"Modelling queueing networks with blocking using probability mass fitting"

Tancrez, Jean-Sébastien

Abstract
In this thesis, we are interested in the modelling of queueing networks with finite buffers and with general service time distributions. Queueing networks models have shown to be very useful tools to evaluate the performance of complex systems in many application fields (manufacturing, communication networks, traffic flow, etc.). In order to analyze such networks, the original distributions are most often transformed into tractable distributions, so that the Markov theory can then be applied. Our main originality lies in this step of the modelling process. We propose to discretize the original distributions by probability mass fitting (PMF). The PMF discretization is simple: the probability masses on regular intervals are computed and aggregated on a single value in the corresponding interval. PMF has the advantage to be simple, refinable, and to conserve the shape of the distribution. Moreover, we show that it does not require more phases, and thus more computational effort, than co...

Document type: Thèse (Dissertation)

Référence bibliographique
Tancrez, Jean-Sébastien. Modelling queueing networks with blocking using probability mass fitting. Prom. : Semal, Pierre ; Chevalier, Philippe
Modelling Queueing Networks with Blocking using Probability Mass Fitting

Thesis submitted in partial fulfillment of the requirements for the degree of Philosophiæ Doctor in Applied Science by

Thèse présentée en vue de l’obtention du grade de Docteur en Sciences Appliquées par

JEAN-SÉBASTIEN TANCREZ

Promoteurs: P. CHEVALIER
             P. SEMAL
Jury:       V. BLONDEL (Président)
            B. MACQ
            J.-F. MACQ
            N. VANDAELE
            T. VAN WOENSEL
Acknowledgements

My Ph.D. studies were a period full of learning and joy. I would like to thank many people for this, and for helping me achieve this long-term work.

Firstly, I would like to thank my supervisors Philippe Chevalier and Pierre Semal for their support and their guidance. All this would not exist without their help. Their expertise and their smartness taught me a lot and were essential in the final success of my research. Their kindness and humanity also contributed to make my work very pleasant.

I would also like to thank the other members of the jury. I want to thank Benoît Macq for giving me the opportunity to enter the academic world, for following my progress as a researcher and for showing me much confidence. Jean-François Macq, Nico Vandaele and Tom Van Woensel also followed my progress throughout my Ph.D. research. I want to thank them for their acute advises. They brought a significant improvement to my work. Their interest in my research has been a real source of motivation to me. Finally, I would like to thank Vincent Blondel for accepting to be the president of the jury, and for doing it with his well-known qualities.

Throughout my Ph.D. studies, other professors also helped me with their advises and shared their expertise with me. I want to thank Fouad Riane from FUCaM for his unfailing support, Neil Hurley and Guénolé Sylvestre for their warm welcome in UCD Dublin, and Laoucine Kerbache from HEC Paris for his interest in my work and the interesting discussions.

During four years, my work has been funded by the European Social Fund and the Walloon Region of Belgium, as a research project “First Europe Objectif 1” (No. 415773). In this context, I collaborated with the industry ArcelorMittal Fontaine. This collaboration helped me to keep in touch with the reality of a practical application in a manufacturing plant. I really appreciated the help, the kindness and the availability of Florin Rotaru, Jean-Pascal Leloire and Vincent Galand from ArcelorMittal Fontaine. David Duvivier also helped me a lot in this collaboration. I want to thank him for his constant good will and his advises.

During these four years, I had the chance to be part of several research groups, and to share many happy moments with very nice colleagues. CORE is like a small world concentrated in one building. I met many kind and interesting people. I enjoyed the very good company of Géraldine (the caring metronome), Matthias (the kind questioner), Jean-Christophe (the curious player), Rafael (the child in an athletic body), Santanu (the child who became a Professor), Joël (tipoulé), Ksenia (the smile generator), Robert (la force tranquille), François (the figure dropper),
Deniz (the generous host), Paolo, Anton, Peter, Oliviers, Nicolas, Joachim, Ga-their, Alexis, Maia, Salome, Martin, as well as the Professors Per, Laurence and Français and the administrative staff Catherine, Maddy, Laurent, Sylvie and Fabienne. I also want to thank colleagues from LSM: Jean-Charles, Xavier, Quentin, Roman, Yves and Youssef. Besides UCL, I really enjoyed to be researcher in FUCaM too. FUCaM is like a small family in one building. I truly appreciate this atmosphere and the kindness of Benoit (the hairy heart), Valérie (the funny mum), Jean-Philippe I, Jean-Philippe II, Arnaud, Hongying, David, Olivier, Christine, Guillaume, Florence, Sabine, Karine, Marie, Emerence, and the Professors Fouad, Nadine and Bart. Moreover, I also spend six wonderful months in UCD Dublin. I really enjoyed the walks in the drizzle with Kevin, Mark, Liz, Pedro, Alan and Félix. Furthermore, I would like to thank Peter Broderick, Sam Amidon, Phil Elverum, Bret Easton Ellis and Charles Neuhaus for their unintentional but helpful contribution.

Finally, I would like to thank my parents, my beloved sisters and my friends for the support and the happiness. Special thanks also go to Marion for the many reviving breaths at the end of the thesis.
Abstract

In this thesis, we are interested in the **modelling of queueing networks with finite buffers and with general service time distributions**. Queueing networks models arise in many application fields (manufacturing, communication and computer networks, traffic flow, healthcare, etc.) and have proven to be very useful tools to evaluate the performance of complex systems. In order to analyse queueing networks with general service time distributions, the distributions are most often transformed to tractable distributions so that the Markov theory can then be applied. In practice, the modelling process of a queueing network with general distributions can thus be seen as made of three steps: data collection, distribution fitting and analytical modelling.

Our main originality lies in the second step, i.e. in the way we build tractable service time distributions. We propose to discretize the distributions by **probability mass fitting** (PMF). The PMF discretization is simple: the probability masses on regular intervals are computed and aggregated on a single value in the corresponding interval, in order to get a discrete distribution. By definition, and it is one of the main motivations of it, PMF has the advantage to conserve the shape of the original distribution. Moreover, PMF is refinable (by reducing the intervals width), and bounding (as the discretized value allows to bound the original one). The PMF requires distributions with a finite support, but this is not restrictive in most applications. In our research, we define probability mass fitting, present its properties, and study the global modelling method coupling PMF with an analytical model (state model or decomposition).

To begin, from the distributions transformed by PMF, the evolution of the system is then modelled by a discrete Markov chain, and the performance of the system is evaluated from the chain. In other words, we exactly analyse the modified system using a state model. First, it allows to compute **bounds** on the throughput of fork-join queueing networks, and of split systems. Bounds are the only exact information available for queueing networks with general service time distributions, and few such bounds have been proposed in the literature. We propose two bounding methodologies. The first methodology extends the bounding property of the probability mass fitting. From the bounds on the service time, we deduce upper and lower, refinable, bounds on the throughput. The second bounding methodology offers a tight lower bound on the cycle time. The critical path, i.e. the sequence of jobs which covers the running time without gap and without overlap, can be computed in the discretized time. The same sequence of jobs is
non-overlapping in the original time, and thus leads to a bound. The tightness of this bound has been shown by numerical experiments (relative error on the order of one percent).

Second, probability mass fitting leads to accurate approximation of the performance measures. For fork-join queueing networks, the relative errors are in the order of a few tenths of a percent. Moreover, the accuracy level is remarkably stable in the system configuration: topology, number of stations, buffers sizes and service time distributions. For split-and-merge queueing networks, the method behaves less nicely but is still accurate. Other performance measures, such as the work-in-progress and the flow time, can also be accurately evaluated. As the PMF approximation of the distributions is refinable, the accuracy of the estimations can be improved, according to the computational effort one can afford. Together with the bounds, the approximations allow to get a good grasp on the exact measure with certainty.

Third, the shape conservation of the probability mass fitting (on the service time distributions) can also be extended. The cycle time distribution can be computed in the discretized time and shows to be a good approximation of the original cycle time distribution. The distribution provides more information on the behavior of the system, compared to the isolated expectation. Percentiles, for example, help the manager to evaluate the risk and the service level of the system.

However, when the transformed system is exactly analysed, by a state model, the computational cost quickly increases with the system size. This is the main limitation of this global modelling method. It comes from the state model and it is independent of the probability mass fitting. Indeed, the number of values in the distributions discretized by PMF is reasonable, compared to the one obtained with concurrent methods (such as moments fitting). In order to decrease the computational cost, the decomposition technique can be applied after PMF, instead of a state model. The decomposition is approximate. The accuracy of the global modelling is thus less good, and the bounds are lost. However, decomposition allows to analyse larger systems much quicker, and with good accuracy (relative error on the order of one percent, and less with buffer sizes larger than two). Moreover, PMF seems to bring an improvement in the application of the decomposition technique. Indeed, PMF allows a fine approximation of the blocking and starving time distributions, which are crucial in decomposition. Furthermore, even if less accurately, decomposition (with PMF) still allows to approximate the cycle time distributions.

In conclusion, we believe that probability mass fitting can be considered as a valuable alternative in order to build tractable distributions for the analytical modelling of queueing networks. It does not require more phases, and thus more computational effort, than concurrent methods and offers bounds and accurate estimations of the performance measures and of their distributions. Moreover, PMF has the advantage to be simple, intuitive, and refinable.
Résumé

Dans cette thèse, nous nous intéressons à la modélisation des réseaux de files d’attente à stocks finis et dont les distributions des temps de service sont générales. Les modèles de réseaux de files d’attentes sont très utiles pour évaluer la performance de systèmes complexes, dans de nombreux domaines d’application (industrie, réseaux de communication ou d’ordinateurs, trafic routier, gestion hospitalière, etc.). Cependant, afin d’analyser des réseaux de files d’attente dont les distributions sont générales, les distributions doivent d’abord être transformée en distributions utilisables de sorte que la théorie de Markov puisse être appliquée. En pratique, le processus de modélisation d’un réseau de files d’attente à distributions générales est donc composé de trois étapes: la récolte de données, la transformation des distributions et la modélisation analytique.

Notre principale originalité se situe dans la deuxième étape, dans la façon dont nous construisons des distributions de temps de service utilisables. Nous proposons de discrétiser les distributions par ajustement sur les masses de probabilité (probability mass fitting, PMF). La discrétisation PMF est simple: les masses de probabilité sur des intervalles réguliers sont calculées puis agrégées sur une valeur unique dans l’intervalle correspondant, de sorte à obtenir une distribution discrète. Par définition, et c’est une de ces motivations principales, le PMF a l’avantage de conserver la forme de la distribution. De plus, le PMF est affinable (en réduisant la taille des intervalles) et bornant (car la valeur discrétisée permet de borner la valeur originale). Le PMF doit être opéré sur des distributions à support fini, mais ceci n’est pas restrictif dans la majorité des applications. Dans notre recherche, nous définissons l’ajustement sur les masses de probabilités, présentons ces propriétés, et étudions la méthode de modélisation globale couplant le PMF et un modèle analytique (un modèle d’états, ou la décomposition).

bornes inférieures et supérieures, affinables, sur le débit. La seconde méthodologie offre une borne inférieure, précise, sur la durée de cycle. Le chemin critique, c'est-à-dire la séquence de travaux qui couvre le temps sans écart et sans chevauchement, peut être calculé dans le temps discrétisé. Or, la même séquence de travaux est sans chevauchement dans le temps original, et même donc à une borne. La précision de cette borne a été montrée par des expériences numériques (erreur relative de l'ordre d'un pour cent).

Deuxièmement, l'ajustement sur les masses de probabilité permet une approximation précise des mesures de performance. Pour les réseaux d'assemblage/désassemblage, les erreurs relatives s'expriment en quelques dixièmes de pour cent. De plus, le niveau de précision est remarquablement stable quand la configuration du système change, que ce soit la topologie, le nombre de serveurs, la taille des buffers ou les distributions des temps de service. Pour les réseaux routés (séparation et rejoignement), la méthode se comporte un peu moins bien mais reste précise. D'autre mesures de performance, comme l'encours et le temps de circulation, peuvent aussi être évaluées précisément. Comme l'approximation PMF est affinable, la précision des estimations peut être améliorée, en fonction du temps de calcul que l'utilisateur est prêt à accepter. Avec les bornes, les approximations permettent d'avoir une bonne idée de la mesure exacte, avec certitude.

Troisièmement, la conservation de la forme des distributions (des temps de service), caractéristique de l'ajustement sur les masses de probabilités, peut aussi être étendue. La 

Distribution de la durée de cycle peut être calculée dans le temps discrétisé et est une bonne approximation de la distribution originale. La distribution fournit plus d'information sur le comportement du système, par rapport à l'espérance. Les quantiles, par exemple, aident le gestionnaire à évaluer le risque et le taux de service du système.

Cependant, quand le système transformé est analysé exactement, par un modèle d'état, le temps de calcul augmente rapidement avec la taille du système. C'est la principale limitation de cette méthode de modélisation globale. Elle vient du modèle d'états et est indépendante de l'ajustement sur les masses de probabilité. En effet, le nombre de valeurs dans les distributions discrétisées par PMF est raisonnable, comparé à celui obtenu avec des méthodes concurrentes (comme l'ajustement sur les moments). Afin de diminuer le temps de calcul, la technique de décomposition peut être appliquée suite au PMF, à la place d'un modèle d'état. La décomposition est approximative. La précision de la modélisation globale est donc moins bonne, et les bornes sont perdues. Cependant, la décomposition permet d'analyser des systèmes plus larges nettement plus rapidement, et avec une bonne précision (erreur relative de l'ordre d'un pour cent, et moins pour des buffers de tailles plus grandes que deux). De plus, le PMF semble apporter une amélioration dans l'application de la technique de décomposition. En effet, le PMF permet une approximation fine des distributions du temps de blocage et de famine, qui sont cruciales pour la décomposition. En outre, la décomposition (avec le PMF) permet encore d'approcher la distribution de la durée de cycle, même si cette approximation est moins précise.

En conclusion, nous pensons que l'ajustement sur les masses de probabilité peut être considéré comme une alternative intéressante pour construire des distributions utilisables pour la modélisation analytique des réseaux de files d'attente.
Il ne requiert pas plus de phases, et donc pas plus de temps de calcul, que ces concurrentes et offre des bornes et une estimation précise des mesures de performance et de leur distribution. De plus, le PMF a l’avantage d’être simple, intuitif, et affinable.
List of Publications

Historically, our work began with the particular PMF called “grouping at the end”. We first studied its properties, developed the modelling method, and found bounds and approximations for this particular PMF. These results were first presented in April 2006, in Rabat, Morocco, at the conference MOSIM’06 : 6ème Conférence Francophone de MOdélisation et SIMulation [80]. We then refined and presented them in June 2006, in Charleston, USA, at the conference MAM 2006: Markov Anniversary Meeting [79] (also published as Working Paper UCL-IAG 0607). An extended version of this work (notably adding the critical path computation and a comparison with moments fitting) is forthcoming in European Journal of Operational Research [85]. This research, focusing on grouping at the end, has been presented in two seminars: the Third CEMS SCM Seminar, in January 2006, in Riezlern, Austria; and the Workshop in Logistics and Supply Chain Management, in March 2007, in Louvain-la-Neuve, Belgium. Moreover, we showed that the method could compute approximations of the cycle time distribution and bounds on other performance measures, in a paper presented in Paris, France, in April 2008, at the conference MOSIM’08: 7ème Conférence Francophone de MOdélisation et SIMulation [83].

