginal steady-state probability of observing n jobs in warehouse i. By definition, this probability is given by the following equation:

$$\pi_i(n) = \frac{1}{G(N, \mathcal{K})} \sum_{\mathbf{n} \in \mathcal{S}: n_i = n} \prod_{j=1}^{K} g_j(n_j) = \frac{G(N-n, \mathcal{K} \setminus \{i\})}{G(N, \mathcal{K})} g_i(n) \,.$$

This quantity is particularly important as it allows us to compute the probability of fetching, which is useful to determine the throughput at each queue as we will see in Section 5.

4.1 Example

We consider the model in Figure 6. Accordingly, for this example we have the following parameters:

$$\mathbf{P} = \begin{bmatrix} 0 & p & 1-p & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

As in Example 3.4, in Figure 7 we can observe the graphical representation of the processes, divided in isolated components, underlying the warehouses of the closed network under study. The system of rate equation is:

$$\begin{cases} x_{41} = x_{12} + x_{13} \\ x_{12} = x_{24} = x_{41} \dfrac{\mu_2}{\mu_2 + \mu_3} \\ x_{13} = x_{34} = x_{41} \dfrac{\mu_3}{\mu_2 + \mu_3} \end{cases}$$

We know that for closed networks, this system is under-determined and that all the (possibly infinite) non-trivial solutions will differ by a multiplicative non-zero factor. Therefore, we fix x_{41} to 1 and obtain solution

$$x_{41} = 1, \qquad x_{12} = x_{24} = \frac{\mu_2}{\mu_2 + \mu_3}, \qquad x_{13} = x_{34} = \frac{\mu_3}{\mu_2 + \mu_3} \,.$$

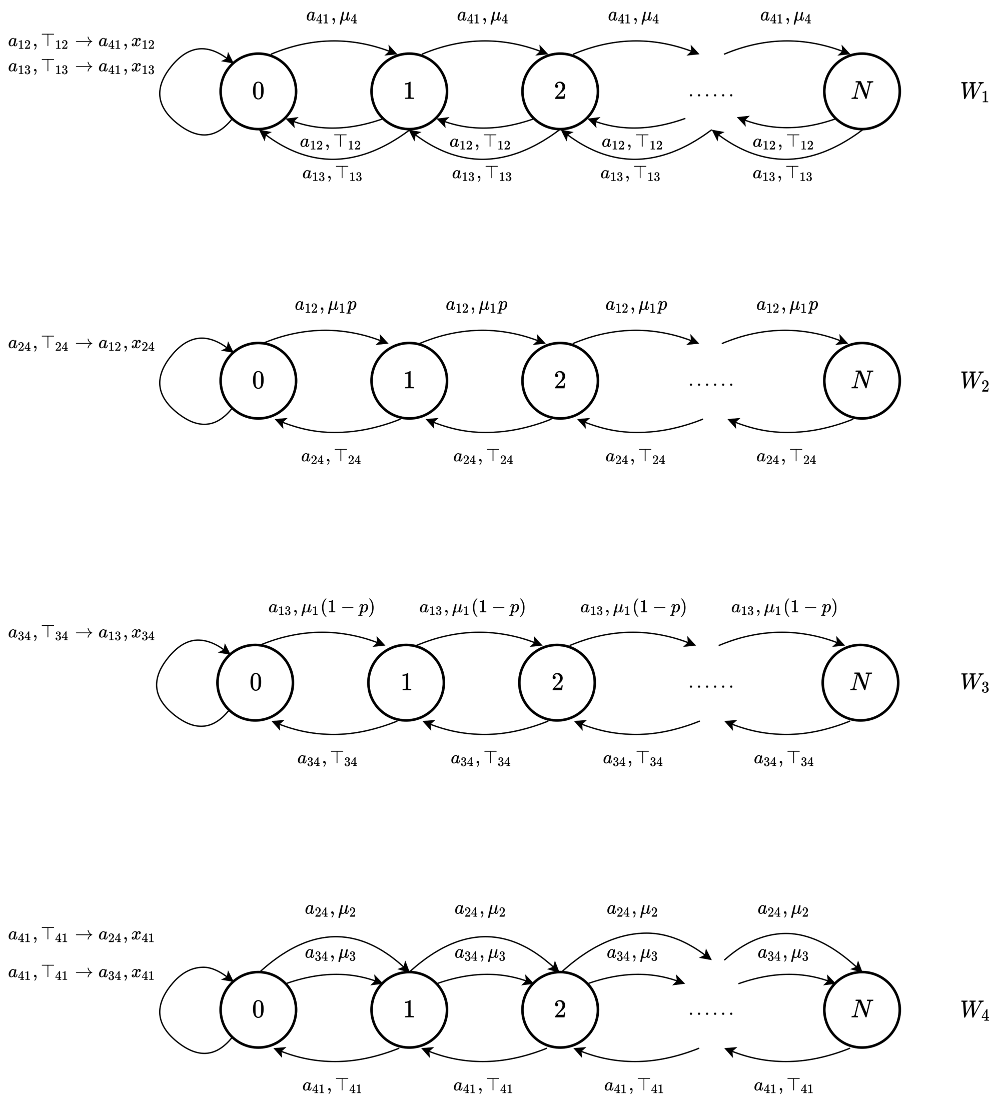
$a_{12}, \top_{12} \to a_{41}, x_{12}$
$a_{13}, \top_{13} \to a_{41}, x_{13}$
a_{41}, μ_4
0
1
2
......
N
W_1
$a_{12}, \top_{12}$
$a_{13}, \top_{13}$
$a_{24}, \top_{24} \to a_{12}, x_{24}$
$a_{12}, \mu_1 p$
W_2
$a_{24}, \top_{24}$
$a_{34}, \top_{34} \to a_{13}, x_{34}$
$a_{13}, \mu_1(1-p)$
W_3
$a_{34}, \top_{34}$
$a_{41}, \top_{41} \to a_{24}, x_{41}$
$a_{41}, \top_{41} \to a_{34}, x_{41}$
a_{24}, μ_2
a_{34}, μ_3
W_4
$a_{41}, \top_{41}$

