Graphs, fluid paths and false fluid paths in exponential priority queues I. Construction of graphs MI-report 2003-12

Flora Spieksma[*]

September 19, 2003

Abstract

This paper is the first of two papers, where we study exponential priority queueing networks with deterministic routing. Our main aim is to formalise the phenomenon of 'repeated priority swapping', which seems to be responsible for the fact that the load of each service station being smaller than 1, does not always imply stability of the network.

In this paper we will associate a graph with the network and study a number of simple properties. In the second paper we will study networks with an acyclic associated graph. It will turn out that for such networks, the fluid equations have a unique solution, without cycles. Using results from Chen [3], it is then a rather straightforward consequence that stability is implied by the load of each service station being smaller than 1.

The idea for the construction of the relevant graph has been inspired by Dumas' paper [7].

[*]Postal address: Mathematical Institute, University of Leiden, P.O.Box 9512, 2300RA Leiden, the Netherlands; Email-address: spieksma@math.leidenuniv.nl

Contents

Introduction

It has long been a truth (in fact: a conjecture stated by Dobrushin), universally acknowledged, that the load being smaller than 1 for each service station should guarantee stability of a queueing network. In other words: local stability should imply overall stability. Unfortunately, this truth has not been acknowledged by priority queueing networks, as has been a well-known fact this past decade now. In [13], Rybko and Stolyar presented a counter-example consisting of two service stations, where an extra condition is needed for stability. Lu and Kumar ([11] introduced a related but deterministic model, exhibiting this same feature. The stochastic version of this model has been analysed by Dai and Weiss in [6].

A main method for investigating the stability region for such networks is the use of Lyapunov function criteria. The construction of a suitable Lyapunov function can be based on solution paths of the so-called fluid equations, one of which is the space-time scaled limit of the scaled network state: the fluid limit path (presumably it exists). The formulation of the fluid equations for priority queueing networks, has been initiated by Rybko and Stolyar in [13], and further elaborated by Dai [5] and Chen [3]. However, non-unique solutions may emerge, they do not necessarily have a probabilistic meaning ('false fluid paths'), and even cycles may occur. The problem is to distinguish how these observed properties depend on the network iself or on the set of equations posed.

In [12] (see also [9]) another approach is advocated in the more general context of face-homogeneous random walks on the orthant in $\mathbf{Z}^p$. In the queueing context it requires the study of sub-networks: a given set of queues is assumed saturated, thus inducing a stochastic process describing the evolution of the states of the other (non-saturated) queues. Provided the sub-network is 'stable', its fluid limit exists on a small time scale and is easily computable, see also [1], [8]. The speed along the fluid limit path, is called the 'second vector field' in the Malyshev and Menshikov terminology. The problem with this analysis is, that solving the stability problem for the whole network, is now reduced to solving stability problems for sub-networks. However, it does yield the true fluid limit path for the whole network. From a study of small examples it then emerges, that 'true' fluid limit paths have the same undesirable properties as fluid solutions: non-unicity and hence a *stochastic* fluid limit, as well as cycles. The examples by Rybko and Stolyar [13], and Lu and Kumar [11] typically have these features.

The aim of this paper is two-fold: to identify priority rules for which non-unicity and cycles may occur, and on the other hand, to identify priority rules for which local stability does not necessarily imply stability of the network. For the moment we restrict to deterministic routing.

The counter examples from the literature indicate that the latter phenomenon is generally known to require repeated priority swapping: that is, priority relations are not conserved while jobs travel through the network and may thus cause inefficient use of service capacity. My goal has been to formalise the 'repeated priority swapping' causing extra stability conditions to be required. It appears that one can associate a graph with the network, such that the occurrence of cycles in this graph corresponds to the 'repeated priority swapping' that we need. This graph is completely determined by the priority rules at hand and routes of the jobs through the network. Hence, this provides a tool for determining 'risky' priority rules.

Acyclicity of this graph can be shown to imply the existence of a unique fluid solution path, which is also acyclic for any initial state. Hence, the fluid limit path is deterministic. Using results from for instance Chen [3], sufficiency of local stability for stability is then a rather straightforward consequence.

Typically graphs associated with feed-forward networks [5] are acyclic and so the corresponding network is stable provided it is locally stable.

Although this graph arises in a natural way from the analysis of sub-networks, the idea for how the construct this graph is based on an interesting paper by Dumas [7], which to my knowledge has never been published.

I would like to point out, that although repeated priority swapping is one the obstacles for local stability to imply stability, there is another complicating feature in priority networks as opposed to Jackson networks (see [1] for a complete analysis of this model): sub-networks of Jackson networks are Jackson networks themselves, whereas this is not the case with prioirity networks. Indeed, when the set of non-saturated queues in a given service station is empty, then the saturated queue with highest priority (if present) will start to send jobs into the sub-network, till one of the higher priority non-saturated queues contains jobs again. On a more detailed scale, this implies that certain desirable monotonicity properties when comparing sub-networks that are important in the analysis in [1] for Jackson networks, do *not* hold for priority networks.

This work has been divided into two papers, mainly for reasons of size, but also for not being able to withstand the pressure of modern high-publication machine. In the first we will discuss the model, give an overview of Dumas' basic paper [7], and construct the relevant graph. We will derive some simple relations between the graph associated with a sub-network and solutions of the corresponding load equations. Solving the load equations is the first step in computing the local fluid limit.

In the companion paper, we will show that an acyclic graph implies the existence of a unique fluid solution, which is acyclic. Moreover, we plan to investigate the reverse problem, whether there exist service and/or arrval parameters such that the fluid limit path is cyclic, whenever the graph associated with the network is cyclic.

A third paper is considered, where we plan to study the relation between cycle lengths of the graph and complexity of the stability conditions. Additionally it is an open problem yet, whether exponential priority networks with *deterministic* routing allow unstable fluid solutions, whilst the network itself is stable. This might be a subject for a third paper as well. Bramson's example [2] requires a more complicated network structure.

1 Model and known results

1.1 Description of the model

We consider a network with the set of nodes $\mathcal{N} = \{1, \ldots, \mathfrak{n}\}$ and p job classes. The set $\mathcal{J}(n)$ denotes the job classes served at node n. This means that $\mathcal{J}(n)$, $i = 1, \ldots, \mathfrak{n}$, are a partition of the set of job classes. By $n(j)$ we will denote the node serving job class j, i.e. $n(j) = n$ iff $j \in \mathcal{J}(n)$.

Each node n consists of a preemptive single server and $\mathcal{J}(n)$ infinite capacity buffers, storing the jobs according to their classes. There is a fixed priority rule prescribing the class of the job to be served at any moment. By preemptiveness, whenever a higher priority job arrives at a node, service of a lower priority job immediately stops and the job is sent back to its queue.

Since we will assume the service times to be exponentially distributed, the preemptive and the preemptive-resume service discipline generate the same job process. Neither does the mechanism for selecting jobs of the same class from their queues play a role in our analysis. We only assume that the discipline is *work conserving.*

We may number the jobs such, that the job class with the highest number has highest priority in a node. Thus we can partially order the set of job classes by writing $j \prec k$ ($j \preceq k$) whenever j and k are served at the same node, and $j < k$ ($j \leq k$). That is, $j \prec k$ is equivalent to j having *strictly* lower priority than k and $j \preceq k$ to j having lower or equal priority to k.

The external arrival stream of class k jobs is a Poisson stream with rate λ_k ($\lambda_k = 0$ is allowed!). They require an exponentially distributed amount of service with parameter μ_k. Upon service completion a class k job becomes a class j job with probability r_{kj} and it leaves the network with probability $1 - \sum_j r_{kj}$. The matrix $R = (r_{ij})_{i,j=1,\ldots,p}$ is the routing matrix. We consider the case of the open network, that is, all jobs eventually leave the system. This amounts to assuming that $\sum_t R^t < \infty$. Finally we assume the arrival, service and routing processes to be independent of each other.

Though in this general setting the routing matrix is stochastic, we will restrict to the degenerate case of deterministic routing. Also, we assume for class k that if $r_{jk} = 1$ for some j, then $\lambda_k = 0$: each class only has one 'input channel'. Note that this model is not a re-entrant line but slightly more general.

Thus each job class i has a uniquely defined direct ancestor $a(i)$, which equals 0 whenever $\lambda_i > 0$. Similarly, it has a uniquely defined direct successor $s(i)$, set equal to 0 whenever $\sum_j r_{ij} = 0$, that is when job class i leaves the network with probability 1 upon completing its service.

The ancestor function is therefore an injective function $a : \{1, \ldots, p\} \to \{0, 1, \ldots, p\}$ and it is natural to require that the t-iterate $a^t \equiv 0$ for all large t, so that each class is originally input from outside the network. The successor function is a injective function $s : \{1, \ldots, p\} \to \{0, 1, \ldots, p\}$. By assumption on the routing matrix, the t-th iterate s^t equals 0 for all large t. For simplicity reasons related to certain homogeneity properties, we assume that there is no feedback with priorities. Formally this amounts to assuming that $s(i) \not\succ i$.

With each class i we can then associate a uniquely defined route $r(i, \sigma) \in \{1, \ldots, p\}$, $\sigma = 1, \ldots, \ell_i$, through the set of *classes*. The first stage $\sigma = 1$ corresponds to the 'input' class for i, i.e. $a\big(r(i, 1)\big) = 0$; ℓ_i is the *length* of the route, i.e. $s\big(r(i, \ell_i)\big) = 0$, and there is a unique stage σ_i with $r(i, \sigma_i) = i$. For later convenience, denote $r(i, 0) = 0$. The classes $r(i, \sigma)$, $\sigma < \sigma_i$ will be informally called the ancestors of i and the classes $r^(i, \sigma)$, $\sigma > \sigma_i$, its successors.

Note that all this is not at all a usual terminology. In general the route is taken to be through the set of *nodes*. For out purposes this 'classy' approach is the more convenient description.

The evolution of the number of jobs per class at the different nodes is a continuous time Markov chain $\mathcal{L}^c = \{\xi_t^c\}$, with i-th component equal the number of class i jobs present in the network at time t. The corresponding jump intensities are given by

$$\mu_{xy} = \begin{cases} \lambda_k, & y - x = e(k), \quad a(k) = 0 \\ \mu_k, & y - x = -e(k) + \mathbf{1}_{\{s(k)=j\}} e(j), \quad k \in \mathfrak{p}(x), \end{cases}$$

where $\mathfrak{p}(x) \subset \{1, \ldots, p\}$ is the set of highest priority job classes present in the different nodes in network state x.

Uniform boundedness of the jump intensities implies that the continuous time process has the same macro properties of our interest as a first order approximating Markov chain $\{\xi_t\}$ in discrete time, with transition probabilities $\mu_{xy} h$ for $y \neq x$, and $1 - \sum_{z \neq x} \mu_{xz} h$, $y = x$, for any sufficiently small number h.

Notice that by construction $\{\xi_t\}_t$ is aperiodic, but it need not be irreducible. However, the empty state is reachable from any other state, so that there is at most one closed class.

Without loss of generality we may rescale the jump intensities by multiplying by h and so we may assume that

$$\sum_{i: a(i)=0} \lambda_i + \sum_i \mu_i < 1.$$

The *discretised deterministic open priority network* is the aperiodic Markov chain $\xi_t = (\xi_{i,t})_{i=1,\ldots,p}$ on $\mathbf{Z}_+^p$ with transition probabilities

$$p_{xy} = \begin{cases} \lambda_k, & y - x = e(k), \quad a(k) = 0 \\ \mu_k, & y - x = -e(k) + \mathbf{1}_{\{s(k)=j\}} e(j), \quad k \in \mathfrak{p}(x) \\ 1 - \sum_{z \neq x} p_{xz}, & y = x. \end{cases}$$

Whenever the initial state x is given, we denote this by $\xi_t(x)$.