Subsequently, the second, linked, particular PMF, called “grouping at the beginning” has been briefly studied (properties, modelling method, bounds and approximations) and compared to grouping at the end. It has been the subject of a conference paper, presented in June 2007, in Beijing, China, at IESM’2007: International Conference on Industrial Engineering and Systems Management [81]. Moreover, we presented it in the Fourth CEMS SCM Seminar, in Riezlern, Austria, in February 2007.

The extension to the general definition of probability mass fitting was then quite natural. We extensively present the general PMF, and apply it to the modelling of tandem queues, in a paper which is forthcoming in Annals of Operations Research [86] (also published as CORE Discussion Paper 2008/28). The application of the general PMF to more complex queueing networks (fork-join and split-and-merge) has been presented in four seminars: the Fifth CEMS SCM Seminar, in Riezlern, Austria, in January 2008; the Meeting of the Euro Working Group on Stochastic Modeling, in Istanbul, Turkey, in June 2008; the INFORMS Annual Meeting, in Washington D.C., USA, in October 2008; and the ORBEL Annual Meeting, in Leuven, Belgium, in February 2009. In this general setting (general
PMF, queueing networks), a particularly interesting result, the computation of a tight lower bound on the cycle time, was the subject of the CORE Discussion Paper 2008/63 [82] (submitted to a journal in November 2008).

Finally, note that, besides our main subject (modelling queueing networks with blocking using probability mass fitting), we applied stochastic modelling in the field of health care management. We proposed a model to help operating theater managers dealing with the stochasticity of their environment (operating times, emergencies). However, this work is not related in this thesis. It has been presented in September 2008, in Lausanne, Switzerland at the conference GISEH 2008 : Gestion et Ingénierie des Systèmes Hospitaliers [84], and an extension (including the blocking caused by recovery beds) will be presented in May 2009, in Montréal, Canada, at the conference IESM’09: International Conference on Industrial Engineering and Systems Management [73]. This work will also lead to the publication of a chapter of the book Intelligent Patient Management [87] (also published as CORE Discussion Paper 2008/82).

International Journal Papers


Book Chapters


Refereed International Conference Proceedings


Contents

1 Introduction 1
   1.1 Illustrating Example ........................................ 2
   1.2 Queueing Networks with Blocking .............................. 6
      1.2.1 Definition, assumptions and notations ..................... 6
      1.2.2 Topologies ............................................. 9
      1.2.3 Application Fields ..................................... 12
   1.3 Steps of the Modelling Process ............................... 14
      1.3.1 Data Collection ......................................... 15
      1.3.2 Distribution Fitting .................................... 15
      1.3.3 Analytical Modelling .................................... 18
      1.3.4 Bounds on the Performance Measures ...................... 21
   1.4 Outline of the Thesis ........................................ 22

2 Probability Mass Fitting ........................................ 25
   2.1 Particular PMF ................................................ 25
   2.2 Probability Mass Fitting with Instant Jobs .................. 27
      2.2.1 Definition ............................................. 27
      2.2.2 Properties ............................................. 29
   2.3 PMF without Instant Job ..................................... 32
      2.3.1 Definition ............................................. 33
      2.3.2 Properties ............................................. 34
   2.4 Strengths and Weaknesses .................................... 36

3 Modelling Method ................................................. 41
   3.1 Tandem Queues ................................................ 41
      3.1.1 PMF Without Instant Job ................................ 41
      3.1.2 PMF With Instant Jobs ................................... 49
   3.2 Fork-Join QN .................................................. 52
   3.3 Split-and-Merge QN ........................................... 55

4 Bounds ............................................................. 59
   4.1 Critical Path ................................................. 59
      4.1.1 Synchronization Constraints ............................... 60
      4.1.2 Definition and Construction ............................... 63
      4.1.3 Properties ............................................... 64
      4.1.4 Computation ............................................. 67
## 4.2 Bounds from the Discretized System

4.2.1 Bounding Methodology ........................................ 69
4.2.2 Application of the Methodology with PMF .................... 73

## 4.3 Bound from the Critical Path Computation

4.3.1 Bounding Methodology ........................................ 80
4.3.2 Computational Experiments .................................... 83

## 4.4 Split-and-Merge QN

## 4.5 Other Performance Measures

## 4.6 Monotonicity

## 5 Performance Evaluation

5.1 Cycle Time Evaluation ............................................ 93
5.1.1 Simple Approximations ........................................ 94
5.1.2 Approximation from the Critical Path Computation ........... 102
5.1.3 Managerial Example ............................................ 105

5.2 Other Performance Measures ...................................... 106

## 5.3 Cycle Time Distributions ...................................... 110
5.3.1 Computation .................................................. 110
5.3.2 Shape Conservation ........................................... 112
5.3.3 Examples ..................................................... 113
5.3.4 Managerial Example ........................................... 115
5.4 Comparison with Other Distribution Fittings .................... 118

## 6 Decomposition

6.1 Modelling Method ................................................ 123
6.1.1 Tandem Queues ............................................... 124
6.1.2 Fork-Join Queueing Networks ................................ 126
6.1.3 Complexity .................................................. 128

6.2 Computational Experiments ..................................... 130
6.2.1 Comparison on Small Systems ................................ 130
6.2.2 Larger Fork-Join QN ......................................... 131
6.2.3 Cycle time Distributions ..................................... 133

## 7 Conclusions

7.1 Results Achieved ................................................ 137
7.2 Future Prospects .................................................. 142
Chapter 1

Introduction

In this thesis, we address a classical and general problem: the modelling of queueing networks with blocking, and with general service time distributions. A queueing network is a set of interconnected waiting lines. In his foundational work, more than fifty years ago, Jackson [44, 45] defined queueing networks which could be analyzed quite easily, as each queue could in fact be studied independently. The assumptions allowing this result to be proved are the following: Poisson arrival process and exponentially distributed service times, infinite storage spaces, independent routing, and stability (the utilization of the queues is smaller than one). However, in practice, these assumptions reveal to be restrictive and unrealistic.

The fundamental result of Jackson, as well as its limitations, gave rise to a considerable amount of research in queueing theory, and much progress have been made in the analysis of queueing networks in the last fifty years. Queueing networks have shown to be of broad practical interest. They arise in many application fields (manufacturing, communication and computer networks, traffic flow, healthcare, etc.) and have proven to be very useful tools to evaluate the performance of complex systems.

In this work, our goal is to propose and study new ideas to analyse queueing networks for which the two first assumptions of Jackson are relaxed (first, Poisson and exponential distributions, and second, finite storage spaces). First, we assume the service times to be generally distributed (concerning the arrival process, we suppose infinite supply in front of the entry stations). In practice, it is clear that the service time is not always exponentially distributed. In manufacturing for example, the squared coefficient of variation of the service time is most often smaller than one (see Bitran and Dasu [14]). However, queueing networks with general service time distributions cannot be analysed directly. The distributions have first to be transformed to tractable distributions, so that the Markov theory can then be applied. Our main originality lies in this preliminary step, i.e. in the way we build tractable distributions. We propose a new method to build discrete phase-type distributions, called probability mass fitting (PMF), and study the global modelling process using it.
Second, we relax the assumption of Jackson concerning the infinite capacity: we suppose finite capacity queues. Such queueing network models have been introduced to represent real systems with finite capacity resources. They can model telecommunication or computer systems with limited shared resources, production systems with finite storage buffers, or traffic flow with congestion, for example. In manufacturing in particular, limiting the buffer spaces has good implications. It notably allows to limit the work-in-progress (number of units in the system), which reflects the working capital requirement and the ability to quickly react. The willingness to limit the storage space gave rise to pull systems and kanban systems for example. The finiteness of the buffer has a major implication: when a buffer is full, the flow entering the queue is blocked. This phenomenon is called blocking and gives rise to the denomination “queueing networks with blocking”.

In this chapter, we introduce queueing networks with blocking, discuss their modelling and review the existing methods. To begin, in Section 1.1, we illustrate queueing networks modelling and its application with a simple example. In Section 1.2, queueing networks with blocking are defined, various configurations are presented, and applications are described. Then, in Section 1.3, we review the steps of the modelling of queueing networks with general service time distributions, and the existing analytical models. Finally, in Section 1.4, we give the outline of the thesis.

1.1 Illustrating Example

In this section, we illustrate queueing network modelling and its utilization on a very simple example, in the context of manufacturing. The aim is to give intuition on the reason why the methodology and the results presented in this text can be useful, and in which context. Production is one of the main application field of queueing network models, and it is the application we have in mind in priority. The modelling of a manufacturing system firstly aims to understand how the system behaves and evolves, and to estimate measures of its performance. The performance evaluation is crucial for the management of a production system. It allows to take design as well as operational decisions with a good knowledge of their impacts. In this context, queueing network models are a natural approach, as the queueing phenomenon is common in production systems. Moreover, the blocking phenomenon, caused by a limited storage space, is also common and it is well analysed thanks to queueing networks with finite buffers.

Let us suppose an analyst is asked to study a production line with two stations and a buffer where space is available for only one unit. Raw material is always available in the beginning of the system, and there is always some room at the end of the line for the finished products. Small buffer sizes can happen in various industries. It may happen in the steel industry for example: the product cannot be stored for a long time, as it would then cool down (this implies that the storage space should be limited). Hot metal needs to be stored in refracting containers that are expensive and must be maintained at temperature of 1500 degrees Celsius. Another example is the pharmacological industry: as the entire process needs to be sterile, storage is very expensive and has to be limited. Moreover, more broadly
1.1. ILLUSTRATING EXAMPLE

Figure 1.1: Two station production line with a buffer of size one and with exponential service times (left), and Markov process modelling its evolution (right).

Speaking, limiting the work-in-progress, i.e. the number of units in the system, is an objective for the good management of manufacturing systems, in order to limit the working capital and to improve the responsiveness of the process.

To get a first idea about the behavior of the two station line (see Figure 1.1), the analyst may conduct a simple analysis based on a Markov process. The latter implies exponential service time distributions. The analyst thus simply collects the average number of units served per unit of time by each station. The rate of the first (resp. second) station is measured to be \( \lambda \) (resp. \( \mu \)) jobs per hour. For modelling purposes, the service times are supposed to be exponentially distributed (while they are obviously not, in general), with these rates. A Markov process can then be built to model the evolution of the production line. It includes four states and is depicted on Figure 1.1. Each state lists the state of the first station, the state of the buffer, and the state of the second station. Suppose the line begins in state \( W_0W \), i.e. with both stations working and an empty buffer. When the first station ends its job (rate \( \lambda \)), it puts the item in the buffer and begins a new job, leading to \( W_1W \). On the contrary, when the second station ends first (from \( W_0W \), rate \( \mu \)), it puts the finished product in the storage at the end of the line, and finds no new item to work on. In this case, the station has to wait (one say it is starved) leading to the state \( W_0S \). The second station unstarves when the first station ends its job (rate \( \mu \)) and passes an item to the second station, leading back to \( W_0W \). When the first station ends before the second one in state \( W_1W \), the first station is unable to get rid of its job, as the buffer is full. In this case, one says the station is blocked (leading to \( B_1W \)): it keeps the item and cannot begin to work on a new one. Finally, the analyst gets a simple Markov process which gives him a first idea of the stochastic evolution of the system.

From the Markov process, the analyst can estimate various performance measures, i.e. measures reflecting the behaviour of the system. For this, he computes the stationary probabilities of each state. Informally speaking, the stationary probability of a state gives the proportion of time during which the system is in this state. The stationary probabilities \( \pi_i \) of the states of a continuous finite Markov process can easily be computed with the classical formula \( \pi Q = 0 \), with \( \sum \pi_i = 1 \) and where \( Q \) is the transition matrix of the Markov process. Let us suppose the first station of the production line is able to accomplish 10 jobs per hour, while the second station does 8 jobs per hour (\( \lambda = 10 \) and \( \mu = 8 \)). The analyst can easily compute the stationary probabilities:

\[
\begin{align*}
\pi_{WOS} & = 0.17, & \pi_{W0W} & = 0.22, & \pi_{W1W} & = 0.27, & \pi_{B1W} & = 0.34.
\end{align*}
\]

\footnote{By the memoryless property of the exponential distribution, the distribution of the remaining service time in the second station is still exponential.}
CHAPTER 1. INTRODUCTION

From this, it is straightforward that the second station is idle around 17% of the time and the first station around 34% of the time. The fact that the first station is more often idle is not surprising as this station is faster. Other performance measures can be evaluated. The throughput, i.e., the number of products leaving the system per unit of time, reflects the productivity of the line and is clearly one of the most important measures. It is given by the production rate of the last station times the probability of the station to be working: $8 \cdot (1 - 0.17) = 6.6$ products/hour on average. Note that the throughput of the first station is the same: $(1 - 0.34) \cdot 10 = 6.6$. The work-in-progress is also an important performance measure, as argued earlier. It can simply be computed as the product of the work-in-progress of each state weighted by their probabilities. It thus equals $0.17 \cdot 1 + 0.22 \cdot 2 + 0.61 \cdot 3 = 2.4$ units in the system on average. By Little’s law, it can then be deduced that an item spends $6.6/2.4 = 2.7$ hours in the system, on average (this time is called the flow time). Other stationary performance measures could be computed, as well as transient performance measures (the time to output twenty products for example).

From these performance measures, the analyst gets a good knowledge of the behavior of the production line, what is essential for the management of the system. He may also study the impact of modifications brought to the line. He could for example analyse the opportunity to increase the buffer capacity. In case the buffer capacity is brought to two, the new Markov process would be very similar to the previous one. Two states, $W^2W$ and $B^2W$, appear and the state $B^1W$ is removed. The storage enlargement improves the productivity (the throughput increases from 6.6 to 7 products per hour) but has a negative impact on the work-in-progress (from 2.4 to 3.1 units in the system). The manager of the production line thus gets insights on what would be the impact of such a decision. He can make a choice from the benefit brought by the productivity improvement and the cost of the work-in-progress increase.

Getting a good knowledge of the system, and of the impact of the decisions, is of course essential for a good management of a production system. Very large investments may be involved and decisions have to be made with great care, based on accurate analysis. In this sense, the previous analysis is most probably insufficient. It is based on the exponential distribution assumption. This assumption is very restrictive and the quality of the approximation obviously suffers from it. It only uses the mean of the processing time distributions and forgets everything else, about the variance of the distribution and about its shape. Moreover, the exponential distribution has an infinite support, what is quite unrealistic in practice. In real applications, the support of the service time distributions is always finite, i.e., an infinite processing time for a job is impossible. All in all, the exponential assumption is very strong and affects the subsequent analysis of the system: it may significantly affect the performance evaluation and thus the decisions made from it.

To get a better approximation of the real service time distribution, while still be able to apply the Markov theory, the phase-type distributions have been introduced (see Neuts [64]). A phase-type distribution (PH) is a distribution which can be modelled as the time until absorption of a Markov chain with one absorbing state. The simplest PH distribution is the exponential distribution: it can be modelled as
1.1. ILLUSTRATING EXAMPLE

Figure 1.2: Erlang(2, λ) distribution compared to the exponential(λ/2) distribution (left-hand side). Markov process modelling the evolution of a two station line with no storage space and Erlang(2, λ) (first station) and Expo(µ) (second station) service time distributions (right-hand side).

A Markov process with just one state linked to the absorbing state (with transition rate equal to the rate of the exponential distribution). Other examples are the Erlang (states in series), or the hyperexponential distributions (states in parallel).