Fig. 7. Graphical representation of the processes underlying the warehouses of Figure 6.

Accordingly, we obtain for Equation (6):

$$g_1(n_1) = \mu_4^{n_1}, \qquad g_2(n_2) = \left(\frac{\mu_1 p(\mu_2 + \mu_3)}{\mu_2}\right)^{n_2},$$

$$g_3(n_3) = \left(\frac{\mu_1(1-p)(\mu_2 + \mu_3)}{\mu_3}\right)^{n_3}, \qquad g_4(n_4) = (\mu_2 + \mu_3)^{n_4}.$$

5 AVERAGE STATIONARY PERFORMANCE INDICES

As far as performance indices are concerned, given Equations (5) and (6), it should be clear that the methods for the derivation of the expected occupancy is similar to the ones used for Jackson and Gordon-Newell networks, respectively. This means that for open networks we will have:

$$\overline{N}_i = \frac{\rho_i}{1-\rho_i}\,,$$

and in closed networks we observe:

$$\overline{N}_i = \sum_{n=1}^{N} n\pi_i(n) = \sum_{n=1}^{N} g_i(n)\frac{G(N-n,\mathcal{K})}{G(N,\mathcal{K})}\,.$$

It is actually not surprising that the equations to compute the mean number of jobs in each warehouse do not change with respect to the more traditional models cited above, as the main difference between these models and our model resides in the fetching policy. However, this mechanism does not affect the destination warehouse population and this is because fetched jobs go directly in service without waiting in the warehouse of the fetching station.

The precise definition of station throughput requires special consideration. It involves identifying two distinct flows originating from a station: one comprising all the jobs traversing that particular station, independently of whether they actually receive service in that station or not, and the other consisting of only those jobs that are effectively served in the station. We distinguish these notions by referring to the former as throughput and to the latter as *goodput*. In particular, jobs leave a station for two reasons: either because of job completion (and we identify this amount of jobs as *goodput*) or because they have been fetched by a server with an empty warehouse, and we denote this flow with F_i. We compute throughput X_i of a particular station i as the sum of these two quantities. Since servers are always busy, the goodput is μ_i for station i. Whereas, the direct computation of F_i is more complicated as we should be able to distinguish among the jobs waiting to be served in a warehouse i, between the ones that are going to be served in server i and the ones that are going to be fetched from some other server of the network. In order to do so, we should sum the probabilities, of each warehouse in the network, of being empty and observing a job completion in the associated servers. Each one of these probabilities, then should be multiplied by the probability of fetching a job exactly from warehouse i.

To simplify this computation, we can resort to a work-conservation argument. As a matter of fact, the total arrival intensity at a warehouse must equal its throughput, hence:

$$X_i = \sum_{k=0}^{K} \mu_k p_{ki} + \pi_i(0)\mu_i\,. \tag{8}$$

This formula expresses that at each queue the throughput is equal to the sum of the service rates of every other station in the network multiplied by its own routing probability of sending a job to station i, the one under analysis, plus the probability of station i of having an empty warehouse and observing a service completion from its server, situation that will lead server i to fetch a job from some other warehouse in the system. Notice that for closed systems:

$$\pi_i(0) = \frac{G(N,\mathcal{K}\setminus\{i\})}{G(N,\mathcal{K})}\,,$$

and for open networks $\pi_i(0) = (1-\rho_i)$. At this point, the estimation of F_i can be done by difference:

$$F_i = X_i - \mu_i = \sum_{k=1}^{K} \mu_k p_{ki} - (1-\pi_i(0))\mu_i\,. \tag{9}$$

6 MEAN VALUE ANALYSIS

MVA is an analysis technique for closed queueing networks [23]. Differently from convolution, it enables computing average performance indices without the need to compute the normalising constant. This leads to the definition of a recursive algorithm that is more numerically stable with respect to convolution, although they share similar asymptotic complexity.

The usual way to introduce MVA is the so called *Arrival Theorem* [24] but, given the state-dependent movement of jobs it would be difficult to transpose it in our context. Therefore, we will follow a different path. First, we introduce a *virtual* Gordon-Newell queueing network that contains $N - K$ jobs. Although this network has a different behaviour than the one we wish to study, its stationary distribution will correspond to the stationary distribution of the number of jobs in the warehouses of the original model. This virtual network will be solved with the standard equations of MVA and then we will map the average virtual performance indices into those defined in Section 4.