Finally, by *ergodicity* of a Markov chain we mean that the state space consists of one, aperiodic closed positive recurrent class. In this paper we will use the concept *stable* to denote the case of ergodic closed classes, such that absorption into the set of closed classes takes place with probability 1 and in finite expected time, for any initial state.

All assumptions made here are asumed to hold throughout the paper.

Non-empty queues and faces

Let $\mathbf{\Lambda} \subset \{1, \cdots, p\}$ be a set of non-empty queues. The 'face' $\mathcal{F}^{\mathbf{\Lambda}}$ in Malyshev and Menshikov's terminology ([12], [9]) is nothing but the set of all states with non-empty ($i \in \mathbf{\Lambda}$)-queues:

$$\mathcal{F}^{\mathbf{\Lambda}} = \{x \in \mathbf{R}_+^p \mid x_i > 0, i \in \mathbf{\Lambda}; x_i = 0, i \in \overline{\mathbf{\Lambda}}\},$$

where $\overline{\mathbf{\Lambda}} = \{1, \ldots, p\} \setminus \mathbf{\Lambda}$. Note that there are 2^p faces.

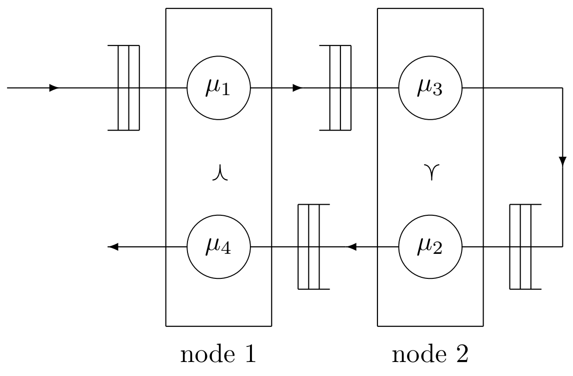
This is a convenient concept, since the jump probabilities are the same within each face. Hence ξ_t can be called a *face-homogeneous* Markov chain. The set function $\mathfrak{p}(x)$ is constant on faces and thus we can write $\mathfrak{p}(\mathbf{\Lambda})$ whenever $x \in \mathbf{\Lambda}$. Also the mean drift vectors from points in the same face are equal and we can therefore write

$$V(\mathbf{\Lambda}) = \sum_y (y - x)\, p_{xy},$$

for any given $x \in \mathcal{F}^{\mathbf{\Lambda}} \cap \mathbf{Z}_+^p$.

We will now introduce our toy example, which is the example intorduced by Lu & Kumar [11]. We will use for illustrating the concepts and statements that we will discuss in the sequel.

Toy example



This example is a priority network with 2 nodes and 4 job classes. Classes 1 and 4 are served at node 1; classes 2 and 3 are served at node 2.

The only exogenous arrivals are class 1 jobs: they arrive at node 1 at rate λ_1. Class 1's route through the network is given by

$$1 \to 3 \to 2 \to 4.$$

By definition $4 \succ 1$ and $3 \succ 2$. So, priorities are not conserved but swap: the first time jobs visit node 2, they have highest priority, whilst having lowest priority when they visit node 1 for the first time.

Remark that our numbering of the classes deviates from the usual one: this is to keep consistency with our numbering of classes according to priorities.

This model has a 4-dimensional state space, so there are 16 faces $\mathcal{F}^{\mathbf{\Lambda}}$ with $\mathbf{\Lambda}$ one of: the empty face $\emptyset$, $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{2,3\}$, $\{2,4\}$, $\{3,4\}$, $\{2,3,4\}$, $\{1,3,4\}$, $\{1,2,4\}$, $\{1,2,3\}$ and the interior of the state space $\{1,2,3,4\}$.

1.2 Mean conservation law and the load equations

In case ξ_t is stable with one ergodic closed class, it is straightforward to derive a conservation law defining the load ρ_i of job class i.

Let $N_i^+(T)$ be the number of class i jobs entering their queue in the time interval $(0,T]$ and $N_i^-(T)$ the number of class i jobs that finish service in the same time interval.

Denoting by ρ_i the stationary probability that class i is the highest priority job class present in its node (in words: the load of class i), we have

$$\lim_{T\to\infty} \frac{1}{T} N_i^+(T) = \left\{ \begin{array}{ll} \lambda_i, & a(i) = 0 \\ \rho_j \mu_j, & a(i) = j \end{array} \right. , \quad \text{and} \quad \lim_{T\to\infty} \frac{1}{T} N_i^-(T) = \rho_i \mu_i.$$

The number of class i jobs present in the system obeys the following conservation law

$$\xi_{i,T} - \xi_{i,0} = N_i^+(T) - N_i^-(T).$$

Dividing by T and taking the limit $T \to \infty$ gives

$$\lim_{T\to\infty} \frac{\xi_{i,T} - \xi_{i,0}}{T} = 0,$$

by stability. This implies flow conservation:

$$\lim_{T\to\infty} \frac{N_i^+(T) - N_i^-(T)}{T} = 0,$$

that is, the average number of class i arrival per unit time equals the average number of class i departures per unit time. As a consequence, we find that the loads are a solution to the following *load equations*

$$\rho_i \mu_i = \mathbf{1}_{\{a(i)=0\}} \lambda_i + \sum_{j \neq i} \mathbf{1}_{\{a(i)=j\}} \rho_j \mu_j. \tag{1.1}$$

These equations always have a *unique* solution, namely

$$\rho_i = \frac{1}{\mu_i} \lambda_{r(i,1)}, \tag{1.2}$$

even in the case of non-ergodicity.

When the network is stable, the load of each node n is necessarily smaller than 1:

$$\rho(n) = \sum_{i \in \mathcal{J}(n)} \rho_i < 1. \tag{1.3}$$

This is true because the load of a node is the stationary probability of the node to be non-empty and there is a positive probability of a node to be empty in case of stability. The solution of (1.1) is said to be *feasible* if (1.3) is satisfied. In the literature (1.3) is referred to as the *usual stability conditions.*

Unfortunately, as is well-known, feasibility is not a guarantee for stability, as the toy example shows (cf. continuation(i)). Thus the usual stability conditions are not sufficient to guarantee stability. However, the following property does hold (cf. [7]).

Lemma 1.1 *Let class i be such that $\sum_{j \succeq i} \rho_j < 1$ and define*

$$\tau_i = \inf\{t > 0 \,|\, \sum_{j \succeq i} \xi_{j,t} = 0\}.$$

Then $\mathsf{E}\{\tau_i \,|\, \xi_0 = x\} < \infty$ for any initial state x, that is, all queues $j \succeq i$ empty in finite expected time.

Proof. With class j jobs, associate the total amount $W_j(t,x)$ of class j service required by all class j jobs and ancestor classes present in the network at time t, given initial state x. This has the same distribution as the sum

$$\sum_{\sigma=1}^{\sigma_j} x_{r(j,\sigma)}$$

of exponentially distributed random variables with rate μ_j, because of the Markov property of the exponential distribution. Consider an $\mathsf{M/M/1}$-queue with input rate $\sum_{j \succeq i} \lambda_{r(j,1)}$ and mean service time

$$\sum_{j \succeq i} \frac{\lambda_{r(j,1)}}{\sum_{k \succeq i} \lambda_{r(k,1)}} \frac{1}{\mu_j}.$$

This system has load $\sum_{j \succeq i} \rho_j < 1$, and thus it is ergodic. Hence, for initial workload equal to $\sum_{j \succeq i} W_j(t,x)$ the queue empties in finite expected time. By using a suitable coupling it is easy to infer that the set of queues $j \succeq i$ empty at *least* as quick as the $\mathsf{M/M/1}$-queue given corresponding initial workloads. This is due to the fact that these queues may empty in spite of ancestor classes still being present elsewhere in the network. However, these ancestor class jobs do contribute to the workload $\sum_{j \succeq i} W_j(t,x)$ of the $\mathsf{M/M/1}$-queue. The conclusion follows. QED

The usual stability conditions can therefore be interpreted as a condition for *local stability*: they imply that each node is stable. Thus it makes sense to speak of local stability whenever the load equations have a feasible solution. We will do so throughout this paper. Our general problem is to study the relation between *local* and *global stability*, the latter meaning stability of the whole network.

Note that $\gamma_i = \rho_i \mu_i$ is called the throughput of class i. The load equations then reduce to the well-known traffic equations. The latter have no good meaning in the case of the sub-networks or projected Markov chains that we will study in the next section. So we prefer to work directly with the load equations.

Toy example: continuation(i)
The computation of the loads is simple: $\rho_i = \lambda_1/\mu_i$, $i = 1, \ldots, 4$. The network is locally stable whenever

$$\rho_1 + \rho_4 < 1, \quad \text{and} \quad \rho_2 + \rho_3 < 1. \tag{1.4}$$

As is well-known [6], for *global* stability the extra condition

$$\rho_3 + \rho_4 < 1 \tag{1.5}$$

is necessary.

1.3 Essentiality

In general, the stochastic process associated with a priority queueing network need not be irreducible: although there is always at most one closed class containing the empty state 0, there can be inessential states. A priori, it is not clear whether the process starting in inessential states can exhibit 'worse' behaviour that when it starts in the closed class.

Dumas' paper [7] studies the occurrence of inessential states and its connections with the problem of non-sufficiency of local stability for global stability. With the network he associates a graph with vertices the set of classes. Cycles in this graph are shown only to occur if and only if there are inessential states. An extension of this graph plays the crucial role in this paper. Because of its importance for this work and because its not having been published, I will describe the graph of Dumas' paper and recall some of his results.

In [7], Dumas assumes that there is no *feedback with priorities*, i.e. $s(i) \not\prec i$ for any job class i. Under this assumption, it holds that the states in a given face $\mathcal{F}^{\mathbf{\Lambda}}$ are either *all* essential or *all* in-essential. Further, if $\mathcal{F}^{\mathbf{\Lambda}}$ is in-essential then so is $\mathcal{F}^{\mathbf{\Lambda}'}$ for any $\mathbf{\Lambda}' \supset \mathbf{\Lambda}$ (cf. Lemma 3.2 [7]). If there is feedback with priorities then one easily sees that all states with $x_{s(i)} > 1$ are inessential, whereas states with $x_{s(i)} = 1$ may be essential.

The following result shows that starting in an inessential state cannot destroy stability.

Theorem 1.1 ([7] Proposition 2.5) *Suppose that there is no feedback with priorities, and that $\rho(n) \leq 1$, for any node n. The closed class of essential states is reached in finite expected time from any inessential state.*

Next, a face $\mathcal{F}^{\mathbf{\Lambda}}$ is called a *minimal inessential face*, whenever $\mathcal{F}^{\mathbf{\Lambda}}$ is in-essential, and $\mathcal{F}^{\mathbf{\Lambda}'}$ is essential for any $\mathbf{\Lambda}' \subset \mathbf{\Lambda}$, $\mathbf{\Lambda}' \neq \mathbf{\Lambda}$. The face $\mathcal{F}^{\mathbf{\Lambda}}$ is called *simple* whenever it contains at most one job class from each node, i.e. $\mathbf{\Lambda} \cap \mathcal{J}(n) \leq 1$ for any node n.