In order for the analyst to improve its modelling of the production line, a classical method is to infer not only the mean but also the variance from the data he has about the service time distributions of the stations. From these two parameters (instead of one), the analyst can fit a phase-type distribution that has the same two first moments, and he then gets a better approximation of the real system. Using PH distributions, the production line can also be modelled by a Markov process. Using an Erlang distribution with two phases for example, a working state has to be decomposed in two states, i.e. working in first stage ($W_1$) and in second stage ($W_2$). In Figure 1.2 (left), we show the Erlang(2, λ) distribution. Compared to the exponential distribution, it can be seen that the Erlang distribution is less variable. On the right-hand side of Figure 1.2, the modelling of a system implying Erlang distributions is illustrated on a simple example. The two phases of a working state ($W_1$ and $W_2$) complicate the Markov process.

From these examples, we understand that the modelling done by an analyst to help in the management of manufacturing systems (or, more broadly speaking, queueing networks) can be seen as made of three steps: data collection, distribution fitting (often to PH distributions), and analytical modelling (often using the Markov theory). Various methodologies have been proposed to fit distributions and to analytically model networks. They can be more or less accurate and complex. In this context, the main originality of our work is to propose a new alternative in order to build phase-type distributions.

The distribution fitting method we propose builds discrete phase-type distributions. The basic idea is simply to conserve the probability masses, and thus the shape of the distributions. The probability masses are computed and aggregated on discrete values to form a discrete PH distribution. The approach is called probability mass fitting (PMF) and is illustrated on Figure 1.3. In this example, the distribution is collected in the form of a histogram. Then, the probability masses
are aggregated on discrete values, to form a discrete PH distribution. To refine the approximation, the number of discrete values can be increased (also leading to a larger Markov chain). The motivation behind this new distribution fitting method is to propose a more detailed fitting of service time distributions, which stick to the shape, rather than to the two first moments for example. Moreover, it can be refined to any desired accuracy, as far as one can afford the consequent computational cost. Furthermore, the proposed approach allows to fit finite support distributions, what is more realistic in practice.

With discrete PH distributions, the evolution of the system is then modelled by a discrete Markov chain. In this thesis, we define and study the distribution fitting method in details and analyse the global modelling of queueing networks with blocking using PMF. Of course, more complex networks are considered than in the illustrating examples given in the present section.

### 1.2 Queueing Networks with Blocking

In this section, we present queueing networks with blocking in details. In particular, we define the networks we focus on, and the assumptions we make.

#### 1.2.1 Definition, assumptions and notations

A queueing network is a set of interconnected queues where the departures from one (or more) queue enter one (or more) other queue, according to a specified configuration, or leave the system. Many queueing networks have been studied in the last fifty years. Many configurations can be thought of, and many options can be switched on or off.

As already said, in this work, we focus on queueing networks with finite capacity, i.e. for which the waiting spaces before the queues (called buffers) are limited. They are used to represent real systems with finite capacity resources in various application fields. The function of the buffers is to decouple the stations, i.e. to decrease their mutual dependence in the network. However, in practice, the capacity is often limited by physical constraints, causing coupling between the buffers. These constraints can come, for example, from finite storage space in production systems, or from limited shared resources, such as interconnecting links, in computer systems. The coupling between the stations in the network appears in two phenomenons: starving and blocking (which is caused by the limited buffer capacities). First, a station is said to be starved when it cannot begin to serve a
new unit because a previous buffer is empty. Second, a station is said to be blocked when it cannot get rid of an item because a next buffer is full. Obviously, increasing the buffer sizes allows to limit these phenomena, to decrease the coupling, and to reduce the efficiency losses due to them.

We now state the assumptions we make for the networks we study. Most of them are not restrictive. Note that, by default, they are taken with the manufacturing application in mind. We first list the assumptions concerning the single queues composing the network (according to Kendall’s notation).

**Arrival.** The arrivals coming from other queues are generally distributed. We suppose infinite arrival from outside the network, i.e. the network is saturated (see “Saturation” below).

**Service.** The service time distributions are assumed to be general, i.e. it can be any distribution. However, the support of the distribution has to be finite, i.e. the largest possible service time is not infinite. Successive service times are independent and identically distributed. The finite support assumption is not restrictive since it is always the case in practice (particularly in manufacturing applications).

**Number of servers.** The number of servers per station is fixed to one in this thesis. Throughout the text, we thus use the term “station” while “node” of the network could also be used. We did not consider multi-server queues. This assumption can be restrictive (but the extension to multi-server queues could be considered).

**Capacity.** The queue capacity is supposed to be finite, i.e. we study queueing networks with blocking. Again, the finiteness assumption is not restrictive since it is always the case in practice (particularly in manufacturing applications).

**Service discipline.** The service discipline is supposed to be first come first served (FCFS), i.e. the customers are served in the order which they arrived. Other disciplines could be considered. FCFS is the most classical assumption.

In other words, the single queues making up the networks are $GI/G/1/b_i/FCFS$ queues (where $b_i$ is the capacity of the $i^{th}$ buffer). Other assumptions, concerning the network notably, are the following:

**Open.** We study open queueing networks, as opposed to closed queueing networks. Customers arrive in the system at some “in” stations (sources) and leave it from some “out” stations (sinks). On the opposite, in a closed network, a fixed number of customers move in the network, endlessly (see Onvural [66]). Moreover, the networks we analyze are supposed to be acyclic, i.e. without loops. A customer is never served twice by the same station. This is another possibly restrictive assumption. The extension to closed queueing networks is discussed in Section 7.2.

**Reliable.** The stations are supposed to be reliable, i.e. without failures. As we assume general service time distributions, this assumption can be considered
as not restrictive. Indeed, in case of unreliable stations, the time to failure and the repair time can be included artificially in the service time, leading to the so-called “completion time”, introduced by Gaver [34], or the “effective process time” according to Hopp and Spearman [43].

**Saturation.** The network is supposed to be saturated, i.e. there is infinite arrival and infinite demand. Stations getting customers from outside the network (sources) are thus never starved, and stations located at the end of the network (sinks) are never blocked. This assumption is the most classical in the literature. However, the equivalence with other networks (with arrival or demand processes) has been shown. Fictitious stations can be used to mimic the arrival or the demand process (see Dallery and Gershwin [26] for example).

**Asynchronous.** The system is supposed to be unsynchronized, i.e. we assume asynchronous part transfer. The stations are not constrained to start or stop their operations at the same time (as in synchronous systems). As long as the intermediate buffer is not full or empty, two consecutive stations do not start or end their job at the same instant. This assumption is true in most real systems, in manufacturing in particular. Moreover, we assume discrete parts (as opposed to continuous material). Individual parts are treated one after the other, and require a finite stochastic service at each station.

**Blocking policy.** Blocking after service (BAS), also called manufacturing blocking, is assumed. It means that, if the next station is full, a job is blocked in its current station, after having been served. This assumption is the most classical in the manufacturing related literature. Our method could easily be fitted to other blocking policies (blocking before service, repetitive service blocking, etc., see Balsamo et al. [13] or Dallery and Gershwin [26]). Moreover, equivalences with other blocking policies have been shown in the literature. For example, a system with a blocking before service policy can be studied using a model with a BAS policy by adapting the buffer size.

The assumptions we make on the queueing networks we study are now clearly stated. In the next subsection, we present in more details the three configurations we particularly have in mind in this work. Before that, let us introduce some notations.

- The total number of stations in the network is given by \( m \). The stations are numbered and denoted \( S_i \), with \( i = 1, 2, \ldots, m \).

- The set of stations which directly precede a station \( i \) is denoted \( E(i) \), while the set of stations which directly follow a station \( i \) is denoted \( F(i) \).

- The number of buffers in the network is denoted \( m_b \). The size of the buffer between stations \( i \) and \( j \) is denoted \( b(i, j) \) (stations capacity excluded). When a buffer is associated to the unique station \( j \) following it, as for tandem queues and split-and-merge networks, the index \( i \) is omitted and \( b_j \) is used for its size.
1.2. QUEUEING NETWORKS WITH BLOCKING

Figure 1.4: Tandem queue including four stations and three buffers.

- Each job is given an index \( k = 1, 2, \ldots \), where the first job to leave the system is given index 1. A job has the same index on each station. The job \( k \) on station \( i \) is denoted \( w_{i,k} \), and the random variable representing the service time of \( w_{i,k} \) is denoted \( l(w_{i,k}) \).

1.2.2 Topologies

Queueing networks may follow various topologies, i.e. various organizations. In the previous section, we detailed the assumptions we make: general service time distributions, mono servers queues, finite buffers, open networks, etc. In the following, we present the three configurations we will particularly focus on, i.e. on which our approach will be applied. However, note that many of the proposed ideas could be applied to other systems (see Section 7.2 for a discussion). Tandem queues, i.e queues in series, are the simplest configuration but are of practical interest and show interesting properties. Fork-join and split-and-merge queueing networks allow to model more complex systems. They correspond to systems that are classically observed in practice and studied in the scientific literature.

Tandem Queues

The simplest topology for a queueing network is the tandem queue. A tandem queue is made of several queues in series. The items are served sequentially by the stations and stored in the buffers between them when necessary. They visit each station and each buffer exactly once and in a fixed sequence order.

The particular structure of tandem queues (with finite buffers) is illustrated on Figure 1.4. It is a linear network of \( m \) stations \( (S_1, S_2, \ldots, S_m) \) separated by \( m - 1 \) buffers. The service of one customer consists in the sequential service by the stations \( S_1 \) to \( S_m \). The customer enters the system at station \( S_1 \) and leaves at \( S_m \). After its service by a station, let us say \( S_i \), the item is given directly to the next station if he is ready to serve it (i.e. starved), or, otherwise, the item is stored in the buffer preceding \( S_{i+1} \) (if a space is available, otherwise it waits in \( S_i \)). There it waits until the next station, \( S_{i+1} \), finishes its job on the previous item and gets rid of it. At this moment, the service of the item on station \( S_{i+1} \) starts. This is repeated until the item gets its last service at station \( S_m \) and then leaves the system.

As already said, an important phenomenon occurring in queueing networks with finite buffers is blocking. Due to the limited capacity, items can get blocked in a station due to a next buffer which is full. This tends to decrease the efficiency of the system, in terms of the number of customers served per unit of time. It also seriously complicates the modelling and the resolution of such networks, compared
to infinite buffer systems. Tandem queues are simplest queueing network configuration. However, even if simple, tandem queues are of practical interest. They are commonly encountered in real applications, to model manufacturing flow lines for example.

Fork-Join Queueing Networks

Fork-join queueing networks (FJQN) present a more complex configuration than tandem queues and allow to model more complex systems. In a FJQN, a station may have more than one preceding station and more than one successor. Such queueing systems are classical in the literature and arise in many applications areas. In manufacturing, they allow to model assembly/disassembly systems.

A fork-join queueing network is a queueing network in which the nodes are linked arbitrarily without forming loops. The stations can have several input or output stations, but each buffer has exactly one upstream station and one downstream station (see Dallery et al. [27]). There is one buffer between each pair of linked stations. When a service ends in a station, one job is taken from each upstream buffer and one job is put in each downstream buffer. An example of FJQN is given in Figure 1.5. In a fork station (e.g. $S_2$ in Figure 1.5), one item is taken from the previous buffer (except if it is empty), disassembled into several pieces, and put in the next buffers (except if they are full). In a join station (e.g. $S_6$ in Figure 1.5), items are taken from each previous buffers (except if they are empty), assembled into one unit, and the latter is put in the next buffer (except if it is full). In FJQN, each item goes through each station. In other words, all jobs $w_{i,k}$ have sense. Furthermore, there are some stations that have no input buffers. They are called sources, and are never starved (e.g. $S_1$ and $S_3$ in Figure 1.5). Some stations have no output buffers. They are called sinks, and are never blocked (e.g. $S_7$ and $S_8$). Note that the tandem queue is a particular case of FJQN.

Here, we suppose the unloading policy to be independent unloading (IU). From a fork station, jobs can be unloaded independently of each other into the downstream stations. The jobs do not have to move out of a fork station in a synchronized manner. In other words, when station $S_2$ ends a job, it puts, if possible, an
1.2. QUEUEING NETWORKS WITH BLOCKING

item in the second buffer (preceding $S_5$), even if the first buffer (preceding $S_4$) is full. However, if any succeeding buffer (preceding $S_4$, for example) is full when a fork station ($S_2$) ends a job, the fork station becomes blocked till he can get rid of all the items he disassembled. Moreover, the loading policy is assumed to be independent loading (IL). The jobs coming from the upstream buffers can be loaded onto a join station independently of each other. When it ends a job, a join station takes every item available from the previous buffers. If necessary, the join station then waits for items to come in the other, empty, buffers, i.e. the join station is starved until he got all the items he has to assemble. These unloading and loading policies are common assumptions, in the manufacturing field in particular (other policies, such as simultaneous loading for example, can be found in Dallery et al. [28]).

Split-and-Merge Queueing Networks

Another classical queueing network topology is the split-and-merge queueing network (SMQN). Like fork-join queueing networks, it allows stations to have more than one predecessor and more than one successor, but the routing policy among the stations is different. Split-and-merge queueing networks are classically studied in the literature, and have shown to be useful in practice (to model job-shops in manufacturing for example, see Bitran and Dasu [14] or Govil and Fu [39]).

In split-and-merge queueing networks, the nodes are linked arbitrarily without forming loops, i.e. the stations can have several input or output stations. Unlike in FJQN, one buffer is associated to each station, and a station exclusively takes items from this buffer. In split-and-merge queueing networks, items are not assembled or disassembled: one unit entering the system leads to exactly one unit leaving the system. In a split station, the item is routed to one of the following stations, according to some routing probabilities (see $S_2$ in Figure 1.6). In a merge station, the items are taken from the buffer, which gathers the items from every preceding station (see $S_6$ in Figure 1.6). Unlike in FJQN, in SMQN, some items may never pass through some stations (and some $w_{i,k}$ do not make sense). In other words, every job does not pass through every station. For example, in Figure 1.6, if an item goes from $S_2$ to $S_4$, it will never be served by $S_5$ and $S_7$. As for fork-join
CHAPTER 1. INTRODUCTION

queueing networks, stations without input are called sources and stations without output are called sinks. Note that the tandem queue is also a particular case of SMQN.

1.2.3 Application Fields

Queueing networks arise in many application fields. They have shown to be a powerful modelling tool, of broad practical interest. In the following, we discuss some of the main application fields.

Manufacturing

A first application field of queueing networks is the modelling of manufacturing or production systems. In this field, variability may have several origins. It may come from the characteristics of the process. Service times may be variable, because the job is done by a human for example. Failures may happen unpredictably and need repairs with random times. At a tactical/strategic level, the mix of the products that will be ordered is unknown, leading to stochastic service times. In these cases, the variability has a significant impact on the behavior of the system. The manager of a production system thus needs to take it into account, and evaluate its impact as accurately as possible.

In this context, queueing theory is a common approach. Queueing networks with blocking reveal to be a natural modelling tool, as the queueing and the blocking phenomena exist and are of particular importance in the manufacturing field. Queueing networks allow to evaluate the performance of the system. They aim to support the decisions of the managers by predicting the effect of various decisions, or possibly to build an optimization model based on the performance evaluation.

Models of production systems are needed at a strategic level, during the design phases, when the following elements have to be selected: the type of production equipments and their power, the layout of the different stations and the mechanisms of synchronization (equipment, transfer lot size, buffer size, ...). At a more operational level, models are also needed to provide support for daily production decisions like the load or the necessary staffing, and for customer oriented decisions like accepting a new job or promising some delivery time.