6.1 The Virtual Gordon-Newell Network

Suppose we have to solve a network with K stations and N jobs. A closed Gordon-Newell queueing networks consists of stations with exponentially distributed service time and independent probabilistic routing. At each job completion, jobs move to another station according to the probabilistic routing. The Gordon-Newell theorem states that this network has product-form solution that resembles Equation (6) where functions g are clearly defined in a different way, that is, $g_i^v(n_i) = (e_i^v/\mu_i^v)^{n_i}$, where e_i^v are some constants that depend on the routing and μ_i^v is the service rate.

We apply the standard algorithm for MVA, using as service demand $g_i^v(1) = g_i(1)$ as defined in Equation (7) for every station i. We also recall that we are interested in the number of jobs in the warehouses, since we know that one job is always in the service room. Thus, if the original network serves $N+K$ jobs, the virtual network serves N jobs. Without loss of generality, we consider station 1 as reference station, that is, $e_1^v = 1$. MVA computes the average number of jobs $\overline{N}_i^v$ at all stations of the virtual network and the throughput of the reference station X_1^v. By definition of the virtual network, we have $\overline{N}_i^v = \overline{N}_i$, for all i.

As for the throughput, the reasoning is not straightforward. The first problem is that of finding the throughput of station i in the virtual network. By the forced flow law [8], we have:

$$X_i^v = X_1^v e_i^v, \qquad \text{for } i = 1, \ldots, K\,.$$

Therefore, we can obtain the utilization of station i, that is, the fraction of time the virtual server is busy, thanks to the utilisation law [8]:

$$U_i^v = \frac{X_i^v}{\mu_i^v} = \frac{X_1^v e_i^v}{\mu_i^v} = X_1^v g_i^v(1) = X_1^v g_i(1), \qquad \text{for } i = 1, \ldots, K\,. \tag{10}$$

It is interesting to observe that we have obtained the expected occupancy and the utilisation for station i without defining e_i^v explicitly.

Noting that we have $\pi_i(0)^v = \pi_i(0) = 1 - U_i^v$, we can obtain the throughput of all stations i with Equation (8).

Coherently with the theory of MVA, the asymptotic complexity of this solution is $O(NK)$.

6.2 Example

We apply MVA on a traditional Gordon-Newell network and on a closed fetching network with $K = 4$ and compare the results as far as mean queue lengths and throughput at each node are concerned. Both networks share the same topology of the network in Figure 6, with $p = 1/7$, they

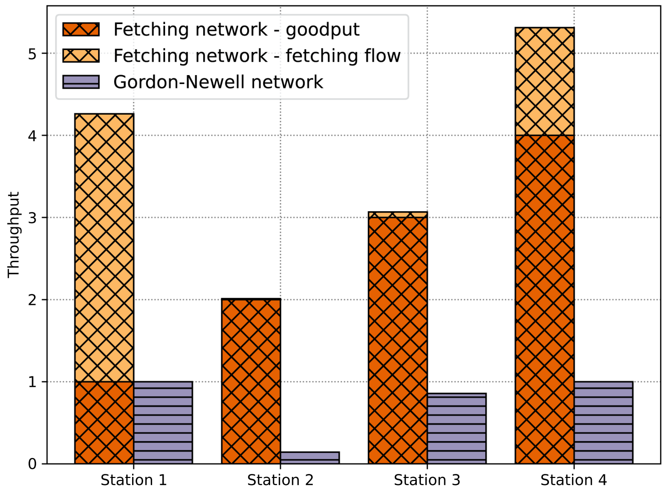

Fig. 8. Throughput at each station. For fetching networks, goodput and fetching flow are characterised.

Table 1. Comparison of Mean Queue Lengths in a Closed Fetching Queueing Network and in a Gordon-Newell Queueing Network Obtained through MVA

$\overline{N}_i$	Fetching	Gordon-Newell
Station 1	2.022	9.190
Station 2	0.070	0.077
Station 3	0.351	0.400
Station 4	3.556	0.333

also have the same service rates that is, $\mu_1 = 1.0$, $\mu_2 = 2.0$, $\mu_3 = 3.0$, $\mu_4 = 4.0$, and the same number of jobs in the network, $N = 10$.

These results reveal some interesting aspects of this specific model. In particular, in Figure 8, we can appreciate the advantages obtained by the fetching policy. It becomes evident that, for each station, throughput is maximized due to the number of jobs that, in Gordon-Newall networks, only move according to the probabilistic routing, resulting in periods when the service areas of the stations remain idle. Conversely, this does not happen in fetching networks because as soon as a service room finds itself idle and with no job waiting to be served in its corresponding warehouse the fetching mechanism will take place so that servers are never idle.

Furthermore, as shown in Table 1, the fetching policy significantly reduces the queue length at the first station. This reduction comes at the expense of a slight increase in the queue length at the fourth station. However, this trade-off is offset by the substantial increase in station throughput. Additionally, this policy enables a certain level of load distribution among the stations, leading to an overall expectation of improved waiting times per job.

7 CONCLUSION

In this work, we have introduced a novel class of product-form stochastic queueing network where server are never idle. This is achieved through a particular load balancing policy that allows the steal of jobs from other stations by those servers, that upon a service completion and a request for a new job from their own buffer, find the latter empty. This possibility of skipping some processing phases may degrade the quality of the final results, this is why this model suits fairly scenarios in which successive refinements on jobs improve the processes quality but are not rigorously to still obtain a sufficient result.