The graph $\mathcal{G}^e$ and (in)essentiality
The graph has the set of vertices $V = \{1, \ldots, p\}$ corresponding to the job classes. We draw a (directed) arrow $i \to j$, whenever $a(i) \prec j$.

The following result holds.

Theorem 1.2 ([7] Proposition 3.9) *There are inessential states if and only if the graph $\mathcal{G}^e$ has cycles. The set of classes of a minimal cycle is a simple face without in-transitions.*

This analysis is used to derive the following assertion.

Theorem 1.3 ([7] Proposition 4.2) *A necessary condition for the network to be stable is that for any face $\mathcal{F}^{\mathbf{\Lambda}}$*

$$\sum_{i \in \mathbf{\Lambda}} \rho_i < \max\{|\mathbf{\Lambda}'| : \mathbf{\Lambda}' \subset \mathbf{\Lambda}, \mathcal{F}^{\mathbf{\Lambda}'} \textit{ is essential and simple}\}. \tag{1.6}$$

This provides nice *linear* conditions on the loads on top of local stability that are straightforward to compute. Unfortunately many examples from the literature require non-linear conditions (cf. [4]). Moreover, in the absence of inessential states, condition (1.6) is even weaker than requiring local stability. This, in spite of the fact that restrictive conditions on top of local stability may be required indeed for achieving for achieving global stability. This will be motivated below in continuation(ii) of the toy example.

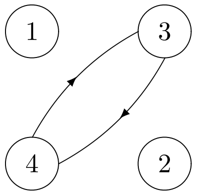
Toy example: continuation(ii)
The graph $\mathcal{G}^e$ associated with this model is depicted below. The cycle $3 \to 4 \to 3$ is a minimal cycle. Theorem 1.2 hence implies that the face $\mathcal{F}^{\{3,4\}}$ must be a face without in-transitions. This is clearly true: as soon as there is at least one class 3 job present, service of class 2 stops and so there are no class 4 arrivals. Thus class 4 arrivals can only occur as long as no class 3 jobs are present in the network. In turn, presence of class 4 jobs bars arrival of class 3 jobs (cf. also continuation(iii)).

The extra condition (1.5) is clear now. Suppose that $\rho_3 + \rho_4 > 1$. These are the stationary probabilities that class 3 and class 4 are present and being served. As a consequence the event that class 3 and class 4 are simultaneously present has positive (stationary) probability mass, which is at least equal to the 'overlap' $\rho_3 + \rho_4 - 1$ of ρ_3 and ρ_4. But we have just argued that this event cannot at all occur in equilibrium.

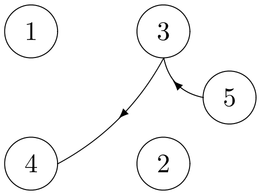
We will comment on this. As a consequence of Dumas' analysis, inessential states only can occur when there is some kind of priority swapping phenomenon (influencing the stability conditions). The reverse is certainly not true. An example of this is simply the following: add an extra node after jobs leave node 2 for the second time and before they re-enter node 1; take routing structure

$$\underbrace{1}_{\text{node 1}} \to \underbrace{3 \to 2}_{\text{node 2}} \to \underbrace{5}_{\text{node 3}} \to \underbrace{4}_{\text{node 1}} .$$

The corresponding graph $\mathcal{G}^e$ has no cycles and hence the state space is essential. However, we have the same priority swapping and a similar extra condition will be necessary for global stability, whenever service at the extra node is fast. This implies that inessentiality is not responsible for the need of extra conditions on top of local stability. Curiously enough at first sight, putting a *slow server* in the extra node has a stabilising effect.



Graph $\mathcal{G}^e$
of Lu & Kumar's example



Graph $\mathcal{G}^e$
of extended example

Priority swapping can be formalised through cycles in the graph with arrows $i \to j$ (or vice versa) when *some* ancestor of i has lower priority than j. This will be the topic of the next section.

2 Sub-networks

As mentioned before, the second-vector field setup is a local study space-time scaling limits, by means of sub-networks. These sub-networks are almost priority networks, but not quite. This seems to be one of the responsible agents for disrupting phenomena, which already emerge when setting up the load equations.

2.1 Sub-networks or induced chains and the Λ-load equations

We need to introduce a number of concepts. Interpret the set $\mathbf{\Lambda}$ as a set of saturated queues. The *induced chain* $\xi_t^{\mathbf{\Lambda}}$ is the Markov chain associated with the sub-network with classes $j \in \overline{\mathbf{\Lambda}}$, given that the classes $i \in \mathbf{\Lambda}$ are saturated, that is, they never empty. Formally, $\xi_t^{\mathbf{\Lambda}}$ is the orthogonally projected Markov chain ξ_t unto the state space

$$C^{\mathbf{\Lambda}} = \{y \in \mathbf{Z}_+^p \,|\, (x_0, y - x_0) = 0\},$$

where $x_0 \in \mathcal{F}^{\mathbf{\Lambda}} \cap \mathbf{Z}_+^p$ is fixed but arbitrary, i.e. it has transition probabilities

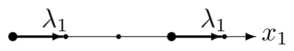
$$p^{\mathbf{\Lambda}}_{xy} = \sum_{\substack{z \in \mathbf{Z}_+^p: \\ y-z \in \mathcal{F}^{\mathbf{\Lambda}}}} p_{x+x_0,z}, \qquad x+x_0, y \in C^{\mathbf{\Lambda}}.$$

The induced chain $\xi_t^{\mathbf{\Lambda}}$ is *almost* a priority network on the set of classes $i \in \overline{\mathbf{\Lambda}}$. Indeed, it has additional input whenever node n is empty in the subnetwork (i.e. no class $i \in \overline{\Lambda} \cap \mathcal{J}(n)$ jobs are present) and $\mathfrak{p}(\mathbf{\Lambda}) \cap \mathcal{J}(n) \neq \emptyset$.

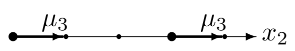
For clarity reasons we will often speak of the induced chain $\xi_t^{\mathbf{\Lambda}}$ as the sub-network on the $\overline{\mathbf{\Lambda}}$-classes or queues or the sub-network with saturated $\mathbf{\Lambda}$-queues.

Toy example: continuation(iii)
We will describe all possible sub-networks. First, the sub-network corresponding to $\mathbf{\Lambda} = \emptyset$ is the original priority network. When $\mathbf{\Lambda} = \{1,2,3,4\}$, the sub-network has a state space consisting of 1 point only, which is stable by definition.

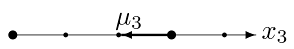
In the pictures representing the transition structure, we will only depict transitions leading out of a state, one per face.



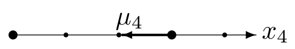
$\mathbf{\Lambda} = \{2,3,4\}$



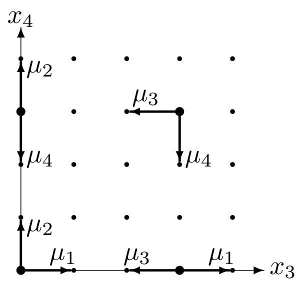
$\mathbf{\Lambda} = \{1,3,4\}$



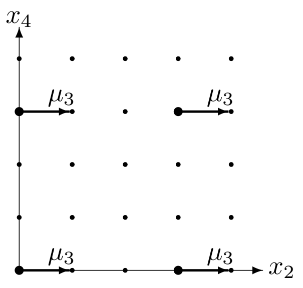
$\mathbf{\Lambda} = \{1,2,4\}$



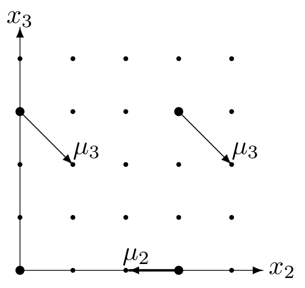
$\mathbf{\Lambda} = \{1,2,3\}$



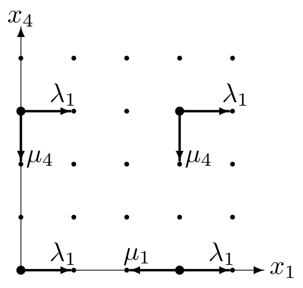
$\mathbf{\Lambda} = \{1,2\}$



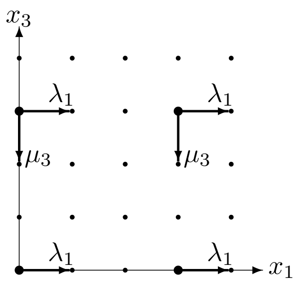
$\mathbf{\Lambda} = \{1,3\}$



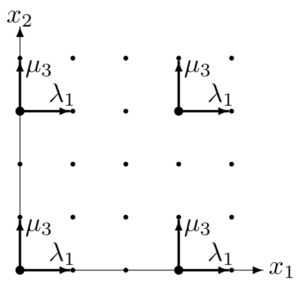
$\mathbf{\Lambda} = \{1,4\}$



$\mathbf{\Lambda} = \{2,3\}$



$\mathbf{\Lambda} = \{2,4\}$



$\mathbf{\Lambda} = \{3,4\}$

When $|\mathbf{\Lambda}| = 1$, the corresponding sub-networks are three-dimensional Markov chains, that in general cannot be represented in a picture. For $\mathbf{\Lambda} = \{3\}$, class 1 and 4 queues behave the same as in the sub-network with saturated $\mathbf{\Lambda}' = \{2,3\}$ queues; the class 2 queue blows up at rate μ_3, and has no output. For $\mathbf{\Lambda} = \{4\}$, classes 2 and 3 queues behave the same as in the sub-network with saturated $\mathbf{\Lambda}' = \{1,4\}$ queues. Additionally, class 1 queue blows up at rate λ_1. Faces $\{1\}$ and $\{2\}$ are more complicated to describe, so we will not do that.

The pictures evidence that projecting the jumps in the interior unto the boundary gives a different transition structure from the one on the boundary: this is because of additional input occurring.

2.2 Essentiality

As to the class-structure (here we mean the concept 'class' used in Markov chain theory) of sub-networks, this can also be quite different from the class-structure of the whole priority network as evidenced in the above example. For instance, sub-networks may have a countably infinite number of closed classes: the subnetwork with saturated queues $\{2,4\}$ of the toy example (cf. continuation(iii)) has infinitely many inessential classes. The case of infinitely many essential classes may occur as well, only not in this example. We will have to study this in order to understand the meaning of solutions to load equations for sub-networks and of scaled limits. It seems that *monotonicity* of a face will be a useful tool. We introduce some concepts.