Moreover, inventory has become a crucial question in production, due to the investment it requires and ties up. In the present economic situation, companies are submitted to an ever more competitive environment. Investments have to bring ever higher profit margin. It is essential for companies to get high productivity, while limiting the investments used by stocks. In this context, the analysis of manufacturing systems with finite storage capacity is of practical interest. Models of such systems need to be realistic, accurate and complete in order to allow to design and manage them efficiently, and, in particular, to find the best balance between productivity benefits and inventory costs.

In this application field, tandem queues model manufacturing flow lines, also called production lines, fork-join networks model assembly/disassembly systems, and split-and-merge networks model job-shops. They allow to compute performance measures such as the productivity, the throughput, the work-in-progress,
the flow time, or the buffer occupancy for example. We refer the reader to Gershwin [36], Altink [5] and Hopp and Spearman [43] for books focusing on this application field.

**Computer and Communication Networks**

Queueing networks with blocking have also been extensively used to model computer networks or communication networks. In this field, the variability may come from the messages which are transmitted in the network. Their length, or the number of data packets, is a priori unknown and affects the service time, which is thus variable. Other origins of variability are failures in the system or external impacts (if some resources are shared with other systems or applications for example).

The modelling of computer and communication networks should thus take the stochastic nature of the system into account, in order to reliably help to design and manage them. Queueing network models are a natural approach in this matter. The finite capacity allows to take into account the limited memory space of a computer for example (the number of jobs that can wait for service by a CPU is bounded). Communication networks can be modelled using stations to model communication channels and buffers to model intermediate storage. The latter have limited capacity, and blocking may thus occur in the network, i.e. communication can be delayed because of a lack of space downstream.

Queueing network models allow to evaluate several performance measures of interest such as the queue length distribution, the utilization of the stations, the throughput of the network, or the response time. These measures aim to help managers and designers to take decisions with a good knowledge of their consequences. Examples of computer/communications systems which have been modelled using queueing networks with blocking are: disk input/output systems, multiprocessor architectures, computer grids, disk to tape backup systems or mass storage systems. The reader is referred to Balsamo et al. [13] for a book focusing on the computer/communication application field.

**Others**

Queueing network models have shown to be useful in many other applications fields such as traffic flow, healthcare, call centers, supply chains, service (supermarkets, administrative services), etc.

**Vehicle traffic** on roads can be modelled using split-and-merge queueing networks. Tandem queues (particular case of SMQN) are used to model basic freeway segments, split operations to model a segment splitting to several segments and merge operations to model segments merging into one. Finite buffers allow to model the finite capacity of the roads, and consequently to study the congestion phenomenon. We report the interested reader to Van Woensel [96] or to Van Woensel et al. [97].

Queueing network models have also been used in healthcare management. The patient flow in hospitals includes various stages, with service times and queueing, following many different possible routes. Finite buffers allow to model the finite capacity at some stages, and the consequent blocking (recovery beds after surgical operations for example). We report the interested reader to Green [40]
for a review paper and to Tancrez et al. [84, 87] for our contribution in this field, where we apply queueing theory to the performance evaluation of an operating theater.

1.3 Steps of the Modelling Process

As detailed in the previous section, our aim is to analyze queueing networks with blocking and with general service time distributions. In practice, when a real system is analyzed, the modelling process includes several stages. First, the analyst collects data about the system, about its configuration, its functioning, and about the service times in particular. Then, he transforms the information he collected in order to get a model which can be analyzed. Concerning the service times, he builds tractable service time distributions from the data. Finally, the analyst has two options to evaluate the performance of the system. He can use simulation or an analytical method. In our work, we choose the analytical way. In this case, phase-type type distributions (see Section 1.3.2 for a definition) are most often built previously to the application of the proper analytical method, in order to be able to apply the Markov theory on the modified system.

In this text, we thus see the modelling process of queueing networks with general service time distributions as split into three steps: data collection, distribution fitting, and analytical modelling. In the literature, the two first steps are often omitted. However, in practice, the modelling includes these three steps, and it is thus useful to study the modelling process as a whole. Notably, the study of the modelling error should include the errors made in the second step and not only in the third step, i.e. by the analytical modelling, as frequently done. It is important to note that the transformation of the original distributions unavoidably introduces an error in the subsequent performance analysis. According to Altiok [5], “there is always an error in this type of approximation [i.e. distribution approximation], and it is important to invest every effort to minimize it”. In other words, all the models of queueing networks with general service time distributions are approximate in practice, as they include the distribution fitting stage.

Our originality mainly lies in the second step of the modelling process. In our research, we propose a new alternative to build tractable distributions, called “probability mass fitting” (PMF), and study its implications on the global modelling process. PMF is quite intuitive: the probability masses on regular intervals are computed and aggregated on a single value in the corresponding interval, leading to a discrete (phase-type) distribution. After PMF, from the discrete phase-type distributions, an analytical model can be applied. It models the evolution of the system by a Markov chain, from which the performance measures can be computed. Moreover, our approach allows us to find bounds on the throughput of queueing networks.

In the following, we detail the three steps of the modelling process and document them giving a brief state of the art. Moreover, we also present a state of the art on bounding methodologies.
1.3. STEPS OF THE MODELLING PROCESS

1.3.1 Data Collection

The first step of the global modelling process is the data collection. In practice, most often, real service time distributions are not available. Measures are taken in order to fairly assess the service times but the original distributions cannot be drawn exactly. A good way to get a precise knowledge of the service time of a station is to build a histogram. When an analyst studies a queueing system (a manufacturing system in particular), the information he has about the service times is often in this form. A histogram is the most classical data form in practice for service time distributions. Another alternative is to use kernel based estimation (a histogram is a possible output in this case).

From its informal presentation, it can be seen that probability mass fitting is well suited to histograms. Indeed, PMF computes the probability masses on regular intervals and aggregates them on a single value in the corresponding interval. The intervals can thus be chosen to correspond to those of the histograms, and, in this case, PMF simply aggregates the area of one bar on a single value, to get a discrete distribution. Doing so, no information is lost.

The method presented in this text can be applied supposing histograms as well as theoretical distributions to represent the service time distributions. If the original distribution is supposed to be exactly known, the method can be applied simply by integrating the distribution function on steps of the desired length (to get the probability masses) instead of the histogram. The probability masses can then be carried to a particular point of the interval, according to the PMF definition.

1.3.2 Distribution Fitting

The second step of the global modelling process is the distribution fitting. From the data collected on the real service times, one has to build a valid input for the proper analytical model selected in the final step (see below). The characteristics of this input are thus dictated by the selected modelling method. Three cases may be isolated:

- There exists some simple closed-form approximate analytical models (see Section 1.3.3). They are applicable for very small systems. The approximate method presented in Hopp and Spearman [43], for example, allows to quickly estimate the throughput from the two first moments, but the accuracy suffers from the simplicity of the approach (see Section 5.4). No proper distribution fitting is made in these cases. The moments (one or two) are just computed from the data.

- For every other analytical methods, phase-type distributions (exponential distributions in the simplest case) are required and have thus to be built, so that the Markov theory can be applied.

- In case simulation is applied, any distribution can be used.

In our research, the goal is to analytically model complex systems, with good accuracy. In other words, we are in the second of the three cases given in the
previous list. In the following state of the art, we thus focus on phase-type distributions fitting, i.e. on methods to build phase-type distributions from general distributions.

A **phase-type distribution** (PH) is a distribution which can be modelled as the time until absorption of a Markov chain with one absorbing state. It is fully characterized by the transition matrix of the Markov chain and the initial probabilities of starting in any of the states of the chain (see Neuts [64] or Latouche and Ramaswami [56] for reference books). A continuous PH distribution is thus a complex composition of exponential distributions, represented by a Markov process. Typical examples are the Erlang distribution (exponential distributions in series), the hyperexponential distribution (exponential distributions in parallel), the Coxian distribution, or the generalized exponential distribution. A discrete PH distribution is represented by the time until absorption in a discrete Markov chain. It thus has a fixed interval between two consecutive possible values. Note that any discrete distribution with the latter property and with a finite support can be represented as a discrete PH distribution. Other typical examples of discrete PH distributions are mixtures of geometric distributions or the negative binomial distribution.

The literature concerning PH distributions fitting is reviewed in the following sections. Most of the methods to approximate general distributions by phase-type distributions fit continuous PH distributions. We first list the main contributions in this literature and then focus on the few methods fitting discrete PH distributions. There are essentially three kinds of methods: moments fitting (the most popular one), maximum likelihood estimation and distance minimization.

**Moments Fitting**

The first fitting method, historically and in popularity, is moments fitting. This method computes the two (or three) first moments from the original distributions and the phase-type distributions are then built in order to get the same first moments. Various methods have been proposed, matching two or three moments, with various types of PH distributions, and for original distributions showing various properties.

Sauer and Chandy [74] (using hyperexponential and generalized Erlang distributions), Marie [60] (using mixed generalized Erlang distributions) and Whitt [99] (using hyperexponential and mixed Erlang distributions) were the first to propose closed-form two moments fittings. Altiok [3] proposed to add the third moment, using mixed Erlang distributions with two stages, and gave the necessary conditions. Botta and Harris [20] (using generalized hyperexponential distributions) and Johnson and Taaffe [47] (using mixtures of Erlang distributions), extended the applicability of three moments fitting. More recently, Osogami and Harchol-Balter [67] proposed a closed-form three moments fitting for any original distributions, using Erlang-Coxian distributions with an almost minimal number of phases. Subsequently, Bobbio et al. [19] found a fitting with a minimal number of phases (number which depends on the properties of the distributions), using acyclic PH distributions.
1.3. STEPS OF THE MODELLING PROCESS

Moments fitting is the most popular distribution approximation method. It has the advantage to be simple, computationally cheap, and to lead to a limited number of phases in good cases. However, the number of phases can also be quite large (see Section 5.4), while it directly influences the complexity of the global modelling (more precisely, the size of the Markov chain which has to be built and resolved). The minimal number of phases depends on the moments (see details in Bobbio et al. [19]). In particular, the following relation, proved by Aldous and Shepp [2], is well-known for two moments fitting: \( 1/k \leq cv^2 \leq 1/(k - 1) \), where \( k \) is the number of phases and \( cv^2 \) is the squared coefficient of variation of the original distribution.

Moreover, the extent to which the moments characterize the distributions is not so clear. McCullagh [61] showed by an example that two distributions having almost identical moment generating functions can have very different density functions. Subsequently, Lindsay and Basak [58] demonstrated, giving bounds on the difference between two distributions sharing common moments, that little can be said about the central part of the distribution, while the tail is more sharply defined. Pearson et al. [70] showed that distributions with the same four first moments can have very different shapes (percentiles). Concerning the application to queues, Whitt [100] showed that extremal distributions, with the same two first moments, may lead to very different performance measures on GI/M/1 queues (relative error of 200 percent), and Kincwicz and Whitt [52] showed that the error can be reduced by imposing shape constraints (8 percent) (see Section 5.4 for more details). Very little information is available about the error made when modelling more complex systems. Tancrez et al. [85] provide a very simple example, two three station tandem queues with triangular distributions showing the same two first moments, where the cycle time differs by 1.6\% (what is quite large compared to the results obtained with our approach, see Section 5.4).

Likelihood Maximization and Distance Minimization

The second type of method to approximate distributions is based on the computation of the maximum likelihood. In 1992, Bobbio and Cumani [15] proposed such a method, fitting a Coxian distribution which maximizes the likelihood with the original general distribution. Later, Asmussen et al. [10] made a significant contribution by applying the EM (expectation maximization) algorithm for the maximum likelihood estimation.

The third type of method aims at minimizing a given distance measure between the original and the approximated distributions. Examples of such method are the least squares method by Varah [98] and the method proposed by Schmickler [75], which mix distance minimization and moments fitting.

These two methods, likelihood maximization and distance minimization, are quite similar and share the same advantages and weaknesses. The first goal of their authors is to propose approximations that are more reliable than moments fitting, hoping that more evolved statistical concepts lead to better approximations. Unfortunately, these methods are more complex, computationally more expensive (they imply nonlinear optimization) and often lead to a larger number of phases.
Lang and Arthur [55] conducted a vast experimental study of state of the art fitting methods, based on moments fitting as well as on likelihood maximization. They concluded that methods are effective for continuous “phase-type behaved” (i.e. unimodal, without abrupt change) distributions and quite unsatisfying for others (low variance, steep increase or decrease, multiple modes, finite support). Concerning approximations based on the maximum likelihood principle, Lang and Arthur [55] claim that a “moderate order (order 4 or order 8)” is satisfactory for the continuous “phase-type behaved” distributions while a higher order is necessary for other distributions (the order is the number of states in the Markov chain corresponding to the PH distribution).

Discrete Phase-type Distributions

The fitting of discrete phase-type distributions has been markedly less investigated than the fitting of continuous PH distributions. Concerning moments fitting, Adan et al. [1] proposed a two first moments fitting method and Telek [89] gave the necessary conditions in terms of the second moment (which are less restrictive than for continuous distributions). The likelihood maximization was applied by Bobbio et al. [17] to fit acyclic discrete PH distributions (with infinite support). Bobbio et al. [18] propose to use the “scale factor” in order to choose between continuous and discrete PH distributions.

All in all, moments fitting and likelihood maximization have the same respective advantages and weaknesses than with continuous PH distributions. Bobbio et al. [18] summarized the advantages of discrete PH distributions: their ability to approximate original distributions with a low coefficient of variation, with abrupt changes, with finite support, or with deterministic values. Nevertheless, discrete PH distributions have the disadvantage to generate coincident events. In its fundamental book, Neuts [64] argue that continuous phase-type distributions are better suited to model heavy tails\(^2\). He however notes that continuous PH distributions are not well suited to model delayed distributions and distributions with steep increases or decreases.

1.3.3 Analytical Modelling

The third step of the global modelling process is the analytical modelling. Once the required tractable phase-type distributions have been built, the analytical methods can be applied. The final goal is to evaluate the performance measures of the system, in order to be able to help taking decisions, with a good knowledge of their consequences. Classical performance measures are the cycle time, the work-in-progress or the flow time. The cycle time is the average time between two units leaving the system. It is the inverse of the throughput which gives the number of items served in one unit of time. The work-in-progress gives the average number of items in the system, in the stations as well as waiting in the buffers. The flow time is the average time from an item’s entry in the system to its leaving.

\(^2\)Note that studies have shown that communication networks sometimes exhibit burstiness. In this case, heavy-tail distributions are appropriate to capture this feature. This gave rise to a dedicated literature (see Feldmann and Whitt [33], Starobinski and Sidi [77] or Riska et al. [72] for example)
1.3. STEPS OF THE MODELLING PROCESS

The literature related to distribution fitting and the literature related to the analytical modelling of queueing networks are quite separated. The former rarely studies the effect of the approximation on the global modelling process (some compute the accuracy reached on single queues but never on networks). The latter takes the PH distributions as given and neglect the effect of the preliminary step, on the approximation error for example.

In this research, we choose to use analytical methods, as opposed to simulation. Simulation is a classical and very general technique to analyze systems. It does not rely on any assumption and is therefore very general and flexible. Any distribution can be used and almost all systems can be simulated. The weakness of the simulation approach mainly lies in its development cost and in its running time. The application of simulation to tandem queues is presented in Diner and Deler [31], for example. Simulation and analytical models have both their usefulness. In a sense they are complementary as simulation allows to judge the quality of the analytical methods (this is done in the following to evaluate the tightness of the bounds and the accuracy of the approximations) and, vice versa, simulation is checked thanks to analytical models.