Notice that both in the case of open and close settings we were able to find exact solutions for these models. As far as the open system is concerned we were able to apply the multi-way extension of the RCAT and to retrieve accordingly its product-form solution. The latter has then been useful to retrieve the product-form also for closed systems, to which we were also able to apply the convolution and MVA algorithms in order to retrieve some important average performance indices.

As possible future research paths, we foresee the possibility of retrieving geometrical non-iterative bounds for the closed system, as convolution and MVA being exact methods do not provide great efficiency. Moreover, we aim to model also multi-class queueing networks applying the *fetching* policy we have introduced, as they would probably be more descriptive of and close to real world scenarios. Finally, looking ahead, future work may also investigate *hybrid* networks composed of both queues and warehouses and scenarios where stations are load-dependent.

8 DECLARATION OF COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

APPENDICES

A REVERSED COMPOUND AGENT THEOREM

A.1 Performance Evaluation Process Algebra

PEPA is a SPA. Process algebras [3] were first used as a modelling technique for the functional analysis of concurrent systems and then they became an established tool for performance and dependability analysis. One of the main attractive features of SPA is compositionality. From a practical point of view, this means that SPAs allow one to model a system as the interaction of its subsystems. This principle resulted to be particularly useful because when a system is composed of interacting components we are now able to model and analyse interactions and components separately.

PEPA is a SPA designed for modelling computer and communication systems by J. Hillston as presented in [13]. This formalism can be used for studying quantitative and qualitative properties of the model under analysis. In particular, it provides an elegant syntax for expressing continuous-time Markov processes in a compositional way. PEPA peculiarity, with respect to other SPAs, consists in the fact that it associates a random variable, that will represent duration, to every action. These random variables are assumed to be exponentially distributed and this is the connection between this process algebra and Markov processes.

Syntax. PEPA is based on three main ingredients: *components* that are the active units within the system, *activities* that capture the actions of those units and *cooperation* that expresses the interaction between components.

Models are constructed from components which perform activities. Each activity has an *action type* α and an *activity rate* r. Each system action is uniquely typed and there is a countable set $\mathcal{A}$ of all possible types, activities with the same action type are different instances of the same action performed by the system. Since an exponential distribution is uniquely determined by its parameter, the duration of an activity is represented by a single real number parameter, the so-called *activity rate.* This rate may assume the value of any positive real number or the distinguished symbol $\top$ that has to be read as *unspecified.* The standard notation is as follows: $\mathcal{A}$ is the set of all action types (τ included, that denotes the type *unknown*), $\mathcal{R}^+$ is the set of all positive real numbers, including $\top$ and $\mathcal{A}ct = \mathcal{A} \times \mathcal{R}^+$ is the set of all activities. Therefore, an activity is denoted as

$$a = (\alpha, r),$$

with $a \in \mathcal{A}ct, \alpha \in \mathcal{A}$ and $r \in \mathcal{R}^+$. The combinators of the language allow one to describe the behaviour of each component via the activities they undertake and the interactions between them. Consequently, we are able to characterize the behaviour of the whole system. The combinator we are mostly interested in is *Cooperation* which is denoted as follows

$$C = C_1 \underset{L}{\bowtie} C_2.$$

$L \subseteq \mathcal{A}$ is the so-called *cooperation set* and defines the action types on which the components involved must synchronize. $\bowtie$ assumes that each component proceeds independently with any activities whose types do not occur in L. Whereas, activities with action types in L require the simultaneous involvement of both components in an activity of that type (notice that $\tau \notin L$). From a practical point of view, cooperation forms a new shared activity with the same action type as the two cooperating activities and with rate reflecting the rate of the *slowest* participant. If one of the two cooperating activities has an unspecified rate in a component, then the component is said to be passive with respect to that action type. These activities must be shared with another active component so that the other component can determine the rate of this shared activity. If more than one activity of a given passive type can be simultaneously enabled by a component, each unspecified activity rate must be assigned a weight; weights are natural numbers used to determine the relative probabilities of the possible outcomes of the activities of that action type. If no weights are assigned we assume that multiple instances have equal probabilities of occurring.

A.2 Reversed Compound Agent Theorem (RCAT)

RCAT is a theorem that gives sufficient conditions for a model to have the equilibrium distribution in product-form. For the sake of simplicity, consider a model consisting of only two interacting PEPA components on a set of action types L, as the one in Figure 9. For each action type in L, where in the particular case of Figure 9, we have $L = \{b\}$, one is always active in a component and always passive in the other one.

The joint model has a well-formed underlying Markov chain because all transitions have their rates. In principle, the stationary distribution can be obtained by solving the set of GBE of the joint process and by normalization. Product-form allows us to derive the stationary distribution of the joint model as normalized product of the distributions of the isolated model. However, the presence of passive types in the components requires some attention since their rates are unspecified.

Under some conditions, RCAT gives the way to specify the rates of passive types in the isolated components in such a way that the product-form solution holds. It is worth noticing that these rates are, in general, different from the synchronizing rates in the joint process.