Consider a class $i \in \overline{\mathbf{\Lambda}}$. Its $\mathbf{\Lambda}$-*route* $r^{\mathbf{\Lambda}}(i,\sigma)$, $\sigma = 1,\ldots,\ell_i^{\mathbf{\Lambda}}$, through the set of classes is defined by: $a\big(r^{\mathbf{\Lambda}}(i,1)\big)$, $s\big(r^{\mathbf{\Lambda}}(i,\ell_i^{\mathbf{\Lambda}})\big) \in \{0\}\cup\mathbf{\Lambda}$; $r^{\mathbf{\Lambda}}(i,\sigma) \in \overline{\mathbf{\Lambda}}$, $s = 1,\ldots,\ell_i^{\mathbf{\Lambda}}$. So the $\mathbf{\Lambda}$-route of class i is the part of class i's whole route r containing i itself (strictly) between two successive visits to the $\mathbf{\Lambda}$-queues or the outside world. As before, denote by $\sigma_i^{\mathbf{\Lambda}}$ the unique stage that the $\mathbf{\Lambda}$-route of i equals i: $r^{\mathbf{\Lambda}}(i,\sigma_i^{\mathbf{\Lambda}}) = i$. Then a $\mathbf{\Lambda}$-ancestor (-successor) of i s is simply any class $j = r^{\mathbf{\Lambda}}(i,\sigma)$, $1 \le \sigma < \sigma_i^{\mathbf{\Lambda}}$ ($\sigma_i^{\mathbf{\Lambda}} < \sigma \le \ell_i^{\mathbf{\Lambda}}$). For simplicity of exposition, define $r^{\mathbf{\Lambda}}(i,0) = a\big(r^{\mathbf{\Lambda}}(i,1)\big)$. We do not assume this to be part of the class i $\mathbf{\Lambda}$-route, but we will call it the $\mathbf{\Lambda}$-source of class i. Similarly, denote $\sigma_i^{\mathbf{\Lambda}} = 0$ for $i \in \mathfrak{p}(\mathbf{\Lambda})$.

It is sufficient to restrict to studying 'monotonic' faces defined below.

Definition 2.1 i) The set $\mathbf{\Lambda}$ of saturated queues is called monotonic (and the face $\mathcal{F}^{\mathbf{\Lambda}}$ monotonic), if $i \in \mathbf{\Lambda}$ implies $j \in \mathbf{\Lambda}$ for all $j \prec i$. The monotonic closure $\mathbf{\Lambda}(m)$ of $\mathbf{\Lambda}$ is the smallest monotonic set of saturated queues containing $\mathbf{\Lambda}$.

ii) For a monotonic set $\mathbf{\Lambda}$, we say that class $i \in \overline{\mathbf{\Lambda}}$ has a passive $\mathbf{\Lambda}$-source, whenever $r^{\mathbf{\Lambda}}(i,0) \in \mathbf{\Lambda}\setminus\mathfrak{p}(\mathbf{\Lambda})$. Whenever $r^{\mathbf{\Lambda}}(i,0) \in \{0,\mathfrak{p}(\mathbf{\Lambda})\}$, we say that class $i \in \overline{\mathbf{\Lambda}}$ has an active $\mathbf{\Lambda}$-source.

Note that for a non-monotonic face, the fact that class i has an active $\mathbf{\Lambda}$-source, does not necessarily imply that there will ever be any class i jobs in the system. Monotonicity *does* guarantee this.

Furthermore, studying monotonic sets $\mathbf{\Lambda}$ is sufficient for understanding the stability structure of sub-networks with non-monotonic sets of saturated queues. Let us explain this.

Suppose $\mathbf{\Lambda}$ is non-monotonic. Let class i be given. First we suppose that $r^{\mathbf{\Lambda}}(i,\sigma) \prec \mathfrak{p}(\mathbf{\Lambda})\cap\mathcal{J}\big(n\big(r^{\mathbf{\Lambda}}(i,\sigma)\big)\big)$ for some $0 \le \sigma < \sigma_i^{\mathbf{\Lambda}}$.

Then there will not be any class i arrivals to this subnetwork. If class $i \in \overline{\mathbf{\Lambda}(m)}$, then queue i will be served. The time spent on it by the server till it empties has a finite expectation and queue i will remain empty forever. As a consequence, class i jobs do not influence the stability structure of the sub-network nor the amount of closed classes. The same holds for monotonic $\mathbf{\Lambda}$ and any class i with a passive $\mathbf{\Lambda}$-source.

On the other hand, if class $i \in \mathbf{\Lambda}(m)\setminus\mathfrak{p}(\mathbf{\Lambda})$, then it is never served. Hence, the state of queue i does not alter. There are a countably infinite number of closed classes. However, the behaviour of the other queues is independent of class i jobs and so the stability structure of the other queues can be analysed separately (from class i).

Finally, suppose $r^{\mathbf{\Lambda}}(i,\sigma) \succeq \mathfrak{p}(\mathbf{\Lambda})\cap\mathcal{J}\big(n\big(r^{\mathbf{\Lambda}}(i,\sigma)\big)\big)$, $\sigma < \sigma_i^{\mathbf{\Lambda}}$. If $i \in \mathbf{\Lambda}(m)\setminus\mathfrak{p}(\mathbf{\Lambda})$, then there are class i arrivals but no services: queue i is transient and there are a countable infinite number of in-essential closed classes. This class makes the sub-network transient if the sub-network with saturated $\mathbf{\Lambda}(m)$-queues were not already so. However, the behaviour of the other queues is again independent of the class i queue and they can be analysed separately.

As a consequence, we may restrict to monotonic sets. This reduces the amount of sub-networks to be investigated considerably, as can be seen from continuation (iv) of Lu & Kumar's example.

From now on, we will denote by superscript $\mathbf{\Lambda}$ all corresponding quantities to the induced chain $\xi_t^{\mathbf{\Lambda}}$.

Lu & Kumar's example: continuation (iv)
The monotonic faces are $\mathcal{F}^{\mathbf{\Lambda}}$ with $\mathbf{\Lambda}$ one of: $\emptyset$, $\{1\}$, $\{2\}$, $\{1,2\}$, $\{1,4\}$, $\{2,3\}$, $\{1,2,3\}$, $\{1,2,4\}$ and $\{1,2,3,4\}$.

The set of saturated queues equal $\mathbf{\Lambda} = \{4\}$ is not monotonic. Since class 1 has non-zero $\{4\}$-input, the corresponding queue explodes.

The set $\mathbf{\Lambda} = \{1,2,3\}$ is monotonic, but class 4 has passive $\{1,2,3\}$-source 2: in the long run it is empty.

2.3 Mean conservation law and the $\mathbf{\Lambda}$-load equations

Suppose that $\mathbf{\Lambda}$ defines a sub-network with one ergodic closed class. Similarly to § 1.2 one can derive a system of linear equations for the stationary probabilities $\rho_i^{\mathbf{\Lambda}}$, that class i is the highest present in its node for the sub-network, $i \in \overline{\mathbf{\Lambda}} \cup \mathfrak{p}(\mathbf{\Lambda})$.

This set of equations may have a feasible solution for general $\mathbf{\Lambda}$. However, again this need not imply stability of the sub-network. Our main goal is get hold on the problem of how to infer stability of a sub-network from information on solution(s) to these equations. The generalised version of the graph introduced by Mr. Dumas that we will describe later-on in this subsection, will provide the main tool for this. It emanates naturally from analysing the $\mathbf{\Lambda}$-load equations.

In view of the previous subsection, it is sufficient to restrict our study to *monotonic* sets of saturated queues. Indeed, if $\mathbf{\Lambda}$ is not monotonic, then for the classes $i \in \overline{\mathbf{\Lambda}}$ that make it non-monotonic ($i \in \mathbf{\Lambda}(m)$), we can extend the solution to the $\mathbf{\Lambda}(m)$-load equations (to be derived below) by putting $\rho_i^{\mathbf{\Lambda}} = 0$. Similarly, we can view $\rho^{\mathbf{\Lambda}}$ as a vector with p components, by setting $\rho_i^{\mathbf{\Lambda}} = 0$, $i \in \mathbf{\Lambda} \setminus \mathfrak{p}(\mathbf{\Lambda})$.

Monotonic $\mathbf{\Lambda}$
So, let a monotonic $\mathbf{\Lambda}$ be given and assume that the corresponding sub-network (induced chain) $\xi_t^{\mathbf{\Lambda}}$ has one essential closed class that is ergodic. Recall that the sub-network is almost a priority network, with additional input from $\mathfrak{p}(\mathbf{\Lambda})$-queues whenever higher priority jobs are absent.

Denote by $\rho_i^{\mathbf{\Lambda}}$ the stationary probability that class $i \in \overline{\mathbf{\Lambda}} \cup \mathfrak{p}(\mathbf{\Lambda})$ is the highest priority class present in its node. Note for $i \in \mathfrak{p}(\mathbf{\Lambda})$, that $\rho_i^{\mathbf{\Lambda}}$ is the stationary probability that node $n(i)$ is *empty* in the sub-network.

A similar derivation as in §1.2 yields that the $\rho_i^{\mathbf{\Lambda}}$ are a solution of

$$\rho_i^{\mathbf{\Lambda}} \mu_i = \mathbf{1}_{\{a(i)=0\}} \lambda_i + \sum_j \mathbf{1}_{\{a(i)=j:\ j \in \overline{\mathbf{\Lambda}} \cup \mathfrak{p}(\mathbf{\Lambda})\}} \rho_j^{\mathbf{\Lambda}} \mu_j, \quad i \in \overline{\mathbf{\Lambda}}. \tag{2.1}$$

This equation has a $|\mathfrak{p}(\mathbf{\Lambda})|$-dimensional solution space. For $i \in \overline{\mathbf{\Lambda}}$ the solution can be expressed by (cf. (1.2))

$$\rho_i^{\mathbf{\Lambda}} = \begin{cases} \frac{\lambda_{r(i,1)}}{\mu_i} = \rho_i, & r^{\mathbf{\Lambda}}(i,0) = a\big(r(i,1)\big) = 0 \\ 0, & r^{\mathbf{\Lambda}}(i,0) \in \mathbf{\Lambda} \setminus \mathfrak{p}(\mathbf{\Lambda}) \\ \frac{\mu_{r^{\mathbf{\Lambda}}(i,0)}}{\mu_i} \rho^{\mathbf{\Lambda}}_{r^{\mathbf{\Lambda}}(i,0)}, & r^{\mathbf{\Lambda}}(i,0) \in \mathfrak{p}(\mathbf{\Lambda}), \end{cases} \tag{2.2}$$

where $\rho_j^{\mathbf{\Lambda}}$, $j \in \mathfrak{p}(\mathbf{\Lambda})$, are still unknowns. Since these quantities have the interpretation of being the stationary probability that the corresponding node is empty in the sub-network, we have to require in addition that

$$\rho_j^{\mathbf{\Lambda}} + \rho^{\mathbf{\Lambda}}(n(j)) = 1, \quad \text{for any} \quad j \in \mathfrak{p}(\mathbf{\Lambda}), \tag{2.3}$$

with

$$\rho^{\mathbf{\Lambda}}(n) = \sum_{i \in \overline{\mathbf{\Lambda}} \cap \mathcal{J}(n)} \rho_i^{\mathbf{\Lambda}}$$

the $\mathbf{\Lambda}$-load of node n. Because of monotonicity

$$\sum_{i \in \overline{\mathbf{\Lambda}} \cap \mathcal{J}(n)} \rho_i^{\mathbf{\Lambda}} = \sum_{i \succ j} \rho_i^{\mathbf{\Lambda}}.$$

We will call (2.1) and (2.3) the $\mathbf{\Lambda}$-*load equations*. It is preferable to represent solutions $\rho^{\mathbf{\Lambda}}$ to $\mathbf{\Lambda}$-load equations by vectors of dimension p, by putting $\rho_i^{\mathbf{\Lambda}} = 0$ whenever $i \in \mathbf{\Lambda} \setminus \mathfrak{p}(\mathbf{\Lambda})$.