In our work, we choose to follow the analytical way. There are two kinds of analytical models: exact or approximate. We review the state of the art methods in more details in the following. Several comprehensive reviews of models for queueing networks are available, see Buzacott and Shanthikumar [22], Askin and Standridge [8], Perros [71], Gershwin [36], Altiok [5], Balsamo et al. [13], Dallery and Gershwin [26], Papadopoulos and Heavey [69], Govil and Fu [39].

Exact Analysis

Exact analytical models are the richest since they allow a direct and exact understanding of the influence of a decision variable on the performance of interest. Unfortunately, most queueing networks are too complex to be modeled exactly. Three methods are of significant interest:

Closed-form models. Closed-form results are available for very simple configurations, e.g. two station tandem queues and exponentially (or sometimes Erlang) distributed service times.

State models. These models build continuous (rarely discrete) Markov chains to exactly analyze systems with exponential or phase-type service time distributions. Based on the identified state space, a transition matrix is derived and the stationary equations are solved numerically to obtain the steady-state probabilities and infer the performance measures. The reader is referred for example to Hillier and Boling [42] for an early contribution, Gourgad et al. [38] for the study of tandem queues with exponential service time distributions and infinite buffers, or to Goossens et al. [37] for an exact analysis of a closed fork-join network with multi-server stations. However, these models’ applicability is limited to small instances as the state space size of the Markov chain increases quickly with the system size. This analytical method is applied in Chapter 3.
Holding time models. Introduced by Muth [62], this method aims to be computationally more efficient than state models for tandem queues with zero buffers. It considers the sequence of holding times (blocking time added to service time) for successive jobs at each station, and constructs recursive relationships. For an overview, the reader is also reported to Papadopoulos [68].

Approximate Analysis

The exact models suffer from their complexity, which makes their applicability limited to small networks. As a result, approximate analytical models have been proposed. The system to be analyzed is simplified in order to be analytically modeled. The most popular models are based on the idea of decomposing the system into smaller subsystems, and then including back the interdependencies between the subsystems. Approximate analysis keeps the development costs low. However, the uncertainty about the results is the weakness of the approach. Here are the four main approximate methods:

Closed-form models. Approximate closed-form models allow to estimate the cycle time from the two first moments very quickly but at the expense of the accuracy (see Buzacott and Shanthikumar [22] or Hopp and Spearman [43]).

Decomposition. The idea is to decompose the queueing network into smaller subsystems (made of one, two or three stations), analyze them in isolation and then mix the results iteratively. Solving more, but much easier, sub-problems, allows to approximately analyze the global system much more quickly and with good accuracy. A set of equations that determines the unknown parameters and the links between the subsystems is first derived. An iterative procedure is then used to solve the equations. This method was initially created for queues with exponentially distributed service times (see Dallery and Frein [25] for a good review and Gershwin [35] or Brandwajn and Jow [21] for examples). Some authors then extended it to systems with phase-type service times (see Altık [4] or van Vuuren and Adan [95] for continuous PH and Gun and Makowski [41] for discrete PH distributions). Also see Chapter 6, where the decomposition is applied, for more details.

Expansion. Introduced by Kerbache and Smith [50, 51], the generalized expansion method (GEM) is also based on the idea of decomposing the system. It supposes exponentially distributed service times and adds artificial nodes which registers the blocked jobs, so that the network can be decomposed similarly to Jackson’s networks. It is made of three stages: network reconfiguration, parameter estimation and feedback elimination. For recent examples of applications, see Kerbache and Smith [49] or Cruz et al. [24].

Aggregation. The idea behind aggregation methods is to replace a two stations subsystem by a single equivalent station. In this way, step by step, the queueing network is reduced to a single queue. Reference papers are the ones by Terracol and David [90] and by de Koster [30]. However, the aggregation methods are not always consistent. In general, decomposition methods have shown to be more successful.
1.3. STEPS OF THE MODELLING PROCESS

1.3.4 Bounds on the Performance Measures

In our work, we analyze queueing networks with general distributions and consider the complete modelling process, including the building of tractable distributions and the proper analytical modelling. As explained earlier, the distribution fitting unavoidably involves an approximation error in the final performance evaluation. In this context, bounds represent the only exact information available, and a valuable source of certainty. It can be used to test approximations or simulation models, or to develop approximations with known accuracy. van Dijk and Lamond [94] illustrate the potential usefulness of the bounds for engineering purposes, in a buffer allocation problem. In their example, the bounds offer a good idea of the optimal solution, and allow to secure the optimal region. As will be shown in the text, one of the contribution brought by our approach is bounds on the throughput.

Consequently, before ending this section, we provide a review of the literature concerning such bounds. The literature proposing bounds on the performance measures of queueing networks can be divided in two types: assuming phase-type distributions or not. Most papers assume phase-type distributions, and often exponential distributions (i.e. they focus on the second step of the modelling process). Very few bounding methodologies have been proposed for networks with general distributions.

Bounds with Phase-type Distributions

When phase-type service time distributions are assumed, exact solutions can be computed, theoretically. Researchers thus look for approximations and bounds which are quicker to compute. In the following, we cite the main bounding methodologies proposed in the literature.

A first approach relies on simplifying assumptions on the original system or modifications of it. It is then proved that these assumptions or modifications lead to bounds on the performance of the system (see van Dijk [93] for a unified view proposition). An early contribution was made by van Dijk and Lamond [94]. They considered a two station tandem queue with exponential service time distributions and a Poisson arrival process and, modifying the capacity limitations, got product form lower and upper bounds on the throughput. Shanthikumar and Jafari [76] then used a similar method and linked it to a decomposition technique (mentioned previously). Tcha et al. [88] extended the idea to more complex systems. Liu and Buzacott [59] proposed other throughput bounds related to a decomposition technique. To get a lower bound on the cycle time (upper bound on the throughput), they increase the capacity of all buffers to infinity, except for the buffer of the first subsystem. Onvural and Perros [65] proposed another modification of the network, using an equivalent closed network.

An alternative bounding methodology is known as bounded aggregation (see Courtois and Semal [23] and Balsamo et al. [13]). It decomposes the underlying Markov process, i.e. its state space, derives bounds on the stationary probabilities, with various levels of accuracy and complexity, and, from them, get bounds on the performance measures of the system. Kumar and coauthors proposed to use constraints on the behavior of the system to obtain a linear program which leads
to upper and lower bounds on the throughput (see Kumar and Kumar [53] and Kumar et al. [54]). Another approach, called Performance Bound Hierarchy, has been proposed by Eager and Sevcik [32]. Their recursive algorithm computes bounds whose tightness improves with the number of recursive steps.

**Bounds with General Distributions**

Concerning queueing networks with general distributions, few papers have proposed bounds. The first approach relies on the concavity and monotonicity properties of queueing systems (see Tsoucas and Walrand [92], Balsamo et al. [13], Dallery and Gershwin [26]). Increasing the buffers capacities leads to lower bounds on the cycle time while decreasing it leads to upper bounds. However, these properties are useful only if the modified network is tractable, which can be the case for zero or infinite buffer capacities. With such an approach, the achieved accuracy is poor, and it is impossible to evaluate the impact of the size of the buffers.

A second approach is the bounding methodology developed by Baccelli and Makowski [12], for systems with synchronization constraints and general distributions. It relies on the recursion equation of the system and on the theory of stochastic ordering. Baccelli and Liu [11] applied the methodology to stochastic decision free petri nets, which can model various queueing networks. They show that stochastic ordering of service times leads to stochastic ordering of the cycle time. Unfortunately, the tightness of the bounds has not been investigated.

Nakade [63] studied tandem queues with general service time distributions and zero buffer sizes. He found upper bounds on the cycle time using an equivalent synchronous tandem queue and lower bounds using recursive equations with respect to the waiting times. On some numerical examples, “the bounds give at most 8% relative errors”, and compare favorably to the bounds shown in van Dijk and Lamond [94] and Shanthikumar and Jafari [76].

**1.4 Outline of the Thesis**

As we detailed in the previous section, the modelling of queueing networks with general service time distributions is composed of three steps: data collection, distribution fitting and analytical modelling. However, the global modelling process is seldom studied as a whole: the effect of distribution fitting is rarely considered in analytical methods, and the interaction between both steps has been little investigated.

In our research, we propose a new alternative, called probability mass fitting, in the second step of the modelling process, i.e. to build tractable distributions, and study the global modelling method using PMF jointly with analytical models (state model or decomposition), on various network configurations (tandem queues, fork-join and split-and-merge queueing networks). PMF discretizes the distributions to get discrete phase-type distributions, by aggregating the probability masses on single values. We define PMF, study its properties and show that it has some interesting consequences on the global modelling process: bounds on the throughput, accurate approximations of the performance measures and of their distribution, and good behavior of the decomposition method.
In Chapter 2, we define probability mass fitting (PMF), beginning with some particular cases ("grouping at the end" and "grouping at the beginning") which we first studied historically and which allow to better understand the general case. Then, general probability mass fitting is defined. It gives rise to instant jobs, which complicates the analysis of the transformed system. This leads us to propose an alternative: PMF without instant jobs. For both alternatives, we give some properties. PMF is bounding: the discretized service time allows to bound the original one. PMF is also monotonic in the shift parameter $\alpha$, so that a value of the parameter exists which leads to the conservation of the mean of the original distribution. To end this chapter, we discuss the advantages and weaknesses of the PMF discretization.

In Chapter 3, we describe the global modelling method. PMF is used in the distribution fitting stage and a state model is applied in the analytical modelling stage (i.e. the evolution of the transformed system is exactly modeled by a Markov chain). The method is first presented on the simplest configuration: tandem queues. It is illustrated on a simple example and its complexity is discussed. Both PMF alternatives, with or without instant jobs, are considered. We then present the method for fork-join and for split-and-merge queueing networks, focusing on their peculiarities.

As already said, when queueing networks with general service time distributions are analysed, bounds are the only exact information reachable. Moreover, very few bounds are available in the literature. In Chapter 4, we present a first contribution of our work: our approach offers some certainty in the analysis of queueing networks with general service time distributions. This certainty is provided by two bounding methodologies. Both of them rely on the concept of critical path, which we present and discuss. The first bounding methodology extends the bounding property of probability mass fitting to the global method. It gives upper and lower bounds on the throughput, in transient or steady-state. Their accuracy is directly proportional to the discretization step used in the PMF. The second methodology computes a tight lower bound on the cycle time of the system. It relies on the fact that the critical path can be computed in the discretized time and that this sequence of jobs is shorter than the original critical path in the original time. Both bounding methodologies are primarily presented for fork-join networks. Their extension to split-and-merge networks is then discussed: the methodologies basically extend to split systems but not to merge systems. The tightness of the bounds is also shown on computational experiments. These bounds on the cycle time lead quite easily to bounds on related performance measures (idle time, utilization). At the end of the chapter, we show that the monotonicity property of PMF also extends to the global modelling method.

Once bounds are proved, the natural next step is to look for approximations. In Chapter 5, we first present some approximations on the cycle time and then turn to other performance measures of interest, such as work-in-progress and flow time. Their good accuracy is shown on extensive computational results, and their behavior is discussed in detail. Another interesting contribution of our approach lies in its ability to compute good approximations of the distribution of performance measures, i.e. not only the mean but also the shape. The distribution offers more detailed information on the behavior of the system, compared to the
isolated expectation. It allows to estimate measures such as the variance or the percentiles. To finish this chapter, we give some insights on how our global method and PMF compare to other similar methods, and methods using moments fitting in particular.

In Chapter 6, we present how the decomposition technique can be applied after the probability mass fitting, leading to a new global modelling method. We show that the specificities of PMF, and its ability to approximate the distributions of the throughput in particular, are advantageous in the application of decomposition. The good accuracy of the computed approximations is shown by numerical experiments, and the ability to approximate the cycle time distributions is shown on some examples. Finally, we conclude in Chapter 7 with a summary of our results, and give some future prospects.
Chapter 2

Probability Mass Fitting

As explained in the previous chapter, in order to model queueing networks with general service time distributions, tractable distributions have to be built before applying analytical methods. Our originality, and the starting idea of our research, lies in the way we build tractable distributions. We propose a new method to discretize the general service time distributions, called probability mass fitting.

Probability mass fitting is presented in details in the present chapter (and in Tancrez et al. [86]). In Section 2.1, we begin with two simple probability mass fittings, called “grouping at the end” and “grouping at the beginning”. These particular cases lead to a more general definition, given in Section 2.2 together with the properties of PMF. In order to avoid instant jobs (i.e. jobs of length zero) which complicate the modelling, an alternative PMF is proposed in Section 2.3. Finally, in Section 2.4, we discuss the characteristics, strengths and weaknesses, of probability mass fitting.

2.1 Particular PMF

To begin, we focus on a particular, simple, probability mass fitting (PMF) in order to provide intuition on the general definition given in the next section. The idea of probability mass fitting is quite simple. Our aim is to make the distribution tractable, i.e. transform it so that the Markov theory can be applied. The Markov theory can notably be applied on discrete distributions with regular values. The idea of PMF is simply to aggregate the probability masses of the original distribution on these discrete values.

The width of the intervals between two consecutive discrete values has to be constant in order to be able to apply the Markov theory afterward. A first simple solution for this is to select intervals of identical length (beginning in zero) and aggregate the probability mass computed on each interval at the end of the corresponding interval. This first, simple, probability mass fitting is called “grouping at the end”. With this PMF, the time step \( \tau \) is first chosen. Each distribution is then transformed into a discrete distribution by aggregating the probability mass.
Figure 2.1: Example of discretization by grouping at the end, with three steps.

in the interval \([ (k-1)\tau, k\tau ] \) \((k = 1, 2 \ldots a)\) on the end of the interval, i.e. on the point \(k\tau\). The parameter \(a\) gives the number of intervals, or, equivalently, the number of discrete values.

The discretization by grouping at the end is illustrated on Figure 2.1. The probability masses are first computed on the original distribution, on three intervals \((a = 3)\) of identical length \(\tau\) (Figure 2.1.a). They are then simply carried forward to the end of the corresponding interval, to get a discrete distribution (Figure 2.1.b). This distribution can be represented as a phase-type distribution, i.e. by a Markov chain with a single absorbing state (Figure 2.1.c). With this representation, the service time equals the time to reach the absorbing state \((A)\). It can be checked that the probability of a service time of length \(\tau\) is 0.5 \((2\tau: 0.5 \cdot 0.8 = 0.4, 3\tau: 0.5 \cdot 0.2 \cdot 1 = 0.1)\).

From the definition, it is easily inferred that grouping at the end is biased. As each job length is increased, the mean of the discretized distribution is larger than the mean of the original distribution. However, the transformation of each original job is exactly known: it is just carried forward to the end of the interval it belongs to. From this, we know that each service time is lengthened by one time step at most (the limit case occurs when the service time is pushed from the beginning of the step to its end). This provides bounds on the original job length from the discretized job length: \(l(w_{i,k}) - \tau \leq l(w_{i,k}) \leq l(w_{i,k})\), where \(l(w_{i,k})\) is the service time of job \(w_{i,k}\) discretized by grouping at the end. The reader interested in this particular PMF is referred to Tancrez et al. [85] and Tancrez and Semal [79] which focus on grouping at the end.