Informally, RCAT identifies the reversed process of a cooperation in terms of the reversed process of its components. Using this result, we are also able to derive the steady-state probabilities of these interactive components replacing the occurrence of the passive action type transition with rates that can be algorithmically computed. This is possible thanks to the fact that instantaneous transition rates of the reversed process are related to those of the forward process through their stationary distribution, which is the same for both processes.

Upon the verification of the conditions given in Theorem 2, we are also ensured that a product-form solution exists and that it is the normalized product of the stationary distributions of the single components involved, in which the unknown rates of the synchronizing actions have been replaced by the reversed rates of the corresponding synchronizing transitions.

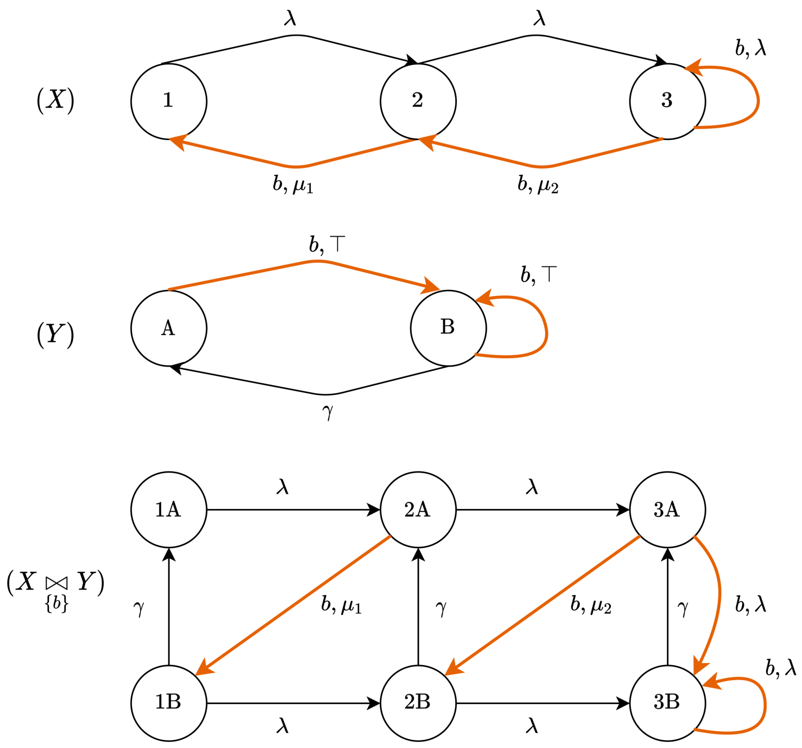

Fig. 9. Graphical representation of two isolated components (X) and (Y) and of their joint model $(X \underset{\{b\}}{\bowtie} Y)$.

More formally, RCAT exploits an abbreviated PEPA syntax. As we have seen in standard PEPA the shared actions of a cooperation occur at the rate of the slowest component; nevertheless, now, we require that for each action type in the cooperation set, exactly one agent is *passive* and concretely its synchronising action has rate $\top = \infty$. This, basically means that the passive agent actually waits for the other one. The set of actions, which an agent P may next engage in, is called the set of *current actions* and when the system is behaving as agent P these are the actions that are *enabled.* In addition, the derivation graph, formed by syntactic PEPA terms at the nodes, with arcs representing the transitions between them, determines the underlying Markov process of an agent P. The transition rate between two agents, C_i and C_j, denoted $q(C_i, C_j)$, is the sum of the action rates labelling arcs connecting C_i and C_j.

We also need to introduce relabelling, which preserves the semantic but will be useful to define the reversed process of cooperations: $P\{x \leftarrow y\}$. describes agent P in which all occurrences of symbol y have been replaced by x; y may be an action type or also a rate. Henceforth, we will refer to all agents as compound if they contain at least one instance of the cooperation combinator, and as sequential if they do not.

Rates of the Reversed Actions. It is rather simple to find a PEPA agent definition $\overline{C}$ that has the derivation graph with arrows in the opposite direction with respect to those of a given agent C. What it is not trivial is to find the appropriate rates of the reversed actions to make $\overline{C}$ the actual reversed process of C as defined by Kelly in [14].

For simple agents, it is possible to directly analyse the state transition graph of the Markov process and after determining the reversed graph $\overline{C}$ the procedure is quite straightforward and detailed in [10].

Notice that one of the reasons for analysing simple agents is to provide base cases for a compositional analysis of larger Markov chains. In fact, any continuous Markov chain can be described using only simple agents, however, the fact that an agent can perform multiple actions leading to the same derivative causes multiple arcs in the derivation graph and consequently also between two states in the transition graph of the underlying Markov chain. This does not create any issue as we can always determine the total reversed rate between any two states with multiple arcs between them. However, we need to consider multiple actions individually in cooperations.

In the reversed cooperation, the portion of the total reversed rate allocated to each individual reversed arc is crucial therefore a rule is needed. The rule we use is stated below.

Definition 1 (Reversed Action of Multiple Actions [10]). The reversed actions of multiple actions (a_i, λ_i) for $1 \leq i \leq n$ that an agent P can perform, which lead to the same derivative Q, are, respectively,

$$\left(\bar{a}_i, \left(\frac{\lambda_i}{\lambda}\right)\bar{\lambda}\right),$$

where $\lambda = \lambda_1 + \cdots + \lambda_n$ and $\bar{\lambda}$ is the reversed rate of the one-step, composite transition with rate λ in the Markov chain, corresponding to all the arcs between P and Q.