A solution $\rho_i^{\mathbf{\Lambda}}$, $i \in \overline{\mathbf{\Lambda}} \cup \mathfrak{p}(\mathbf{\Lambda})$, $\rho_i^{\mathbf{\Lambda}} = 0$, $i \in \mathbf{\Lambda} \setminus \mathfrak{p}(\mathbf{\Lambda})$, to the $\mathbf{\Lambda}$-load equations, is said to be *feasible*, if it is non-negative and the $\mathbf{\Lambda}$-load of each node is smaller than 1, i.e.

$$\rho^{\mathbf{\Lambda}}(n) < 1, \quad \text{for any node } n. \tag{2.4}$$

Note that the latter is an *additional* condition for nodes for which $\overline{\mathbf{\Lambda}}$ contains all classes served at the node.

It is simple to prove that (2.1) and (2.3) have a unique solution (possibly non-feasible) Lebesgue-almost everywhere on the set of parameters λ_i, $a(i) = 0$, and $\mu_i > 0$, $i = 1, \ldots, p$, also in the case of $\mathbf{\Lambda}$ with $\xi_t^{\mathbf{\Lambda}}$ not stable. Nevertheless, it appears useful to study solutions of these equations and the priority structures for which non-uniqueness may occur. We will do so by means of the following recursive algorithm for solving these.

Reduction algorithm for monotonic $\mathbf{\Lambda}$

Step 1. Put $\mathbf{\Lambda}_0 = \mathbf{\Lambda} \setminus \mathfrak{p}(\mathbf{\Lambda})$, $\mathbf{\Lambda}_1 = \emptyset$ and $\mathbf{\Lambda}_2 = \mathfrak{p}(\mathbf{\Lambda})$.

Step 2. Initialisation i) For $i \in \mathbf{\Lambda}_0$, put $\rho_i^{\mathbf{\Lambda}} = 0$. For all $i \in \overline{\mathbf{\Lambda}}$ do the following. If $r^{\mathbf{\Lambda}}(i,0) \in \mathbf{\Lambda}_0$, put $\rho_i^{\mathbf{\Lambda}} = 0$. If $r^{\mathbf{\Lambda}}(i,0) = 0$ put

$$\rho_i^{\mathbf{\Lambda}} = \frac{\lambda_{r(i,1)}}{\mu_i} = \rho_i.$$

Set $\mathbf{\Lambda}_0 := \mathbf{\Lambda}_0 \cup \{i\}$.

ii) For any $i \in \overline{\mathbf{\Lambda}_0 \cup \mathbf{\Lambda}_2}$ check whether $r^{\mathbf{\Lambda}}(i,0) \in \mathfrak{p}(\mathbf{\Lambda}) \cap \mathcal{J}(n(i))$. In words, this amounts to checking whether class i $\mathbf{\Lambda}$-input belongs to the same node as i. Clearly there may be more than one such class i for each node. Then put $\mathbf{\Lambda}_1 := \mathbf{\Lambda}_1 \cup \{i\}$.

Step 3. Choose any $i \in \mathbf{\Lambda}_2$. If $\mathcal{J}\big(n(i)\big) \setminus \{i\} \subset \mathbf{\Lambda}_0 \cup \mathbf{\Lambda}_1$, then $\rho_i^{\mathbf{\Lambda}}$ as well as $\rho_k^{\mathbf{\Lambda}}$ are now explicitly computable for all classes k for which i is a $\mathbf{\Lambda}$-input class, in the following way.

i) If $\mathbf{\Lambda}_1 \cap \mathcal{J}\big(n(i)\big) = \emptyset$, set $\rho_i^{\mathbf{\Lambda}} = 1 - \sum_{j \succ i} \rho_j^{\mathbf{\Lambda}}$.
If $\mathbf{\Lambda}_1 \cap \mathcal{J}\big(n(i)\big) \neq \emptyset$, set

$$\rho_i^{\mathbf{\Lambda}} = \frac{1 - \sum_{j \succ i, j \in \mathbf{\Lambda}_0} \rho_j^{\mathbf{\Lambda}}}{1 + \sum_{j \succ i, j \in \mathbf{\Lambda}_1} \mu_i/\mu_j};$$

ii) for any $k = r^{\mathbf{\Lambda}}(i,\sigma)$ with $1 \leq \sigma \leq \ell_{s(i)}^{\mathbf{\Lambda}}$, set $\rho_k^{\mathbf{\Lambda}} = \rho_i^{\mathbf{\Lambda}} \mu_i/\mu_k$; put $\mathbf{\Lambda}_0 := \mathbf{\Lambda}_0 \cup \{k\}$ and if $n(k) = n(i)$ put $\mathbf{\Lambda}_1 := \mathbf{\Lambda}_1 \setminus \{k\}$;

iii) put $\mathbf{\Lambda}_2 = \mathbf{\Lambda}_2 \setminus \{i\}$; $\mathbf{\Lambda}_0 = \mathbf{\Lambda}_0 \cup \{i\}$.

Repeat Step 3 till there are no such i to be found anymore. Then **stop**.

Remark 2.1 Any unassigned job class i, i.e. any $i \in \overline{\mathbf{\Lambda}_0 \cup \mathbf{\Lambda}_1 \cup \mathbf{\Lambda}_2}$, has the two following properties: $r^{\mathbf{\Lambda}}(i,0) \in \mathbf{\Lambda}_2$ and $r^{\mathbf{\Lambda}}(i,0) \notin \mathcal{J}\big(n(i)\big)$.

Remark 2.2 Suppose $\mathbf{\Lambda}_2 \neq \emptyset$. By the previous remark, and (2.2), for any unassigned job class i, $\rho_i^{\mathbf{\Lambda}} = \rho_{r^{\mathbf{\Lambda}}(i,0)}^{\mathbf{\Lambda}} \mu_{\rho^{\mathbf{\Lambda}}(i,0)}/\mu_i$. After having run the algorithm, one should therefore solve the system of $|\mathbf{\Lambda}_2|$ linear equations with $|\mathbf{\Lambda}_2|$ unknowns of the form

$$\sum_{l=1}^{|\mathbf{\Lambda}_2|} M_{ml} \rho_{i(l)}^{\mathbf{\Lambda}} = \alpha_m, \tag{2.5}$$

where

$$M_{ml} = \sum_{k \succeq i(m)} \mathbf{1}_{\{r^{\mathbf{\Lambda}}(k,0) = i(l)\}} \frac{\mu_{i(l)}}{\mu_k} \tag{2.6}$$

is the ml-th entry of an $|\mathbf{\Lambda}_2| \times |\mathbf{\Lambda}_2|$ matrix M and

$$\alpha_m = 1 - \sum_{\substack{j \succ i(m): \\ j \in \mathbf{\Lambda}_0}} \rho_j^{\mathbf{\Lambda}}. \tag{2.7}$$

For a given set of saturated queues $\mathbf{\Lambda}$, we will next construct a graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ that has cycles whenever the resulting set $\mathbf{\Lambda}_2$ is non-empty.

Construction of the graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ for monotonic $\mathbf{\Lambda}$
Let $\mathbf{\Lambda}$ be given. The directed graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ has set of vertices $\mathbf{V} = \{1, \ldots, p\}$ equal to all classes, and the following set of arrows $\mathbf{A}$. For any $i, j \in \overline{\mathbf{\Lambda}}$ we draw an arrow $i \to j$, whenever $i \succ r^{\mathbf{\Lambda}}(j,0)$, $r^{\mathbf{\Lambda}}(j,0) \in \mathfrak{p}(\mathbf{\Lambda})$ and $j \neq r^{\mathbf{\Lambda}}(i,\sigma)$, for $\sigma_i^{\mathbf{\Lambda}} \leq \sigma \leq \ell_i^{\mathbf{\Lambda}}$. In words: there is an arrow $i \to j$ whenever i has priority over the $\mathbf{\Lambda}$-source of j (provided it is active) in the induced chain $\xi_t^{\mathbf{\Lambda}}$ with saturated queues $\mathbf{\Lambda}$, and j is no successor of i on its $\mathbf{\Lambda}$-route. The latter requirement seems less natural in connection with the above construction, but will appear to slightly reduce technicalities later-on. For instance, it bars the existence of self-loops $i \to i$.

Remark that we have *reversed* the direction of arrows as compared to the graph $\mathcal{G}^e$! This is more natural in the light of an interpretation of these graphs as a special class of influence structure: if class i load changes, then the loads of all classes corresponding to vertices on *downstream* paths, passing through class $r^{\mathbf{\Lambda}}(i,\sigma)$s vertex may be affected in a *non-monotonic* way, for any $\sigma \geq \sigma_i^{\mathbf{\Lambda}}$.

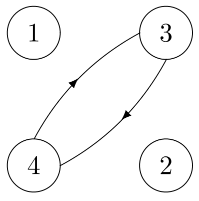
For applicability reasons, at this point we will also introduce a simply computable graph $\mathcal{G}(\mathbf{\Lambda})$, $\mathbf{\Lambda}$ monotonic, having cycles if and only if $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$ contains a cycle for some monotonic $\mathbf{\Lambda}' \supseteq \mathbf{\Lambda}$. This graph does not play any role in further proofs.

The graph $\mathcal{G}(\mathbf{\Lambda})$ has the set of vertices $\mathbf{V} = \{1, \dots, p\}$ and the set of arrows $\mathbf{A}$ such that $i \to j$, $i \neq j$, $i, j \in \overline{\mathbf{\Lambda}}$, belongs to $\mathbf{A}$ whenever $i \succ r^{\mathbf{\Lambda}}(j, \sigma')$ for some $0 \leq \sigma' < \sigma_j^{\mathbf{\Lambda}}$, satisfying $r^{\mathbf{\Lambda}}(j,\sigma) \not\succ j$, for $\sigma' < \sigma < \sigma_j^{\mathbf{\Lambda}}$.

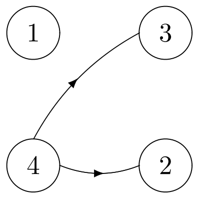
The last requirement merely states that a class having priority over a $\mathbf{\Lambda}$-ancestor cannot have an in-arrow from the same vertex as its $\mathbf{\Lambda}$-ancestor. By $\mathcal{G}$ we will denote $\mathcal{G}(\emptyset)$.

Note that, none of these graphs have self-loops. Therefore cycles have minimum length 2. Also, note that $\mathcal{G}\big(\emptyset, \mathfrak{p}(\emptyset)\big)$ has no arrows, since $\mathfrak{p}(\emptyset) = \emptyset$. We will give an example of the various graphs in the toy example.

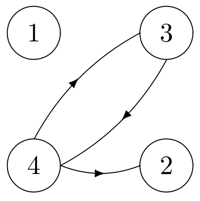
Toy example: continuation(v)



Graph $\mathcal{G}\big(\{1,2\}, \mathfrak{p}(\{1,2\})\big) = \mathcal{G}\big(\{1,2\}\big)$



Graph $\mathcal{G}\big(\{1\}, \mathfrak{p}(\{1\})\big)$



Graph $\mathcal{G} = \mathcal{G}\big(\{1\}\big)$

The graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ has cycles only, when $\mathbf{\Lambda} = \{1,2\}$. The arrow $4 \to 2$ is a typical example of a superfluous arrow, which does not represent a situation in the algorithm.

The following theorem connects the objects introduced in the above.