Another particular PMF follows naturally from “grouping at the end”. The discretization by “grouping at the beginning” computes the probability masses on intervals of identical length and aggregates each probability mass at the beginning of the corresponding interval. Each distribution is transformed into a discrete distribution by aggregating the probability mass in the interval \([ (k-1)\tau, k\tau ] \) \((k = 1, 2 \ldots a)\) on the point \((k-1)\tau\), i.e. the probability mass is carried backward to the beginning of the interval. The discretization by grouping at the end is illustrated on Figure 2.2. It is very similar to “grouping at the end” except that the discrete values are put in the beginnings of the intervals instead of the ends. Similarly to grouping at the end, as each job length is decreased, the mean of the discretized distribution is smaller than the mean of the original distribution. However, the error can be bounded. As each service time is shortened by one time step at most, bounds can be stated on the original job length: \(l(w_{i,k}) \leq l(w_{i,k}) \leq l(w_{i,k}) + \tau\),
2.2. PROBABILITY MASS FITTING WITH INSTANT JOBS

Figure 2.2: Example of discretization by grouping at the beginning, with three steps: (a) original distribution and probability masses, (b) discretized distribution, (c) phase-type representation.

where \( l(w_{i,k}) \) is the service time of job \( w_{i,k} \) discretized by grouping at the beginning. The reader interested in this particular PMF is referred to Tancrez et al. [81] which focus on grouping at the beginning.

2.2 Probability Mass Fitting with Instant Jobs

From the above particular cases, a more general discretization follows quite straightforwardly: the discrete value could be put anywhere in the interval, between the beginning and the end of it. This leads us to the general definition of probability mass fitting (PMF). It is given in this section, together with the properties of PMF.

2.2.1 Definition

Our aim is to get a discrete phase-type distribution, for which the size of the intervals between two consecutive values is constant (so that a Markov chain can then be built). For this, we simply aggregate the original probability mass around one discrete value on this value. The size of the interval on which the probability mass is computed is fixed equal to the size of the interval between two consecutive discrete values (so that the shape of the distribution is conserved). In order to give a more general definition of PMF, one degree of freedom is thus left: the shift of the probability mass interval around the discrete value. This parameter is denoted \( \alpha \). In the following, we give a more formal definition of probability mass fitting.

Let us suppose we have a positive service time distribution (with a finite support) and want to transform it into a tractable discrete distribution by probability mass fitting. To begin, we choose the number of non-zero discrete values, denoted \( a \). As we want to infer a discrete Markov chain from the transformed distribution, the size of the interval between two consecutive discrete values, denoted \( \tau \), has to be constant. The idea behind probability mass fitting is the following: the probability of each discrete value equals the original probability mass around this discrete value. The original distribution is transformed into a discrete one by aggregating on a discrete value the probability mass distributed in an interval around
CHAPTER 2. PROBABILITY MASS FITTING

Figure 2.3: Discretization by probability mass fitting with instant jobs, with $a = 3$.

this value. In order to fix the shift of the intervals, we introduce the parameter $\alpha$, with $0 \leq \alpha \leq \tau$, which gives the size of the part of the interval after the discrete value.

In summary, PMF transforms a given distribution into a discrete one by aggregating the probability mass distributed in the interval $((j - 1)\tau + \alpha, j\tau + \alpha]$ on the point $j\tau$ (where $j = 0, 1, \ldots, a$). The particular PMF presented in the previous section correspond to $\alpha = 0$ (grouping at the end) and $\alpha = \tau$ (grouping at the beginning). The discretization by probability mass fitting is illustrated in Figure 2.3. Note that $\tau$ gives the interval between two discrete values as well as the interval on which the probability mass is computed\(^1\), and that the first integrating interval is smaller. One of the main advantage of PMF, and one of its essential purpose, also appears on Figure 2.3: it preserves the shape of the distribution.

Also note that jobs of length zero, called instant jobs, have a non-zero probability. Instant jobs appear in our approach for modelling purposes essentially, when the distribution is discretized. However, such jobs have real sense in some cases. For example, a join (assembly) station with zero service time can be used to model a synchronization constraint between the preceding stations. The PMF presented in this section is thus called probability mass fitting with instant jobs. In the next section, we present the alternative PMF without instant jobs, as the latter complicates the modelling.

In practice, the parameters $a$, i.e. the number of non-zero discrete values, and the factor $\alpha/\tau$, i.e. the proportion of the interval which is located after the discrete value, are first chosen. The space $\tau$ between two discrete values is then deduced from these parameters so that the finite support of the original distribution is covered. In other words, we want the maximum original time $max$ to be equal to $a\tau + \alpha$, $\tau$ is thus chosen as follows:

$$\tau = \frac{\max - \alpha}{a}. \quad (2.1)$$

This equation can easily be reformulated as a function of the more convenient factor $\alpha/\tau$: $\tau = \frac{\max}{a + (\alpha/\tau)}$. The PMF discretization can be formally defined, and written as a simple mathematical equation. For this, let us first introduce some notations. A sample run in the sample space $\Omega = \{\omega\}$ is denoted $r(\omega)$. The Gantt chart of a sample run, for a three station tandem queue with buffer sizes

\(^1\)The integrating interval width could be a parameter (or even several parameters) but the shape of the distribution would be artificially and improperly modified.
b(1, 2) = b(2, 3) = 1, is given on Figure 2.4. In this chart, the time goes from left to right. The state of a station at a given time is represented either by a letter (B for blocked, S for starved) or by the job currently served (W_k is used instead of w_i,k in this chart, the station’s index is omitted as it is obvious from the chart). The state of a buffer is represented by the number of jobs waiting inside it. In this part of the run, eleven units leave the network.

To construct such a run, we only need a sequence of random service times drawn according to the service time distributions. The jobs then find their places in the run according to the structure of the network (see the synchronization constraints, Section 4.1.1). The realization of \( l(w_{i,k}) \) in this run, i.e. the time the job \( w_{i,k} \) takes in this particular run, is denoted \( l'(w_{i,k})(\omega) \). For the sake of readability, \( (\omega) \) will be omitted in the rest of the text. Moreover, we fix, without loss of generality, that a run starts at time zero. The start time may correspond to various situations. For simplicity, we assume that it is given by the start of the first job on the first station (i.e. the system is empty before the start time).

Moreover, we denote \( l_{\alpha}(w_{i,k}) \) the random variable giving the discretized service time corresponding to the original service time of job \( w_{i,k}, l(w_{i,k}) \). The discretized run and the realization of \( l_{\alpha}(w_{i,k}) \) in this run are denoted \( r_{\alpha} \) and \( l_{\alpha}'(w_{i,k}) \). Probability mass fitting with instant jobs aggregates the probability mass in the interval \( [(j-1)\tau + \alpha, j\tau + \alpha] \) on the point \( j\tau \). It can thus be formulated as follows:

\[
  l_{\alpha}'(w_{i,k}) \triangleq \left\lceil \frac{l'(w_{i,k}) - \alpha}{\tau} \right\rceil \tau, \quad \forall \alpha, r, i, k.
\]

### Properties

The PMF discretization has been well defined in the previous section. We now present some of its properties.

#### Bounding Property

One of the main advantages of probability mass fitting lies in the strong control it offers on the transformation of the individual service times. The transformation of each original service time to a discrete service time is exactly known. We say PMF is “intelligible”. Notably, we know that, when discretized, a service time will neither be shifted by more than \( \alpha \) to the left nor by more than \( \tau - \alpha \) to the right (see Figure 2.3). It leads to the following bounds on the service time.
Proposition 2.1. (Bounding property of PMF with instant jobs) The original service time of a job, \( l'(w_{i,k}) \), can be bounded using its discretized value, \( l_\alpha(w_{i,k}) \). We have:

\[
l_\alpha(w_{i,k}) - (\tau - \alpha) \leq l'(w_{i,k}) \leq l_\alpha(w_{i,k}) + \alpha, \quad \forall r, i, k.
\]  \hspace{1cm} (2.2)

Proof. This result can easily be checked on Figure 2.3. There are two limit cases. The lower bound corresponds to a service time of length \((j - 1)\tau + \alpha\) which is increased to \(j\tau\), with \(j = 1, \ldots, a\). The upper bound corresponds to a service time of length \(j\tau - \alpha\) which is decreased to \(j\tau\), with \(j = 0, \ldots, a\).

The main interest of these bounds on service times comes from their extension to bounds on the global run time, and thus on the throughput, which will be shown in Section 4.2.

Monotonicity

Obviously, the probability mass fitting does not, in general, conserve the expectation of the service time distributions. The expectation of the discretized distribution is not the same as the expectation of the original one, in contrast to moments fitting. More broadly speaking, PMF does not, in general, conserve the moments of the original distribution. We illustrate this point on tables 2.1 and 2.2. They show the average relative error made when the two first moments (expectation and variance) are computed in the PMF discretized time (with instant jobs), compared to the original value. The error is computed for 10 distributions that will be used in the next chapters for computational experiments: uniform(0,1), beta(1,3,1), beta(2,2), beta(4,4), beta(5.5,6), beta(8,8), beta(10,9), triangular(0,1,0.5), triangular(0.2,1,0.3) and triangular(0.1,0.9,0.6). These distributions are drawn on Figure 2.5. Table 2.1 clearly shows that the expectation
2.2. PROBABILITY MASS FITTING WITH INSTANT JOBS

\[ \frac{\alpha}{\tau} \]
\[ a = 4 \]
\[ a = 6 \]
\[ a = 8 \]
\[ a = 10 \]
\[ a = 15 \]
\[ a = 20 \]

\[
\begin{array}{cccccc}
0 & 32.4\% & 19.6\% & 14.0\% & 10.9\% & 7.0\% & 5.2\% \\
0.25 & 14.6\% & 9.3\% & 6.7\% & 5.3\% & 3.4\% & 2.5\% \\
0.5 & 0.61\% & 0.22\% & 0.10\% & 0.06\% & 0.03\% & 0.01\% \\
0.75 & 13.3\% & 8.7\% & 6.4\% & 5.1\% & 3.3\% & 2.5\% \\
1 & 24.5\% & 16.4\% & 12.3\% & 9.8\% & 6.6\% & 4.9\%
\end{array}
\]

<table>
<thead>
<tr>
<th>[ \frac{\alpha}{\tau} ]</th>
<th>[ a = 4 ]</th>
<th>[ a = 6 ]</th>
<th>[ a = 8 ]</th>
<th>[ a = 10 ]</th>
<th>[ a = 15 ]</th>
<th>[ a = 20 ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>28.5%</td>
<td>12.0%</td>
<td>6.2%</td>
<td>3.7%</td>
<td>1.5%</td>
<td>0.83%</td>
</tr>
<tr>
<td>0.25</td>
<td>25.1%</td>
<td>10.1%</td>
<td>5.7%</td>
<td>3.3%</td>
<td>1.4%</td>
<td>0.78%</td>
</tr>
<tr>
<td>0.5</td>
<td>23.2%</td>
<td>9.6%</td>
<td>5.3%</td>
<td>3.3%</td>
<td>1.4%</td>
<td>0.77%</td>
</tr>
<tr>
<td>0.75</td>
<td>19.9%</td>
<td>8.9%</td>
<td>4.7%</td>
<td>3.1%</td>
<td>1.4%</td>
<td>0.75%</td>
</tr>
<tr>
<td>1</td>
<td>18.5%</td>
<td>8.7%</td>
<td>4.6%</td>
<td>3.0%</td>
<td>1.3%</td>
<td>0.76%</td>
</tr>
</tbody>
</table>

Table 2.1: Average relative error made on the expectation, when the distribution is discretized by PMF with instant jobs, with \[ a \] and \[ \alpha \] changing.

Table 2.2: Average relative error made on the variance, when the distribution is discretized by PMF with instant jobs, with \[ a \] and \[ \alpha \] changing.

is not conserved, and that the relative error can be significant. However, it also shows that when \[ \alpha/\tau = 0.5 \], i.e. aggregating the probability masses in the middle of the intervals, the error made on the expectation is small. Concerning the variance, the relative error is quite large for any shift parameter \[ \alpha \], but it decreases when the PMF discretization refines (\[ a \] increases).

The error made on the expectation by the discretization is clearly a function of the parameter \[ \alpha \], i.e. the shift of the interval on which the probability mass is computed. Consequently, it is quite interesting to study the evolution of the discretized service times according to the parameter \[ \alpha \]. It can be checked that this evolution is a monotonic, more precisely decreasing, function.

**Proposition 2.2.** When discretizing by PMF with two parameters \[ \alpha \] such that \[ \alpha_1 \leq \alpha_2 \], and with a constant, the first discretized service time will always be larger than the second one. We have:

\[
l^{r}_{\alpha_2}(w_{i,k}) \geq l^{r}_{\alpha_1}(w_{i,k}), \quad \forall r, i, k.
\]

The expectation of a distribution discretized by probability mass fitting is thus decreasing when \[ \alpha \] is increasing.

**Proof.** To begin, note that \[ j\tau_1 + \alpha_1 \leq j\tau_2 + \alpha_2 \], \( \forall j \), as, by equation (2.1), \( j\tau + \alpha = j((\text{max} - \alpha)/a) + \alpha = (j\text{max} + \alpha(a - j))/a \), and as \( j \leq a \). Let \( l^r(w_{i,k}) \) be an original service time. Two cases have to be considered, according to its position in the discretization intervals.

- In the first case, the original service time \( l^r(w_{i,k}) \) lies between \( (j - 1)\tau_2 + \alpha_2 \) and \( j\tau_1 + \alpha_1 \), for a given \( j \). We thus get \( l^r_{\alpha_2}(w_{i,k}) = j\tau_2 \) and \( l^r_{\alpha_1}(w_{i,k}) = j\tau_1 \) and, as \( \tau_1 \geq \tau_2 \), the proposition inequality is valid in this case.
• In the second case, the original service time lies between \( j\tau + \alpha \) and \( j\tau + \alpha_2 \), for a given \( j \). We thus get \( l'_o(w_{i,k}) = j\tau_2 \) and \( l'_o(w_{i,k}) = (j+1)\tau_1 \) and the proposition inequality is also valid in this second case.

Probability mass fitting does not, in general, conserve the expectation of the original distribution. However, we show in the next proposition that a particular \( \alpha \) can always be found so that the expectation of the discrete distribution equals the expectation of the original distribution for a given \( \tau \).

Proposition 2.3. If the original cumulative distribution function \( F(t) \) is continuous, the evolution in \( \alpha \) of the expectation of the distribution discretized by PMF is also continuous. Consequently, a parameter \( \alpha \) can always be computed so that the expectation is conserved, i.e. so that the expectation of the discretized distribution is the same as the original one.

Proof. To prove the continuity in \( \alpha \) of \( E[t_\alpha] \), the expectation of a discretized distribution, we can simply formulate \( E[t_\alpha] \) as a sum of continuous functions. If \( f(t) \) and \( F(t) \) are the probability density function and the cumulative distribution function of the original distribution, \( E[t_\alpha] \) can be written as follows:

\[
E[t_\alpha] = \sum_{j=0}^{n} \int_{j\tau}^{(j+1)\tau + \alpha} f(t) \, dt = \sum_{j=1}^{n} \int_{(j-1)\tau}^{j\tau + \alpha} f(j\tau + \alpha) - F((j-1)\tau + \alpha)) \]

\[
= \tau(F(\tau + \alpha) - F(\alpha)) + 2\tau(F(2\tau + \alpha) - F(\tau + \alpha)) + \ldots \]

\[
+ \alpha \tau(F(\alpha + \alpha) - F((a-1)\tau + \alpha))
\]

\[
= \alpha \tau - \tau \sum_{j=0}^{a-1} F(j\tau + \alpha).
\]

This proves the first statement of the proposition. In order to prove that an \( \alpha \) leading to the expectation conservation can be found, we just have to show that the expectation \( E[t_\alpha] \) is larger than the expectation of the original distribution when \( \alpha \) equals zero and smaller when \( \alpha \) equals \( \tau \). As it is continuous, the expectation of the discretized distribution will thus equal the expectation of the original distribution for a given \( \alpha \). From equation (2.2), it can be seen that, when \( \alpha = 0 \), i.e. using grouping at the end, \( l'_c(w_{i,k}) \leq l'_o(w_{i,k}) \), in other words every service time is lengthened by the discretization. The expectation of the discretized distribution is thus larger than the original one. Similarly, from (2.2), when \( \alpha = \tau \), i.e. using grouping at the beginning, \( l'_c(w_{i,k}) \leq l'_o(w_{i,k}) \), and the expectation of the discretized distribution is thus smaller than the original one. This completes the proof.