In other words, the total reversed rate is distributed amongst the reversed arcs in proportion to the forward transition rates.

Compound Agents. Under appropriate conditions, the reversed agent of a cooperation between two agents P and Q is a cooperation between the reversed agents of P and Q, after some reparametrisation. Before formally showing the result presented in Theorem 2, we first need to define some new notation.

The subset of actions types in a set L which are *passive* with respect to a process P (i.e., are of the form $(a, \top)$ in P) is denoted by $\mathcal{P}_P(L)$. The set of the corresponding active action types is denoted by $\mathcal{A}_P(L) = L \setminus \mathcal{P}_P(L)$.

Last thing to do before considering the following theorem we need to syntactically transform the agent under analysis so that every occurrence of a passive action $(a, \top)$ is relabelled as $(a, \top_a)$; this guarantees that every passive action rate is uniquely identified with exactly one action type.

Theorem 2 (Reversed Compound Agent Theorem [10]). *Suppose that the cooperation $P \underset{L}{\bowtie} Q$ has a derivation graph with an irreducible subgraph G. Given that*

(1) every passive action type in $\mathcal{P}_P(L)$ or $\mathcal{P}_Q(L)$ is always enabled in P or Q respectively (i.e., enabled in all states of the transition graph);
(2) every reversed action of an active action type in $\mathcal{A}_P(L)$ or $\mathcal{A}_Q(L)$ is always enabled in $\overline{P}$ or $\overline{Q}$, respectively;
(3) every occurrence of a reversed action of an active action type in $\mathcal{A}_P(L)$ (respectively, $\mathcal{A}_Q(L)$) has the same rate in $\overline{P}$ (respectively, $\overline{Q}$)

then the reversed agent $\overline{P \underset{L}{\bowtie} Q}$ with derivation graph containing the reversed subgraph $\overline{G}$, is

$$\overline{R}\{(\bar{a}, \overline{p_a}) \leftarrow (\bar{a}, \top) | a \in \mathcal{A}_P(L)\} \underset{L}{\bowtie} \overline{S}\{(\bar{a}, \overline{q_a}) \leftarrow (\bar{a}, \top) | a \in \mathcal{A}_Q(L)\},$$

where

$$R = P\{\top_a \leftarrow x_a | a \in \mathcal{P}_P(L)\},$$
$$S = Q\{\top_a \leftarrow x_a | a \in \mathcal{P}_Q(L)\},$$

$\{x_a\}$ are the solutions (for $\{\top_a\}$) of the equations

$$\top_a = \overline{q_a}, \quad a \in \mathcal{P}_P(L)$$
$$\top_a = \overline{p_a}, \quad a \in \mathcal{P}_Q(L)$$

and $\overline{p_a}$ (respectively, $\overline{q_a}$) is the symbolic rate of action type $\bar{a}$ in $\overline{P}$ (respectively $\overline{Q}$)

Notice that after the first definition of this theorem the third condition has been relaxed by A. Marin and M. G. Vigliotti in [18] to just require that the sum of the reversed rates of all the incoming transitions with the same active type must be the same in all states.

Consider again the simple model depicted in Figure 9. Component X is completely specified since it does not have any unknown rate, thus we can compute its marginal distribution π_X. This is:

$$\pi_X(1) = G\,, \pi_X(2) = G\frac{\lambda}{\mu_1}\,, \pi_X(3) = G\frac{\lambda^2}{\mu_1\mu_2}\,,$$

where G is a normalizing constant. The second step is to observe that every state has an incoming active type b whose reversed rate is, according to [14]:

— State 1: $\pi_X(2)/\pi_X(1)\mu_1 = \lambda$;
— State 2: $\pi_X(3)/\pi_X(2)\mu_2 = \lambda$;
— State 3: $\pi_X(3)/\pi_X(3)\lambda = \lambda$.

Notice that the reversed rate of action type a is constant and equal for all states. Therefore, $x_b = \lambda$ is the rate that we must use to study Y in isolation, that is, we replace all rates of types b with λ. Notice that this is *not* the synchronizing rate! Thus, the stationary distribution of Y is:

$$\pi_Y(A) = \frac{\gamma}{\gamma + \lambda}\,, \pi_Y(B) = \frac{\lambda}{\gamma + \lambda}$$

At this point, since b is enabled in each state of Y, we conclude that the stationary distribution of $X \underset{\{b\}}{\bowtie} Y$ is in product-form and equal to:

$$\pi_{X \underset{\{b\}}{\bowtie} Y}(n, m) = G\pi_X(n)\pi_Y(m)\,.$$

In this case, since the joint process state space is the Cartesian product of the state spaces of the single components, it is not necessary to renormalize the joint probabilities.