Theorem 2.1 *Let $\mathbf{\Lambda}$ be monotonic. The following assertions hold.*

i) *For the $\mathbf{\Lambda}$-load equations to have no or a non-unique solution, $\mathbf{\Lambda}_2$ must be necessarily non-empty.*

ii) *The set $\mathbf{\Lambda}_2$ is non-empty if and only if there is a cycle in the graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$.*

iii) *If the graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ contains a cycle, then it contains a cycle, $i_1 \to i_2 \to \cdots i_k \to i_{k+1} = i_1$ of length k say, with the following properties for any two classes i_j and i_l belonging to it: (1) $n(i_j) \neq n(i_l)$, for $l \neq j \bmod(k)$; (2) $r^{\mathbf{\Lambda}}(i_j, 0) \neq r^{\mathbf{\Lambda}}(i_l, 0)$, or in words, i_j and i_l belong to different $\mathbf{\Lambda}$-routes whenever $i_l \neq i_j$. Summarising: there is a cycle separating $\mathbf{\Lambda}$-routes and nodes.*

iv) *The graph $\mathcal{G}(\mathbf{\Lambda})$ contains a cycle if and only if $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$ contains a cycle for some $\mathbf{\Lambda}' \supseteq \mathbf{\Lambda}$.*

Proof. Statement (i) follows from the algorithm.

Proof of (ii). First assume that $\mathbf{\Lambda}_2$ is non-empty. Let $\mathbf{\Lambda}_2$ consist of the classes $i(l)$, $l = 1, \dots, |\mathbf{\Lambda}_2|$. For any $i \in \mathbf{\Lambda}_2$ there is at least one class $j \in \overline{\mathbf{\Lambda}_0 \cup \mathbf{\Lambda}_1 \cup \mathbf{\Lambda}_2}$ with the following property $\boldsymbol{P(i)}$: $j \succ i$, $r^{\mathbf{\Lambda}}(j, 0) \in \mathbf{\Lambda}_2$.

By Remark 2.1, $r^{\mathbf{\Lambda}}(j, 0) \neq i$. Hence $\mathbf{\Lambda}_2$ must consist of at least 2 different classes (from different nodes by definition).

Set $i_1 = i(1)$ and choose $j_1 \in \overline{\mathbf{\Lambda}_0 \cup \mathbf{\Lambda}_1 \cup \mathbf{\Lambda}_2}$ with property $\boldsymbol{P(i_1)}$. Take $i_2 = r^{\mathbf{\Lambda}}(j_1, 0)$. Then $i_2 \neq i_1$. Choose next $j_2 \in \overline{\mathbf{\Lambda}_0 \cup \mathbf{\Lambda}_1 \cup \mathbf{\Lambda}_2}$ with property $\boldsymbol{P(i_2)}$. Let $i_3 = r^{\mathbf{\Lambda}}(j_2, 0) \in \mathbf{\Lambda}_2$. Note, that graph $\mathcal{G}\big(\Lambda, \mathfrak{p}(\Lambda)\big)$ contains the arrow $j_2 \to j_1$.

When $i_3 = i_1$ we obtain the cycle $j_2 \to j_1 \to j_2$. Otherwise we continue the construction. Since $\mathbf{\Lambda}_2$ is finite, we end up with a sequence $i_1, j_1, \dots, i_k$, where $i_k \in \{i_1, \dots, i_{k-2}\}$, say $i_k = i_l$, The sequence

$$j_l \to j_{k-1} \to j_{k-2} \to \dots \to j_l$$

is the desired cycle.

The reverse statement that the existence of a cycle in $\mathcal{G}(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda}))$ implies $\mathbf{\Lambda}_2$ not to be empty, follows from the algorithm. The proof is by reductio ad absurdam: we will prove that $\mathbf{\Lambda}_2 = \emptyset$ implies that no cycle can exist.

Carry out steps 1 and 2 of the algorithm. Put $A_0 = \mathbf{\Lambda}_0 \setminus \mathbf{\Lambda}$, the classes of which have no in-arrows. By assumption on $\mathbf{\Lambda}_2$ being empty after having run the algorithm, step 3 of the algorithm successively eliminates all classes from $\mathfrak{p}(\mathbf{\Lambda})$, say in the order $i_1, \ldots, i_p$, where $p = |\mathfrak{p}(\mathbf{\Lambda})|$. For $n = 1, \ldots, p$, put $A_n = \{i \notin \mathbf{\Lambda} \,|\, r^{\mathbf{\Lambda}}(i, 0) = i_n\}$. One has $A_n \cap \cup_{l=0}^{n-1} A_l = \emptyset$ and $\cup_{l=0}^{p} A_l \supset \overline{\mathbf{\Lambda}}$.

Inspection yields that class $i \in A_l$, can only have in-arrows from classes $j \in \cup_{m=0}^{l-1} A_m$, or from a class $j \succ i_l$, with $r^{\mathbf{\Lambda}}(j, 0) = i_l$ and $i = r^{\mathbf{\Lambda}}(j, \sigma)$ for some $\sigma < \sigma_j^{\mathbf{\Lambda}}$. That is, $j \in A_l$ as well. This implies that the graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ cannot have any cycles at all.

Proof of (iii). We assume that $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ contains a cycle $j_1 \to j_2 \to \cdots \to j_m \to j_{m+1} = j_1$. By taking an appropriate sub-cycle, one may assume that $j_l \neq j_s$ for $l \neq s \bmod(m)$. To show the first property (1), assume that $n(j_i) = n(j_l)$ for some j_i, j_l, $1 \leq i < l \leq m$. By property (1), we may assume that $j_i \neq j_l$. If the graph contains the arrow $j_i \to j_{l+1}$, then we may choose the sub-cycle $j_1 \to \cdots j_i \to j_{l+1} \to \cdots j_{m+1} = j_1$.

Suppose that the graph does not contain this arrow. This can only occur in two cases. This first being that $l + 1 = i \bmod(m)$. Because of property (1), $j_{l+1} \neq j_i$, unless the cycle consists of 2 arrows only, so that $m = 2$. The cycle is of the form $j_1 \to j_2 \to j_1$ with j_1 and j_2 belonging to the same $\mathbf{\Lambda}$-route, since $r^{\mathbf{\Lambda}}(j_1, 0) = r^{\mathbf{\Lambda}}(j_2, 0)$. Then either j_2 is a $\mathbf{\Lambda}$-successor of j_1 s or vice versa. Not both arrows can exist by construction.

The second case, is when j_{l+1} is a $\mathbf{\Lambda}$-successor of j_i s, and $l+1 \neq i \bmod(m)$. Since j_{l+1} is not a $\mathbf{\Lambda}$-successor of j_l s, the graph must contain the arrow $j_l \to j_i$, and we can replace the present cycle by its sub-cycle $j_l \to j_i \to \cdots j_l$.

Finally, we prove property (2). Assume property (1). Let j_i and j_l belong to the same $\mathbf{\Lambda}$-route, $1 \leq i < l \leq m$. Then $r^{\mathbf{\Lambda}}(j_i, 0) = r^{\mathbf{\Lambda}}(j_l, 0)$, and so $n(j_{(i-1)\bmod(m)}) = n(j_{l-1})$. This cannot happen, once property (1) is satisfied.

Proof of (iv). Let $\mathcal{G}(\mathbf{\Lambda})$ contain the cycle $i_1 \to \cdots i_k \to i_1$. We will count indices modulo k. If this cycle contains no further sub-cycle, then it can be chosen to have the following two properties for any $l = 1, \ldots, k$.

P(i) $n(i_j) \neq n(i_l)$, for $l \neq j \bmod(k)$.

P(ii) Note that the existence of the arrow $i_l \to i_{l+1}$, implies that there is $i'_l \in \mathfrak{p}(\mathbf{\Lambda}) \cup \overline{\mathbf{\Lambda}}$ with $i'_l \prec i_l$ and $i'_l = r^{\mathbf{\Lambda}}(i_{l+1}, \sigma')$, for some $\sigma' < \sigma_{i_{l+1}}^{\mathbf{\Lambda}}$, such that $r^{\mathbf{\Lambda}}(i_{l+1}, \sigma) \not\succ i_{l+1}, i_l$ for $\sigma' < \sigma < \sigma_{i_{l+1}}^{\mathbf{\Lambda}}$. Then for no $m \neq l, l+1$, one can have $i_m \succ r^{\mathbf{\Lambda}}(i_{l+1}, \sigma)$, for some $\sigma' < \sigma < \sigma_{i_{l+1}}^{\mathbf{\Lambda}}$.

We will prove **P(i)**. Clearly, by choosing an appropriate sub-cycle, one may assume that $i_l \neq i_j$, $l \neq j \bmod(k)$. Suppose that $n(i_l) = n(i_j)$ for some $i_l \neq i_j$. If $i_l \succ i_j$, then by construction of $\mathcal{G}(\mathbf{\Lambda})$, there must be an arrow $i_l \to i_{j+1}$. Hence, there is a sub-cycle $i_l \to i_{j+1} \to \cdots \to i_l$, contradicting minimality. The case $i_j \succ i_l$ is similar.

Suppose that **P(ii)** does not hold. Then for some i_m, $m \neq l, l+1$, and σ, with $\sigma' < \sigma < \sigma_{i_{l+1}}^{\mathbf{\Lambda}}$, one has $i_m \succ r^{\mathbf{\Lambda}}(i_{l+1}, \sigma)$. By construction of $\mathcal{G}(\mathbf{\Lambda})$, the arrow $i_m \to i_{l+1}$ must occur as well. So, there is a sub-cycle $i_m \to i_{l+1} \to \cdots \to i_m$, contradicting assumed minimality.

Let $i_1 \to \cdots \to i_k \to i_1$ be a cycle of $\mathcal{G}(\mathbf{\Lambda})$ without sub-cycles and with the above properties. Then the graph $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$, with $\mathbf{\Lambda}' = \mathbf{\Lambda} \cup_{l=1}^{k} \{j \preceq i'_l\}$, contains this cycle as well. Let us check this. Consider the arrow $i_l \to i_{l+1}$ from the cycle in $\mathcal{G}(\mathbf{\Lambda})$.

Let σ' be given by $i'_l = r^{\mathbf{\Lambda}}(i_{l+1}, \sigma')$. The above properties guarantee that $r^{\mathbf{\Lambda}}(i_{l+1}, \sigma\big) \in \overline{\mathbf{\Lambda}'}$, for $\sigma' \leq \sigma \leq \sigma_{i_{l+1}}^{\mathbf{\Lambda}}$. The only reason that the arrow $i_l \to i_{l+1}$ might not exist in $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$, is when $i_{l+1} = r^{\mathbf{\Lambda}'}(i_l, \sigma)$ for some $\sigma > \sigma_{i_l}^{\mathbf{\Lambda}'}$. Assume this to be true for σ. The graph $\mathcal{G}(\mathbf{\Lambda})$ cannot contain the arrow $i_{l-1} \to i_{l+1}$, otherwise the cycle would have contained a sub-cycle.

Let us consider the $\mathbf{\Lambda}$-route of i_l. It passes through i'_{l-1} and i'_l (presumably we include the $\mathbf{\Lambda}$-input type of i_l). The class i'_l cannot lie on the $\mathbf{\Lambda}$-route after i'_{l-1} and before i_l, since then the arrow $i_{l-1} \to i_l$ would not exist in $\mathcal{G}(\mathbf{\Lambda})$ by construction. It cannot not lie after i_l (and by construction before i_{l+1}), since then i_l and i_{l+1} would have belonged different $\mathbf{\Lambda}'$-routes and the graph $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$ would have contained the arrow $i_l \to i_{l+1}$. Hence i'_l occurs before i'_{l-1}.