Note that the distribution function \( F(t) \) is indeed continuous if the service time distributions are given in the form of histograms.

2.3 PMF without Instant Job

In the previous section, we introduced the basic probability mass fitting. This discretization potentially generates service times of length zero, called instant jobs, with non-zero probability of occurrence (see Figure 2.3). It is not difficult to guess that instant jobs complicate the modelling of the system and its resolution (see Section 3.1.2).
This leads us to propose an alternative to the basic PMF, an alternative which does not generate jobs with length zero. We call it “probability mass fitting without instant job”, and denote it PMF/nIJ. Moreover, in order to avoid confusion, we will, from now on, call the basic PMF presented previously “probability mass fitting with instant job”, and note it PMF/IJ. PMF will be used for the general concept, without differentiation between both alternatives. In the following, we briefly present PMF/nIJ and its properties.

2.3.1 Definition

Probability mass fitting without instant job is very similar to its brother with instant jobs. It is just modified in order to remove the possibility of instant jobs. The small difference thus only concerns the first interval (from 0 to \( \alpha \)). The original probability mass on this interval \([0, \alpha]\) is aggregated on \(\tau\) instead of 0. In summary, PMF/nIJ transforms a given distribution into a discrete one by aggregating the probability mass distributed in the interval \(((j-1)\tau + \alpha, j\tau + \alpha]\) on the point \(j\tau\), for \(j = 1, 2, \ldots, a\), and the mass in \([0, \alpha]\) on \(\tau\). PMF/nIJ is illustrated on Figure 2.6. Compared to Figure 2.3, the differences lie in the width of the interval on which the first probability mass is computed and in the absence of probability in zero. Using PMF/IJ or PMF/nIJ with the same parameter \(a\), the discretized distributions will have the same values for all but the first interval. Note that PMF/IJ and PMF/nIJ are equivalent when \(\alpha = 0\) (grouping at the end), or if there is no probability mass in \([0, \alpha]\).

The parameters \(a\), the number of discrete values, \(\alpha\), the shift parameter, and \(\tau\), the interval size, are defined exactly in the same way than for PMF/IJ. The space \(\tau\) between two discrete values is deduced from \(a\) and \(\alpha/\tau\) using equation (2.1).

The service times discretized by probability mass fitting without instant jobs are denoted \(l'_{\alpha'}(w_{i,k})\). The discretized run and the realization of \(l'_{\alpha'}(w_{i,k})\) in this run are denoted \(r_{\alpha'}\) and \(l'_{\alpha'}(w_{i,k})\). With these notations, PMF/nIJ can be formulated as follows:

\[
\begin{align*}
l'_{\alpha'}(w_{i,k}) &\triangleq \begin{cases} 
\tau, & \text{if } l'(w_{i,k}) \leq \tau + \alpha, \quad \forall r, i, k, \\
\left\lceil \frac{l'(w_{i,k}) - \alpha}{\tau} \right\rceil \tau, & \text{if } l'(w_{i,k}) > \tau + \alpha, \quad \forall r, i, k.
\end{cases}
\end{align*}
\]
The PMF/nIJ discretization is now well defined. In the following paragraphs, we present some of its properties, they are similar to the properties given for PMF/IJ (see Section 2.2.2).

2.3.2 Properties

Bounding Property

By definition, discretization by probability mass fitting, PMF/IJ as well as PMF/nIJ, gathers all the service times in an interval on a particular point of this interval. However, as the first interval is larger using PMF/nIJ (see Figure 2.6), the bounding properties of both alternatives are different.

**Proposition 2.4. (Bounding property of PMF without instant job)** The original service time of a job, \( l^r(w_{i,k}) \), can be bounded using its PMF/nIJ discretized value, \( l^r_{\alpha'}(w_{i,k}) \). We have:

\[
 l^r_{\alpha'}(w_{i,k}) - \tau \leq l^r(w_{i,k}) \leq l^r_{\alpha'}(w_{i,k}) + \alpha, \quad \forall r, i, k.
\]  

**Proof.** This result can easily be checked on Figure 2.6. There are two limit cases. The lower bound corresponds to a service time of length 0 which is increased to \( \tau \). The upper bound corresponds to a service time of length \( j\tau + \alpha \) which is decreased to \( j\tau, \forall j = 1, \ldots, a \). \( \square \)

It can be seen that the lower bound on the service times provided by PMF/nIJ is less tight than the one provided by PMF/IJ. This will lead to a less tight lower bound when these bounds on service times (2.3) will be extended to bounds on the cycle time (see Section 4.2).

Monotonicity

Probability mass fitting does not, in general, conserve the moments of the original distribution. This is illustrated point on tables 2.3 and 2.4, which give the average relative error made when the two first moments are computed in the PMF discretized time. The average error is computed for the ten distributions drawn on Figure 2.5. Similarly to PMF with instant jobs, the error can be quite large, except for the expectation when aggregating the probability masses in the middle of the intervals. The variance tends to be better approximated when using PMF without instant job (compare tables 2.2 and 2.4).
2.3. PMF WITHOUT INSTANT JOB

\[ \alpha / \tau = 4, 6, 8, 10, 15, 20 \]

<table>
<thead>
<tr>
<th>( \alpha / \tau )</th>
<th>0</th>
<th>0.25</th>
<th>0.5</th>
<th>0.75</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>18.5%</td>
<td>17.7%</td>
<td>16.9%</td>
<td>16.8%</td>
<td>16.0%</td>
</tr>
<tr>
<td>5</td>
<td>8.7%</td>
<td>8.1%</td>
<td>7.6%</td>
<td>8.3%</td>
<td>9.2%</td>
</tr>
<tr>
<td>6</td>
<td>4.6%</td>
<td>4.6%</td>
<td>4.7%</td>
<td>5.0%</td>
<td>5.7%</td>
</tr>
<tr>
<td>8</td>
<td>3.0%</td>
<td>3.0%</td>
<td>3.1%</td>
<td>3.3%</td>
<td>3.8%</td>
</tr>
<tr>
<td>10</td>
<td>1.3%</td>
<td>1.4%</td>
<td>1.4%</td>
<td>1.5%</td>
<td>1.8%</td>
</tr>
<tr>
<td>15</td>
<td>0.75%</td>
<td>0.79%</td>
<td>0.84%</td>
<td>0.90%</td>
<td>1.0%</td>
</tr>
<tr>
<td>20</td>
<td>0.7%</td>
<td>0.84%</td>
<td>0.90%</td>
<td>1.0%</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.4: Average relative error made on the variance, when the distribution is discretized by PMF without instant job, with \( a \) and \( \alpha \) changing.

As for probability mass fitting with instant jobs, the error made on the service time expectation is clearly a function of the parameter \( \alpha \). Identically to PMF/IJ, it can be shown that the service time discretized by PMF/nIJ is a monotonic, decreasing function of \( \alpha \).

**Proposition 2.5.** Discretizing by PMF/nIJ, with two parameters \( \alpha \) such that \( \alpha_1 \leq \alpha_2 \), and a constant, the first discretized service time will always be larger than the second one. We have:

\[ l_{\alpha_1}'(w_{i,k}) \geq l_{\alpha_2}'(w_{i,k}), \quad \forall r, i, k. \]

Consequently, the expectation of a distribution discretized by probability mass fitting without instant job is decreasing when \( \alpha \) is increasing.

**Proof.** As for Proposition 2.2, \( j\tau + \alpha_1 \leq j\tau_2 + \alpha_2 \), \( \forall j \), as \( j\tau + \alpha = j((\max - \alpha)/a) + \alpha = (j \max + \alpha(a - j))/a \), and as \( j \leq a \). Let \( l'(w_{i,k}) \) be an original service time. Three cases have to be considered, according to its position in the discretization intervals.

- In the first case, the original service time \( l'(w_{i,k}) \) lies between 0 and \( \alpha_2 \). We thus get \( l_{\alpha_2}'(w_{i,k}) = \tau_2 \) and \( l_{\alpha_1}'(w_{i,k}) = \tau_1 \) (since \( \tau_1 + \alpha_1 > \alpha_2 \)) and, as \( \tau_1 \geq \tau_2 \), the inequality is satisfied.

- In the second case, the original service time lies between \( (j-1)\tau_2 + \alpha_2 \) and \( j\tau_1 + \alpha_1 \), for a given \( j \) (where \( j = 1, \ldots, a \)). We thus get \( l_{\alpha_2}'(w_{i,k}) = j\tau_2 \) and \( l_{\alpha_1}'(w_{i,k}) = j\tau_1 \) and, as \( \tau_1 \geq \tau_2 \), the proposition inequality is valid in this case.

- In the third case, the original service time \( l'(w_{i,k}) \) lies between \( j\tau_1 + \alpha_1 \) and \( j\tau_2 + \alpha_2 \), for a given \( j \). We thus get \( l_{\alpha_2}'(w_{i,k}) = j\tau_2 \) and \( l_{\alpha_1}'(w_{i,k}) = (j+1)\tau_1 \) and this completes the proof.

The continuity in \( \alpha \) of the evolution of the expectation in discretized time can be proved under the same assumption as for PMF/IJ. However, in contrast to PMF/IJ, an \( \alpha \) cannot always be found in order to fit the original expectation. A supplementary condition has to be verified to assure that there exist a discretized expectation which is smaller than the original one.

**Proposition 2.6.** If the original cumulative distribution function \( F(t) \) is continuous, the evolution in \( \alpha \) of the expectation of the distribution discretized by PMF/nIJ is continuous. Moreover, a parameter \( \alpha \) can be computed so that the
expectation is conserved, i.e. so that the expectation of the discretized distribution is the same as the original one, if the following condition is satisfied:

$$a\tau - \tau \sum_{j=2}^{a} F(j\tau) \leq E[t],$$

Where $E[t]$ is the expectation of the original distribution.

Proof. To prove its continuity, we first formulate the expectation of a distribution discretized by PMF/ιJ, denoted $E[t_{\omega}]$. The probability density function of the original distribution is denoted $f(t)$, and its cumulative distribution function is denoted $F(t)$, $E[t_{\omega}]$ can be written as follows:

$$E[t_{\omega}] = \tau \int_{0}^{\alpha} f(t) \, dt + \sum_{j=1}^{a} j\tau \int_{(j-1)\tau}^{j\tau+\alpha} f(t) \, dt$$

$$= \tau F(\alpha) + \sum_{j=1}^{a} j\tau (F(j\tau + \alpha) - F((j-1)\tau + \alpha))$$

$$= \tau F(\alpha) + \tau(F(\tau + \alpha) - F(\alpha)) + 2\tau(F(2\tau + \alpha) - F(\tau + \alpha)) + \ldots$$

$$\ldots + a\tau(F(a\tau + \alpha) - F((a-1)\tau + \alpha))$$

$$= -\tau F(\tau + \alpha) - \ldots - \tau F((a-1)\tau + \alpha) + a\tau F(\text{max})$$

$$= a\tau - \tau \sum_{j=1}^{a-1} F(j\tau + \alpha).$$

Since $F(t)$ is assumed to be continuous, the continuity in $\alpha$ of $E[t_{\omega}]$ is obvious from the above equation. In order to prove that an $\alpha$ leading to the expectation conservation can be found, we just have to show that the expectation $E[t_{\omega}]$ is larger than the expectation of the original distribution when $\alpha = 0$ and smaller when $\alpha = \tau$ (under the given condition). As it is continuous, the expectation of the discretized distribution will thus equal the expectation of the original distribution for a given $\alpha$. From equation (2.3), it can be seen that, when $\alpha$ equals zero, $\tau'(w_{i,k}) \leq \tau'_i(w_{i,k})$, in other words every service time is lengthened by the discretization. The expectation of the discretized distribution is thus larger than the original one.

A similar argument can not be taken to find an expectation (in the discrete time) which is smaller than the expectation in the original time, as equation (2.3) leads to $\tau'_i(w_{i,k}) - \tau \leq \tau'(w_{i,k})$. By monotonicity (Proposition 2.5), we know that the limit case corresponds to $\alpha = \tau$, i.e. it leads to the smallest discretized jobs. The condition given in Proposition 2.6 simply states that the original expectation $E[t]$ should be larger than $E[t_{\omega}]$, the expectation of the distribution discretized with $\alpha = \tau$ (deduced from the above equation). If this condition is satisfied, an expectation (in the discretized time) which is smaller than the original one exists and this completes the proof. 

2.4 Strengths and Weaknesses

We end the presentation of probability mass fitting by a short discussion presenting its main strengths and weaknesses and the differences between PMF with or without instant job. The main strengths of probability mass fitting are the following.

- The idea of PMF is simple and intuitive. It is natural and easy to understand. It is probably the most natural, direct, way to build a discrete distribution from a continuous distribution. Accordingly, the computational cost is very low. The simplicity of the approach, (and its low cost) is an advantage compared to likelihood maximization and distance minimization methods.
• Probability mass fitting preserves the shape of the distribution. In fact, it could have been called “shape fitting”. This point can even be thought as a motivation of the idea of PMF. It is illustrated on Figure 2.7. Moreover, we show further on that this stays true for the computed throughput distribution. This characteristic makes a valuable difference with other fitting methods. Moments fitting, for example, sums up the information in two or three numbers and forgets the shape. Doing so, it looses much information, it modifies the shape of the distribution, and it does not preserve the percentiles for example (see Pearson et al. [70]). This point, the shape conservation, seems quite important. Indeed, in the literature, visual comparisons of the distribution functions is the most common way to evaluate the adequacy of the fitting (see for example the extensive comparison study by Lang and Arthur [55] mentioned in the introduction).

• PMF is particularly natural when the data about the service time distributions is collected in the form of histograms, the most common form in practice. PMF makes full use of the information contained in a histogram, without arbitrarily modifying the shape of the distribution, what is not the case when moments fitting is used for example.

• PMF is intelligible: it offers a strong control on the transformation of the individual service times. The transformation of each original service time to a discrete service time is exactly known. Moreover, each phase of the fitted discrete PH distribution has a clear meaning (for example, it may mean that the original service time is between 20 and 30 seconds). This is not the case with other fitting methods.

• As a result of the previous advantage, PMF allows to bound the original service time by the corresponding discretized time (see propositions 2.1 and 2.4). Moreover, we will show in Chapter 4 that PMF leads to bounds on the throughput of queueing networks with general service time distributions.

Figure 2.7: Original continuous distributions (beta(2,2) and triangular(0,1,0.5)) and discretized distributions by PMF/nIJ (α/τ = 0.5), with a = 5, 10 and 20 discretization steps.
(i.e. through the complete modelling process). This is another characteristic unshared by other fitting methods, and even other modelling methods in general.

- Probability mass fitting is **refinable**: the accuracy improves when the number of discrete values increases. Moreover, at the limit, when the step size decreases, PMF tends to the exact original distribution. It also means that the accuracy of the approximation can be chosen a priori, according to the affordable computational effort. This characteristic can also be considered as an advantage in comparison to moments fitting, which is less flexible.