Propagation of Instantaneous Transitions. In [11], the authors have proved that RCAT can be applied also to models that involve propagating synchronizations. To understand what we mean we briefly need to discuss the concept of G-network. G-networks [6] are a class of product-form queueing networks in which both positive and negative customers are allowed; the first ones behave as we are used to in traditional queueing networks whether the negative ones at arrival to a station delete a positive customer, if any is present, or vanishes otherwise. In [7], it was shown that these negative customers may act as triggers, meaning that they can move a customer from a non-empty queue to another one. The class of G-networks has been then further investigated to comprehend chains of instantaneous state changes that can be modelled as the propagation of instantaneous transition as shown in [1, 9]. Accordingly, we write

$$P = (a \longrightarrow b, \top).Q$$

to denote a passive action with type a that takes process P to Q and instantaneously synchronizes as active on type b.

When studying the components in isolation, the rate at which the transition synchronizes on type b is the reversed rate of the passive action with type a that, in turns, is the reversed rate of the active transition with type a.

B RELEVANT GLOBAL BALANCE EQUATIONS

In this section, we show some relevant GBE characterizing the equilibrium distribution of the CTMC underlying the network represented in Figure 4. As far as this model is concerned, it is of particular interest to observe which are the probability fluxes characterizing state change transitions from a starting state describing a system including at least one station with an associated empty warehouse.

$$\begin{aligned}&\pi(1,1,0)(\mu_0+\mu_1+\mu_2) = \\ &= \pi(0,1,0)\mu_0 + \pi(2,0,0)\mu_1 + \pi(1,2,0)\mu_2(1-p) + \pi(1,0,1)\mu_3\end{aligned}$$

$$\begin{aligned}&\pi(1,0,1)\left(\mu_0+\mu_1+\mu_3+\mu_2(1-p)+\mu_2 p\frac{\mu_1}{\mu_1+\mu_3}\right) = \\ &= \pi(0,0,1)\mu_0 + \pi(1,1,0)\mu_2 p + \pi(1,1,1)\mu_2(1-p) + \pi(2,0,1)\mu_2(1-p)\frac{\mu_1}{\mu_2+\mu_3} \\ &+ \pi(2,0,0)\mu_2 p\frac{\mu_1}{\mu_1+\mu_3} + \pi(1,0,2)\mu_2(1-p)\frac{\mu_3}{\mu_1+\mu_3}\end{aligned}$$

$$\begin{aligned}&\pi(0,1,1)(\mu_0+\mu_1+\mu_2+\mu_3) = \\ &= \pi(0,0,1)\mu_1 + \pi(0,2,0)\mu_2 p + \pi(1,0,1)\mu_1 + \pi(0,0,2)\mu_3 + \pi(0,2,1)\mu_2(1-p)\end{aligned}$$

$$\begin{aligned}&\pi(1,0,0)\left(\mu_0+\mu_1+\mu_2(1-p)+\mu_2 p\frac{\mu_1}{\mu_1+\mu_3}\right) = \\ &= \pi(1,0,1)\mu_2(1-p)\frac{\mu_1}{\mu_1+\mu_3} + \pi(0,0,0)\mu_0 + \pi(2,0,0)\mu_2(1-p) + \pi(1,1,0)\mu_2(1-p)\end{aligned}$$

$$\begin{aligned}&\pi(0,1,0)(\mu_0+\mu_1+\mu_2) = \\ &= \pi(0,0,0)\mu_1 + \pi(1,0,0)\mu_1 + \pi(0,2,0)\mu_2(1-p) + \pi(0,0,1)\mu_3\end{aligned}$$

$$\begin{aligned}&\pi(0,0,0)\left(\mu_0+\mu_1+\mu_2\frac{\mu_1}{\mu_1+\mu_3}\right) = \\ &\pi(0,1,0)\mu_2(1-p) + \pi(1,0,0)\mu_2(1-p) + \pi(0,0,1)\mu_2(1-p)\frac{\mu_3}{\mu_1+\mu_3}\end{aligned}$$

$$\begin{aligned}&\pi(0,0,1)\left(\mu_0+\mu_1+\mu_3+\mu_2(1-p)\frac{\mu_3}{\mu_1+\mu_3}+\mu_2 p\frac{\mu_1}{\mu_1+\mu_3}\right) = \\ &= \pi(0,1,0)\mu_2 p + \pi(0,1,1)\mu_2(1-p) + \pi(1,0,1)\mu_2(1-p)\frac{\mu_1}{\mu_1+\mu_3} + \pi(1,0,0)\mu_2 p\frac{\mu_1}{\mu_1+\mu_3} \\ &+ \pi(0,0,2)\mu_2(1-p)\frac{\mu_3}{\mu_1+\mu_3} + \pi(0,0,0)\mu_2 p\frac{\mu_1}{\mu_1+\mu_3}\end{aligned}$$

REFERENCES

[1] S. Balsamo, P. G. Harrison, and A. Marin. 2010. A unifying approach to product-forms in networks with finite capacity constraints. *ACM SIGMETRICS Performance Evaluation Review* 38, 1 (2010), 25–36.

[2] J. P. Buzen. 1973. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM* 16, 9 (1973), 527–531.

[3] A. Clark, S. Gilmore, J. Hillston, and M. Tribastone. 2007. Stochastic process algebras. *The Computer Journal* 55, 7 (2007), 132–179. DOI: https://doi.org/10.1093/comjnl/bxr094

[4] N. Gast and B. Gaujal. 2010. A mean field model of work stealing in large-scale systems. In *Proceedings of the SIGMETRICS 2010, International Conference on Measurement and Modeling of Computer Systems*. Association for Computer Machinery, New York, NY, USA, 13–24.