Assuming this, the only reason why the graph $\mathcal{G}(\mathbf{\Lambda})$ can*not* contain the arrow $i_{l-1} \to i_{l+1}$, is that $r^{\mathbf{\Lambda}}(i'_{l-1}, \sigma') \prec i_{l+1}$ for some $\sigma_{i'_{l-1}}^{\mathbf{\Lambda}} < \sigma' < \sigma_{i_l}^{\mathbf{\Lambda}}$. But then $\mathcal{G}(\mathbf{\Lambda})$ cannot contain the arrow $i_l \to i_{l+1}$, since i'_l lies before i'_{l+1} on the same $\mathbf{\Lambda}$-route. Note that this argument is valid in case $i_{l+1} = i_{l-1}$ as well, i.e. $k = 2$.

For the reverse statement, assume the existence of a cycle $i_1 \to \cdots \to i_k$ in a graph $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$, $\mathbf{\Lambda}' \supset \mathbf{\Lambda}$. We can choose it to satisfy properties (1) and (2) from (iii). Moreover, choose it to have satisfy a 'minimal priority property': for any $j \in \mathcal{J}\big(n(i_l)\big)$ with $i_{l-1} \to j \to i_{l+1}$ arrows in $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$ for some $j \in n(i_l)$, we have $j \succeq i_l$, for any $1 \le l \le k$.

Suppose that an arrow $i_l \to i_{l+1}$ does not belong to $\mathcal{G}(\mathbf{\Lambda})$. Let $i_l' = \mathfrak{p}(\mathbf{\Lambda}') \cap \mathcal{J}\big(n(i_l)\big)$. Then the $\mathbf{\Lambda}$-route of i_l' passes some class $j \prec i_{l+1}$ before i_{l+1}. Necessarily $j \in \overline{\mathbf{\Lambda}'}$. The separation properties imply $j \to i_{l+2}$ to be an arrow of $\mathcal{G}\big(\mathbf{\Lambda}', \mathfrak{p}(\mathbf{\Lambda}')\big)$. This contradicts the properties of the initial cycle. QED

Even when there is a cycle in the graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$, the solution of the $\mathbf{\Lambda}$-load equations may still be unique and feasible. We give an example.

Toy example: continuation(vi)
As mentioned in continuation(v), the graph $\mathcal{G}(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ has a cycle only for $\mathbf{\Lambda} = \{1, 2\}$. The $\mathbf{\Lambda}$-load equations are

$$\begin{aligned} \mu_3\rho_3^{\mathbf{\Lambda}} &= \mu_1\rho_1^{\mathbf{\Lambda}} \\ \mu_4\rho_4^{\mathbf{\Lambda}} &= \mu_2\rho_2^{\mathbf{\Lambda}} \\ \rho_1^{\mathbf{\Lambda}} + \rho_4^{\mathbf{\Lambda}} &= 1 \\ \rho_2^{\mathbf{\Lambda}} + \rho_3^{\mathbf{\Lambda}} &= 1. \end{aligned} \tag{2.8}$$

The corresponding sub-network is stable for $\mu_3 > \mu_1$ and $\mu_4 > \mu_2$. This can be seen from continuation(iii), in conjunction with standard results on stability of face-homogeneous random walks on $\mathbf{Z}_+$.

By the reduction algorithm $\mathbf{\Lambda}_1 = \{3, 4\}$. Step 3 does not do anything and so we have $\mathbf{\Lambda}_2 = \mathfrak{p}(\mathbf{\Lambda}) = \{1, 2\}$. By Remark 2.2, we plug in (2.2) so that

$$\rho_3^{\mathbf{\Lambda}} = \frac{\mu_1}{\mu_3}\rho_1^{\mathbf{\Lambda}}, \quad \rho_4^{\mathbf{\Lambda}} = \frac{\mu_2}{\mu_4}\rho_2^{\mathbf{\Lambda}}.$$

Then equation (2.5) reduces to

$$\begin{pmatrix} 1 & \dfrac{\mu_2}{\mu_4} \\ \dfrac{\mu_1}{\mu_3} & 1 \end{pmatrix} \begin{pmatrix} \rho_1^{\mathbf{\Lambda}} \\ \rho_2^{\mathbf{\Lambda}} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

This has a unique solution

$$\rho_1^{\mathbf{\Lambda}} = \frac{\mu_3\mu_4 - \mu_3\mu_2}{\mu_3\mu_4 - \mu_1\mu_2}, \quad \rho_2^{\mathbf{\Lambda}} = \frac{\mu_4\mu_3 - \mu_4\mu_1}{\mu_3\mu_4 - \mu_1\mu_2}.$$

whenever

$$\mu_1\mu_2 \neq \mu_3\mu_4.$$

This solution is feasible whenever either $\mu_3 > \mu_1$ and $\mu_4 > \mu_2$ or $\mu_3 < \mu_1$ and $\mu_4 < \mu_2$. The second case corresponds to an instable sub-network (see again continuation(iii)).

No solution or a non-unique solution occurs only when $\mu_1\mu_2 = \mu_3\mu_4$. In particular, there is a non-unique solution when in addition $\mu_1 = \mu_3$ and $\mu_2 = \mu_4$: queues 2 and 4 are both null-recurrent. In that case, any $\rho_1^{\mathbf{\Lambda}}, \rho_2^{\mathbf{\Lambda}} > 0$ with $\rho_1^{\mathbf{\Lambda}} + \rho_4^{\mathbf{\Lambda}} = 1$, $\rho_3^{\mathbf{\Lambda}} = \rho_1^{\mathbf{\Lambda}}$, $\rho_4^{\mathbf{\Lambda}} = \rho_2^{\mathbf{\Lambda}}$ are a feasible solution.

There is no solution when in addition $\mu_1 \neq \mu_3$ and $\mu_2 \neq \mu_4$. This case implies that one queue must be stable whilst the other is transient.

For all other monotonic sub-networks the corresponding load equations have a unique solution, since the corresponding graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ has no cycles. Note however, that $\mathcal{G}(\{1\})$ and $\mathcal{G}(\{2\})$ do contain cycles, and for the corresponding sub-networks the phenomena of scattering and discrepancy of local and global stability appear. Let us compute the $\mathbf{\Lambda}$-loads $\rho^{\mathbf{\Lambda}}$. We will always write it as a four-dimensional vector.

* $\mathbf{\Lambda = \{1, 2, 3, 4\}}$**.** $\xi_t^{\{1,2,3,4\}}$ is stable and $\rho^{\{1,2,3,4\}} = (0, 0, 1, 1)^T$.

* $\mathbf{\Lambda = \{1, 2, 3\}}$**.** $\xi_t^{\{1,2,3\}}$ is stable and $\rho^{\{1,2,3\}} = (1, 0, 1, 0)^T$.

* $\mathbf{\Lambda = \{1, 2, 4\}}$**.** $\xi_t^{\{1,2,4\}}$ is stable and $\rho^{\{1,2,4\}} = (0, 1, 0, 1)^T$.

* $\mathbf{\Lambda = \{1, 4\}}$**.** $\xi_t^{\{1,4\}}$ is stable and $\rho^{\{1,4\}} = (0, 0, 0, 1)^T$. Here node 2 is empty with probability 1. Since $\mathbf{\Lambda} \cap \mathcal{J}(2) = \emptyset$, the $\{1, 4\}$-load equations do not require $\rho_2^{\mathbf{\Lambda}} + \rho_3^{\mathbf{\Lambda}} = 1$.

* **$\mathbf{\Lambda} = \{\mathbf{2},\mathbf{3}\}$.** $\xi_t^{\{2,3\}}$ is stable when $\lambda_1 < \mu_1$ and $\rho^{\{2,3\}} = \big((\lambda_1/\mu_1), 0, 1, 0\big)^T$. This solution is feasible if and only if $\lambda_1 < \mu_1$, which is equivalent to stability.

* **$\mathbf{\Lambda} = \{\mathbf{1}\}$.** We need to solve the relations

$$\mu_1\rho_1^{\{1\}} = \mu_3\rho_3^{\{1\}} = \mu_2\rho_2^{\{1\}} = \mu_4\rho_4^{\{1\}}, \quad \rho_1^{\{1\}} + \rho_4^{\{1\}} = 1.$$

This gives

$$\rho^{\{1\}} = \begin{pmatrix} \dfrac{\mu_4}{\mu_1+\mu_4} \\ \dfrac{\mu_1}{\mu_2}\dfrac{\mu_4}{\mu_1+\mu_4} \\ \dfrac{\mu_1}{\mu_3}\dfrac{\mu_4}{\mu_1+\mu_4} \\ \dfrac{\mu_1}{\mu_1+\mu_4} \end{pmatrix}.$$

In this case, the solution is feasible if and only if $\rho_2^{\{1\}} + \rho_3^{\{1\}} < 1$ or

$$\frac{\mu_1\mu_4}{\mu_1+\mu_4} < \frac{\mu_2\mu_3}{\mu_2+\mu_3} \text{ or } \frac{1}{\mu_2} + \frac{1}{\mu_3} < \frac{1}{\mu_1} + \frac{1}{\mu_4}.$$

However, this does not guarantee stability as we shall see in part (II) of this paper. continuation of the example(vii)).

* **$\mathbf{\Lambda} = \{\mathbf{2}\}$.** Now we have the relations

$$\lambda_1 = \mu_1\rho_1^{\{2\}} = \mu_3\rho_3^{\{2\}}, \quad \mu_2\rho_2^{\{2\}} = \mu_4\rho_4^{\{2\}}, \quad \rho_2^{\{2\}} + \rho_3^{\{2\}} = 1,$$

so that

$$\rho^{\{2\}} = \begin{pmatrix} \dfrac{\lambda_1}{\mu_1} \\ 1 - \dfrac{\lambda_1}{\mu_3} \\ \dfrac{\lambda_1}{\mu_3} \\ \dfrac{\mu_2}{\mu_4}\dfrac{\mu_3-\lambda_1}{\mu_3} \end{pmatrix}.$$

This is feasible if and only if $\rho_4^{\{2\}} > 0$ and $\rho_1^{\{2\}} + \rho_4^{\{2\}} < 1$, or $\mu_3 > \lambda_1$ and $\lambda_1(\mu_3\mu_4 - \mu_1\mu_2) < \mu_1\mu_3(\mu_4 - \mu_2)$. This does not guarantee stability neither.

This toy example suggests that feasibility of the load vector and stability for a sub-network with a monotonic set $\mathbf{\Lambda}$ of saturated queues and an acyclic graph $\mathcal{G}(\mathbf{\Lambda})$ are equivalent. As another motivation, we present an example treated by Dai that we will adapt to the notation in this paper.

Example 2.1 (Re-entrant line [5], §7) Assume $a(p) = 0$, $\lambda_p \neq 0$, and $r_{l\,l-1} = 1$. By construction, if $n(j) = n(l)$ and $j \succ l$, then $j > l$.

Consider any monotonic set $\mathbf{\Lambda}$ of saturated queues. If $j \to l$ is an arrow in the graph $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$, then necessarily $j > l$. Hence, $\mathcal{G}\big(\mathbf{\Lambda}, \mathfrak{p}(\mathbf{\Lambda})\big)$ cannot contain any cycle. As a consequence of Theorem 2.1(iv), the graph $\mathcal{G}(\mathbf{\Lambda})$ is acyclic.

Local stability for this network implies overall stability, see Dai [5].