[5] D. Gates and M. Westcott. 1994. Replacement of train wheels: An application of dynamic reversal of a Markov process. *Journal of Applied Probability* 31, 1 (1994), 1–8.

[6] E. Gelenbe. 1991. Product-form queueing networks with negative and positive customers. *Journal of Applied Probability* 28, 3 (1991), 656–663.

[7] E. Gelenbe. 1993. G-network with triggered customer movement. *Journal of Applied Probability* 30, 3 (1993), 742–748.

[8] M. Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge Press, USA.

[9] P. G. Harrison. 2004. Compositional reversed Markov processes, with applications to G-networks. *Performance Evaluation* 57, 3 (2004), 379–408.

[10] P. G. Harrison. 2003. Turning back time in Markovian process algebra. *Theoretical Computer Science* 290, 3 (2003), 1947–1986.

[11] P. G. Harrison and A. Marin. 2012. Deriving the rate equations characterising product-form models and application to propagating synchronisations. In *Proceedings of the 6th International ICST Conference on Performance Evaluation Methodologies and Tools* 6 (2012), 107–116.

[12] P. G. Harrison and A. Marin. 2014. Product-forms in multi-way synchronizations. *The Computer Journal* 57, 11 (2014), 1693–1710.

[13] J. Hillston. 2005. *A Compositional Approach to Performance Modelling*. Cambridge University Press, USA.

[14] F. P. Kelly. 1979. *Reversibility and Stochastic Networks*. John Wiley & Sons, Cambridge Mathematical Library.

[15] G. Kielanski and B. Van Houdt. 2021. Performance analysis of work stealing strategies in large scale multi-threaded computing. In *Proceedings of the fo Quantitative Evaluation of Systems QEST Lecture Notes in Computer Science, Vol. 12846*. Association for Computing Machinery, New York, NY, USA, 329–348.

[16] A. Marin. 2016. Product-form in G-Networks. *Probability in Engineering and Informational Sciences* 30, 3 (2016), 345–360.

[17] A. Marin, S. Balsamo, and J.-M. Fourneau. 2017. LB-networks: A model for dynamic load balancing in queueing networks. *Performance Evaluation* 115 (2017), 38–53. https://doi.org/10.1016/j.peva.2017.06.004

[18] A. Marin and M. G. Vigliotti. 2010. A general result for deriving product-form solutions in markovian models. In *Proceedings of the 1st joint WOSP/SIPEW International Conference on Performance Engineering*. ACM, New York, NY, USA, 165–176.

[19] Lena Mashayekhy, Nathan Fisher, and Daniel Grosu. 2016. Truthful mechanisms for competitive reward-based scheduling. *IEEE Transactions on computers* 65, 7 (2016), 2299–2312.

[20] Diletta Olliaro, Gianfranco Balbo, Andrea Marin, and Matteo Sereno. 2023. Skipping and fetching: Insights on non-conventional product-form solutions. In *Proceedings of the Quantitative Evaluation of Systems: 20th International Conference* . Springer-Verlag, Berlin, Heidelberg, 110–126.

[21] S. Otten, R. Krenzler, and H. Daduna. 2015. Models for integrated production-inventory systems: Steady state and cost analysis. *International Journal of Production Research* 54, 20 (2015), 6174–6191.

[22] B. G. Pittel. 1979. Closed exponential networks of queues with saturation: The Jackson-Type stationary distribution and its asymptotic analysis. *Mathematics of Operations Research* 4, 4 (1979), 357–378.

[23] M. Reiser and S. S. Lavenberg. 1980. Mean-value analysis of closed multichain queuing networks. *Journal of the ACM* 27, 2 (1980), 313–322.

[24] K. C. Sevcik and I. Mitrani. 1981. The distribution of queuing network states at input and output instants. *Journal of the ACM* 28, 2 (1981), 358–371.

[25] N. Sonenberg, G. Kielanski, and B. Van Houdt. 2021. Performance analysis of work stealing in large-scale multi-threaded computing. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 6, 2 (2021), 6:1–6:28.

[26] Z. Tan, H. Li, and X. He. 2021. Optimizing parcel sorting process of vertical sorting system in e-commerce warehouse. *Advanced Engineering Informatics* 48 (2021), 101279. https://doi.org/10.1016/j.ejor.2018.10.048

[27] E. Tappia, D. Roy, M. Melacini, and R. De Koster. 2018. Integrated storage-order picking systems: Technology, performance models, and design insights. *European Journal of Operational Research* 274, 3 (2018), 947–965.

[28] J. P. Van der Gaast, R. B. M. de Koster, I. J. B. F. Adan, and J. A. C. Resing. 2019. Capacity analysis of sequential zone picking systems. *Operations Research* 68, 1 (2019), 161–179.

[29] M. Westcott. 1990. Modelling and analysis of wheel replacement and restoration. In *Proceedings of the 1990 Mathematics-in-Industry Study Group*, N. G. Barton (Ed.). CSIRO Division of Mathematics and Statistics, Clayton, Australia, 30–40.

Received 9 February 2023; revised 26 November 2023; accepted 11 January 2024