Thie desired equivalence is postponed to the next part. At this point, the only question concerning the relation between solutions of the $\mathbf{\Lambda}$-load equations and stability of the corresponding sub-network or induced chain, that we can answer, is the following theorem. The first part is essentially due to [7].

Theorem 2.2 *Let $\mathbf{\Lambda}$ be monotonic. Consider the sub-network $\xi_t^{\mathbf{\Lambda}}$ with set $\mathbf{\Lambda}$ of saturated queues.*

1. *Suppose that $\xi_t^{\mathbf{\Lambda}}$ has an ergodic closed class. Then aborption into the set of essential states from inessential states (if any) takes place in finite expected time. Hence $\xi_t^{\mathbf{\Lambda}}$ is stable.*
2. *The sub-network is not stable whenever there is no feasible solution or a non-unique solution to the $\mathbf{\Lambda}$-load equations.*

Proof. The first statement follows similarly to [7] Proposition 2.5. The assertion there was for the whole network under the assumption that $\rho_n \leq 1$ for each node n. Here we have to assume ergodicity, in order to ensure that the unique (by the second part of this theorem) feasible solution to the $\mathbf{\Lambda}$-load equations has the correct probabilistic meaning.

We prove the second statement. If there is no solution to the $\mathbf{\Lambda}$-load equations, then clearly the closed class of sub-network cannot be ergodic.

Suppose that the solution is non-unique and that the closed class is ergodic. Then the set $\mathbf{\Lambda}_2$ from the algorithm is non-empty, say it is the set of classes $\{i(1), \ldots, i(|\mathbf{\Lambda}_2|)\}$.

The $\mathbf{\Lambda}$-load equations again reduce to (2.5) with matrix M given by (2.6) and right-hand side α by (2.7). By non-uniqueness M has linearly dependent rows. Say the maximum number of linearly independent rows equals $k < |\mathbf{\Lambda}_2|$ and suppose that these are the first k rows of the M. The $k+1$-th row is a linear combination of the previous k rows, with unique coefficients $c_i \neq 0$, $i = 1, \ldots, k$, say:

$$\text{row}_{k+1} = c_1 \,\text{row}_1 + \cdots + c_k \,\text{row}_k.$$

Since we have at least one solution to the $\mathbf{\Lambda}$-load equations, the right-handside must satisfy a similar relation as the rows

$$\alpha_{k+1} = c_1\alpha_1 + \cdots + c_k\alpha_k.$$

The idea of the proof is to add a new class to for instance node $n\big((i(1)\big)$ of the sub-network, without losing ergodicity and such that the corresponding $\mathbf{\Lambda}$-load equations have no solution anymore.

So, the extended sub-network will have an extra class $i(*)$ in node $n(i(1))$: it arrives from outside the network at rate $\lambda_{i(*)}$. It has priority over all other classes at the same node and so at will be immediately served at rate $\mu_{i(*)} > \lambda_{i(*)}$. After having been served, it leaves the system. Its route therefore consists of 1 element only: $i(*)$. By choosing these parameters small enough, we can again consider a first order approximation with transition probabilities to other states equal to the rates.

To distinguish between the original and extended sub-networks, we will use superscript $\mathbf{\Lambda}$ and $*$ respectively. So state $x^{\mathbf{\Lambda}}$ is a state of the original sub-network and $x^* = x^{\mathbf{\Lambda}} + x_{i(*)}e\big(i(*)\big)$ is a state of the extended one.

The class structure is not affected by this extension. By stability, there exists a Lyapunov function $f : \mathbf{Z}_+^{p-|\mathbf{\Lambda}|} \to [1, \infty)$ (cf. [9]) for the original subnetwork, and $\epsilon > 0$, such that

$$\sum_{y^{\mathbf{\Lambda}}} p^{\mathbf{\Lambda}}_{x^{\mathbf{\Lambda}}y^{\mathbf{\Lambda}}} f(y^{\mathbf{\Lambda}}) \leq f(x^{\mathbf{\Lambda}}) - \epsilon, \quad x^{\mathbf{\Lambda}} \neq 0(\mathbf{\Lambda}).$$

Additionally, for the $\mathsf{M/M/1}$-queue with arrival rate $\lambda_{i(*)}$ and service rate $\mu_{i(*)}$ there exists an exponential Lyapunov function $g(x) = \exp\{\gamma x\}$, $x \in \mathbf{Z}_+$ for some constant $\gamma > 0$ (cf. [10], [14]). More precisely, denote

$$p^e_{xy} = \begin{cases} \lambda_{i(*)}, & y = x+1 \\ \mu_{i(*)}, & y = x-1, x > 0 \\ 1 - \lambda_{i(*)} - \mathbf{1}_{\{x>0\}}\mu_{i(*)}, & y = x. \end{cases}$$

Then there exists a positive constant $\eta < 1$ such that

$$\sum_y p^e_{xy} g(y) \leq (1-\eta) g(x), \quad x \neq 0.$$

To show stability for the extended sub-network, we will construct a Lyapunov function $h : \mathbf{Z}_+^{p+1-|\mathbf{\Lambda}|} \to [1, \infty)$, such that

$$\sum_{y^*} p^*_{x^*y^*} h(y^*) \leq h(x^*) - \delta, \quad x^* \notin A, \tag{2.9}$$

for some $\delta > 0$ and some finite set A of states for the extended sub-network.

Choose

$$h(x^*) = \beta f(x^{\mathbf{\Lambda}}) + g(x_{i(*)}),$$

where $\beta > 0$ still has to be determined. By the additive structure of both the transition probabilities as well as our choice of Lyapunov function, we have that

$$\sum_{y^*} p^*_{x^*y^*} h(y^*) = \sum_{y^{\mathbf{\Lambda}}} p^{\mathbf{\Lambda}}_{x^{\mathbf{\Lambda}}y^{\mathbf{\Lambda}}} \beta f(y^{\mathbf{\Lambda}}) + \sum_{y_{i(*)}} p^e_{x_{i(*)}y_{i(*)}} g(y_{i(*)}).$$

For any $\beta > 0$ we have for $x^{\mathbf{\Lambda}} \neq 0(\mathbf{\Lambda})$, $x_{i(*)} \neq 0$

$$\sum_{y^*} p^*_{x^*y^*} h(y^*) \leq \beta f(x^{\mathbf{\Lambda}}) + g(x_{i(*)}) - \big(\beta\epsilon + \eta g(x_{i(*)})\big) \leq h(x^*) - (\beta\epsilon + \eta). \tag{2.10}$$

Choose now $\beta > 1$ with

$$\beta > \frac{2}{\epsilon} \sum_y p^e_{0y}\big(g(y) - g(0)\big).$$

Then for $x^{\mathbf{\Lambda}} \neq 0(\mathbf{\Lambda})$ and $x_{i(*)} = 0$ we have that

$$\sum_{y^*} p^*_{x^*y^*} h(y^*) \leq h(x^*) - \frac{\beta\epsilon}{2}. \tag{2.11}$$

Finally, choose a finite set of states $\{1, \ldots, a\}$, such that

$$\eta \exp\{\gamma a\} \geq 2\beta \sum_{y^{\mathbf{\Lambda}}} p^{\mathbf{\Lambda}}_{0(\mathbf{\Lambda})y^{\mathbf{\Lambda}}} \Big(f(y^{\mathbf{\Lambda}}) - f(0(\mathbf{\Lambda}))\Big).$$

Setting $A = \{0(\mathbf{\Lambda}) + e(i(*))k, k = 0, \ldots, a\}$, we have for states $x^* \notin A$ with $x^{\mathbf{\Lambda}} = 0(\mathbf{\Lambda})$

$$\sum_{y^*} p^*_{x^*y^*} h(y^*) \leq h(x^*) - \eta \exp\{\gamma(a+1)\} + \frac{1}{2}\eta \exp\{\gamma a\} \leq h(x^*) - \frac{\eta}{2}. \tag{2.12}$$

Combining (2.10), (2.11) and (2.12) yields (2.9) for $\delta = \frac{1}{2}\min\{\beta\epsilon, \eta\}$.

Now we have a stable extended sub-network. Solving the $\mathbf{\Lambda}$-load equations for this network, yields the same set $\mathbf{\Lambda}_2$ and matrix M in (2.5). However, the right-hand side in (2.5) has slightly changed: α_1 from (2.7) has changed into $\alpha_1 - (\lambda_{i(*)}/\mu_{i(*)})$. So the linear relation between the first $k+1$ rows of M does not apply to the right-hand side anymore. The $\mathbf{\Lambda}$-load equations cannot have a solution for the extended sub-network. So it cannot be stable. As a consequence the original sub-network cannot have been stable. QED

Clearly, Theorem 2.2 (2) seems a trivial assertion. The problem is that it is not clear whether a solution to the load equations is extendable to a solution of the equations for the stationary distribution of the sub-network with saturated queues $\mathbf{\Lambda}$: $\pi P^{\mathbf{\Lambda}} = \pi$.

Finally, I would lijke point out that this can be used to reduce the number of 'false' fluid solutions. Nevertheless, in the companion paper we will study the full set of solutions.

References

[1] D.D. Botvich and A.A. Zamyatin (1995), Fluid approximations for conservative networks. *Markov Proc. Relat. Fields* **1**, 113–141.

[2] M. Bramson (1999), A stable queueing network with unstable fluid model. *Ann. Appl. Prob.* **9**, 818–853.

[3] H. Chen (1995), Fluid approximations and stability of multiclass queueing networks: work-conserving disciplines. *Ann. Appl. Prob.* **5**, 636–665.

[4] H. Chen and H. Ye (1999), Piecewise linear Lyapunov function for the stability of priority fluid networks. In: *Proc. 38th Conf. on Decision and Control, Phoenix.*

[5] J. G. Dai (1995), On positive Harris recurrence of multiclass queueing networks: a unified approach via fluid limit models. *Ann. Appl. Prob* **5**, 49–77.

[6] J. G. Dai and G. Weiss (1996), Stability and instability of fluid models for reentrant lines. *Math. Operat. Res.* **21**, 115–135.

[7] V. Dumas (1996), Essential faces and stability conditions of multiclass networks with priorities. Technical report 3030, INRIA. *To appear in: Markov Proc. Relat. Fields.*

[8] V. Dumas (1997), A multiclass network with non-linear, non-convex, non-monotonic stability conditions. *QUESTA* **25**, 1–43.

[9] G. Fayolle, V.A. Malyshev, and M.V. Menshikov (1995), *Constructive Theory of countable Markov Chains.* Cambridge University Press, Cambridge.

[10] A. Hordijk and F.M. Spieksma (1992), On ergodicity and recurrence properties of a Markov chain with an application to an open Jackson network. *Adv. Appl. Prob.* **24**, 343–376.

[11] S.H. Lu and P.R. Kumar (1991), Distributed scheduling based on due dates and buffer priorities. *IEEE Trans. on Autom. Control* **36**, 1406–1416.

[12] V.A. Malyshev and M.V. Menshikov (1981), Ergodicity, continuity and analyticity of countable Markov chains. *Trans. Moscow Math. Soc.* **1**, 1–48.

[13] A.N. Rybko and A.L. Stolyar (1992), Ergodicity of stochastic processes describing the operation of open queueing networks. *Probl. Peredachi Inf.* **28**, 3–26.

[14] F.M. Spieksma and R.L. Tweedie (1994), Strengthening ergodicity to geometric ergodicity of Markov chains. *Stoch. Models* **10**, 45–75.