- As will be shown later on, PMF leads to **accurate** estimations of the performance measures of the queueing system (see Chapter 5).

- PMF builds **discrete** phase-type distributions. As so, it allows to approximate distributions with a low variance, with abrupt changes, steep increases or decreases, or with deterministic values. These advantages of the discrete PH distributions are listed in Bobbio et al. [18], and were already given as weaknesses of continuous PH distributions by Lang and Arthur [55]. Note that Bobbio et al. [18] present coincident events as the main disadvantage of discrete PH distributions.

However, probability mass fitting has, of course, some **weaknesses**.

- PMF does not, in general, conserve the **moments** of the distribution (see tables 2.1 to 2.4). However, we showed that the parameter $\alpha$ can be chosen in order to conserve the expectation, if the original cumulative distribution function is continuous (see propositions 2.3 and 2.6). Moreover, the significance and the prominence of the moments to characterize distributions is sometimes discussed (see McCullagh [61] or Lindsay and Basak [58]). Also note that this weakness (moments non-conservation) is shared by likelihood maximization and distance minimization techniques.

- For the PMF to be applied as defined here, the distributions have to have a **finite** support (infinite support is impossible in the current setting). However, it is always the case in practice, in manufacturing in particular. In this context, this property can even be seen as an advantage compared to all other existing distributions fitting methods. Indeed, with other approaches, the support of the fitted distribution is always infinite (a finite support is impossible using continuous phase-type distributions).

The extension to infinite support distributions could be considered (for example, adding a loop transition in the last state of the PH distribution, state 3 in the example of Figure 2.1.c). In particular, note that the (commonly used) exponential distribution can be straightforwardly approximated using a geometric distribution (i.e. a discrete PH distribution showing only one non-absorbing state, with a loop). Moreover, the accuracy of the approximation can be improved by reducing the step size $\tau$, without increasing the number of states in the PH representation and then in the subsequent Markov chain.
• PMF is badly suited for service time distributions including rare events. Indeed, rare events extend the width of the distribution, leading to many insignificant discrete values in the discrete distribution (to reach the rare event). This increases the complexity without improving the accuracy. However note that this weakness is true for any other available distribution fitting method.

Finally, to compare both PMF alternatives, with or without instant job, we can say that instant jobs have a positive contribution on the accuracy of the distribution fitting, but at the expense of a more complex modelling. This trade-off will be studied in more details by computational experiments in the following. PMF with instant jobs also leads to better bounds, as the lower bound on the service time is tighter.

Probability mass fitting, with or without instant jobs, is now well defined and its properties have been shown. We are in position to present the global modelling method in the next chapter.
CHAPTER 2. PROBABILITY MASS FITTING
Chapter 3

Modelling Method

As explained in the introduction, the analytical modelling of queueing networks with general service time distributions has to be preceded by a tractable distribution building. In the previous chapter, we describe in details the way we build tractable distributions, i.e. probability mass fitting. We are now in a position to explain how the global modelling method works. In very few words, from the discrete service time distributions built by probability mass fitting, the evolution of the queueing system is described by a Markov chain, using a state model, and the performance measures of the system can then be estimated from the chain. Note that another analytical modelling could be used after PMF. A state model has the advantage to be exact, but suffers from high complexity. In Chapter 6, we study the application of an approximate method, called the decomposition technique, which allows to analyse larger systems.

In Section 3.1, we first describe the state model in details for tandem queues, using probability mass fitting with or without instant jobs. Then, we present the models for fork-join QN in Section 3.2, and split-and-merge QN in Section 3.3, focusing on their specificities. We remind the reader that the studied queueing networks have finite buffers, are open and acyclic, operate under saturation and with a FCFS discipline, and are composed of mono-server queues.

3.1 Tandem Queues

To begin, we explain the model on the simplest configuration: tandem queues. It allows us to present the application of the state model, which is a classical exact analytical method (see Section 1.3.3).

3.1.1 PMF Without Instant Job

We first present the modelling method relying on probability mass fitting without instant job, which is simpler. The method using probability mass fitting with instant jobs will be described in the next subsection.
Description

From the service time distributions (theoretical function or histogram), the method builds discrete distributions by probability mass fitting without instant job (see Section 2.3). The originality mainly lies in this step of the method. To begin, the number of discrete values, \( a \), and the parameter \( \alpha \) (or \( \alpha/\tau \)) are chosen. The interval width \( \tau \) is inferred from \( (2.1) \), for the longest distributions. Note that \( \tau \) has to be the same for each station’s distribution (so that a Markov chain modelling the whole system can be built). Every service time distribution is then transformed into a discrete distribution by aggregating the probability mass in the interval \( [(j-1)\tau + \alpha, j\tau + \alpha] \) on the point \( j\tau \), for \( j = 1, 2, \ldots, a \), and the mass in \( [0, \alpha] \) on \( \tau \) (see Figure 2.6). The discrete distributions obtained by PMF/nIJ can easily be formulated as a discrete phase-type distribution.

An analytical model can then be used. At this stage, we use a state model, which has the advantage to be exact (i.e. it provides and exact modelling of the modified system). It will notably allow us to compute bounds on the throughput (of the original system, see Chapter 4). Note that approximate analytical models could also be applied after PMF (see Chapter 6 for the application of the decomposition technique). With a state model, the evolution of the queueing system is described by a Markov chain whose states are the possible combinations of the stages of the various stations and the contents of the various buffers. The performance of the system can then be estimated from the analysis of the Markov chain. Transient performance measures, like the throughput at some time \( t \), can be derived from the matrix of transition probabilities. The steady-state performance measures (the cycle time, the work in progress or the flow time for example) can be computed from the steady-state probabilities derived from the Markov chain.

In Tancrez and Semal [79], the method has been named “Bounding Discrete Phase-type” (BDPH) since it relies on a discrete phase-type approximation of the various distributions of the system to be studied and since it leads to bounds.

Example

The method can be more clearly understood when explained on a simple example. Let us consider the simplest possible queueing network: a two station tandem queue (see Figure 3.1.a). The buffer size equals one. The original service time distributions are shown on Figure 3.1.b. The method can be applied directly to the distributions or on the histograms collected from them. In practice, most of the time, an analyst works with histograms measured from the real service times (Figure 3.1.c). At this stage, our method comes into play. The first step of the method aims to build tractable distributions. The histograms are transformed, by probability mass fitting without instant jobs, into the discrete distributions shown in Figure 3.1.d. The particular case of grouping at the end (\( \alpha = 0 \)) is illustrated (see Section 2.1). Next, the discrete service time distributions can easily be represented as discrete phase-type distributions (Figure 3.1.e).

Then, the second step of the method comes into play: the system is analytically modelled. The behavior of the system can now be modeled by a Markov Chain, i.e. using a state model. The Markov chain given in Figure 3.1.f lists all the possible recurrent states of the system and the transitions between these states. The first
3.1. TANDEM QUEUES

Figure 3.1: Steps of the method (in the particular case of grouping at the end) applied to a two station tandem queue (see a) with service times (b) collected in the form of a histogram (c): construction of the tractable distributions by PMF/nLI with $\alpha = 0$ (d), discrete PH representation (e) and Markov chain modelling the evolution of the system (f).

Symbol of a state refers to the first station, the second to the buffer and the third to the second station. Each station can be starved (S), blocked (B) or in some stage of service (for example, 1 means that the station already spent one time step serving the current job). Each buffer is described by its content (0 or 1 for a buffer of size one).

For example, state B12 means that the first station is blocked, that the buffer is full and that the second station already served the current unit during two time steps. Two transitions are possible from B12, depending if the second station continues to serve the same customer or ends. In the first case, the new state will be B13. The probability of this transition is 0.2, i.e. the probability given in the PH representation of Figure 3.1.e. In the second case, the second station ends its job, picks up the next unit in the buffer and begins to serve it. The first station can thus get rid of its blocking item and begins a new job. The new state is thus 111. From the Markov chain and its stationary probabilities, the performance measures can be computed. For example, the idle (blocked/starved) probability of a station is easily computed and the cycle time easily deduced from it.

The steps of the modelling process detailed in the introduction (Section 1.3) clearly appear on Figure 3.1. The data collection corresponds to the transition from (b) to (c). The tractable discrete PH distributions are built by “grouping at the end” from (c) to (d), and the PH representations, giving the conditional probabilities, are computed to go from (d) to (e). From them, the system can be analytically modelled, using a state model, by the Markov chain given in (f).
CHAPTER 3. MODELLING METHOD

Construction of the Transition Matrix

In order to use the proposed method, the transition matrix of the Markov chain modelling the evolution of the queueing network has to be constructed. In a few words, to do so, we first build the matrix including each individual station and buffer state combination and then remove the impossible ones. To build this matrix, we begin by looking at the states of a single station. Two transitions are possible for each station if it is in service (it stops or continues), and only one if it is starved or blocked. Once the states of the Markov chain are carefully ordered, the transition from one state to another can be represented by a matrix. For example, the state transition of a busy station which continues to serve corresponds to a matrix which shifts the station state from one stage of service to the next one. When we combine the states of the different stations to form the state of the tandem queue, the transitions can be formulated as a Kronecker product of the matrices corresponding to the transitions between the states of the stations. Finally, the whole matrix equals a sum of Kronecker products, one product for each kind of transition of the states of the system.

We now describe the construction of the Markov chain’s transition matrix in more details and sketch it in the form of a pseudo-code (see Algorithm 1). To begin, the states of the Markov chain have to be carefully ordered. As already said, each individual station and buffer state combination is first included in the matrix (and the impossible ones are then removed). The station states are ordered as follows: the starved state \( S \), the working states \( 1, 2, \ldots, a \), and the blocked state \( B \). The buffer states are sorted in increasing order: \( 0, 1, \ldots, b \). When gathered, the chain states are sorted in increasing order of the station indexes, and then in increasing order of the buffer indexes. In the example of Figure 3.1, the states are ordered as follows to build the matrix: \( S0S 10S 20S B0S S01 101 \ldots S02 102 \ldots S03 \ldots \)

\( S0B \ldots B0B S1S 11S 21S B1S S11 111 \ldots S12 112 \ldots S13 \ldots S1B \ldots B1B \). The transition matrix is built from matrices corresponding to the various transitions for stations and for buffers. These matrices are then combined using Kronecker products and sums. We now list these matrices, beginning with the matrices corresponding to the individual station transitions. They are called “station matrices”. The first matrix corresponds to a station \( i \) continuing its current job. In this case, the stage of the station is simply increased by one. The matrix shifts the station state from a stage of service to the next one. The probability for station \( i \) to continue when in stage \( j \) is denoted \( p_c(i, j) \). It is easily deduced from the discrete phase-type service time distributions. The variables \( b_p(MC(i)) \) and \( b_n(MC(i)) \) stands for the modification of the previous and next buffers, respectively, due to this station transition. As it continues to serve the current job, the station does neither take an item from the previous buffer nor put one in the next buffer. The station matrix \( MC(i) \) and the variables \( b_p(MC(i)) \) and \( b_n(MC(i)) \) have the following form. In the matrix, a dot “.” stands for a zero.

\[
MC(i) = \begin{pmatrix}
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & p_c(i, 1) & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & p_c(i, a - 1) & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\end{pmatrix}, \quad b_p(MC(i)) = 0, \quad b_n(MC(i)) = 0.
\]
The next three matrices correspond to a station $i$ ending its current job (probability $p_e(i,j)$, from stage $j$). $ME_1(i)$ stands for a station then beginning a new job (going to the second column, i.e. first stage of service). $ME_B(i)$ corresponds to a station which becomes blocked (last column). $ME_S(i)$ stands for a station which gets starved (first column). The variables $b_p(ME_1(i))$, $b_n(ME_1(i))$, $b_p(ME_B(i))$, $b_n(ME_B(i))$, $b_p(ME_S(i))$ and $b_n(ME_S(i))$ represent the modifications of the previous and next buffers in each case, due to the corresponding station transition.

$$ME_1(i) = \begin{pmatrix}
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & p_e(i,1) & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & p_e(i,a-1) & \ldots & \ldots & \ldots & \ldots \\
\ldots & 1 & \ldots & \ldots & \ldots & \ldots \\
\ldots & 1 & \ldots & \ldots & \ldots & \ldots \\
\end{pmatrix}, \quad b_p(ME_1(i)) = -1, \quad b_n(ME_1(i)) = 1.$$  

$$ME_B(i) = \begin{pmatrix}
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & p_e(i,1) & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & 1 & \ldots & \ldots & \ldots & \ldots \\
\end{pmatrix}, \quad b_p(ME_B(i)) = 0, \quad b_n(ME_B(i)) = 0.$$  

$$ME_S(i) = \begin{pmatrix}
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & p_e(i,1) & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & 1 & \ldots & \ldots & \ldots & \ldots \\
\end{pmatrix}, \quad b_p(ME_S(i)) = 0, \quad b_n(ME_S(i)) = 1.$$  

Note that the three previous matrices include the case were the station is blocked (last line of the matrices). Indeed, the transitions are the same if the station is working and ends, or if it is blocked (the probability to “end” is one in this case). The following matrices correspond to the case were the station is starved. The station then begins a new job ($MS_1$), if it gets a new job, or it stays starved ($MS_S$) otherwise. The previous and next buffer stages modifications, due to the corresponding station transitions, are given by the variables $b_p(MS_1)$, $b_n(MS_1)$, $b_p(MS_S)$ and $b_n(MS_S)$.

$$MS_1 = \begin{pmatrix}
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & 1 & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
\end{pmatrix}, \quad b_p(MS_1) = -1, \quad b_n(MS_1) = 0.$$
CHAPTER 3. MODELLING METHOD

Algorithm 1. Markov chain’s transition matrix construction

Define the station matrices: $MC(i), ME_1(i), ME_B(i), ME_S(i), MS_1, MS_S$.
Define the buffer matrices: $BM_0, BM_1, BM_{-1}$.

$M = 0.
\textbf{for} \text{ Each combination of station matrices do}
\text{ $SM(i)$ is the $i^{th}$ element of the combination.}
\textbf{if} \text{ compatible combination then}
\hspace{1em} A = SM(m) \otimes \ldots \otimes SM(2) \otimes SM(1).
\hspace{1em} \textbf{for} i = 2 \text{ to } m \text{ do}
\hspace{2em} buf(i) = b_n(SM(i - 1)) + b_p(SM(i)).
\hspace{1em} \textbf{end for}
\hspace{1em} A = BM(buf(m)) \otimes \ldots \otimes BM(buf(3)) \otimes BM(buf(2)) \otimes A.
\hspace{1em} M = M + A.
\textbf{end if}
\textbf{end for}
Remove impossible states from $M$.

$MS_S = \begin{pmatrix}
  1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\end{pmatrix} , \quad b_p(MS_S) = 0,$

$BM(0) = \begin{pmatrix}
  1 & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\end{pmatrix} , \quad BM(1) = \begin{pmatrix}
  \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\end{pmatrix} , \quad BM(-1) = \begin{pmatrix}
  \cdot & 1 & \cdot & \cdot & \cdot \\
  \cdot & \cdot & 1 & \cdot & \cdot \\
  \cdot & \cdot & \cdot & \cdot & \cdot \\
\end{pmatrix} .

Furthermore, three matrices are also needed to represent the modifications of the individual buffer states. They are called “buffer matrices”. The matrix $BM(0)$ correspond to a buffer staying at the same level, $BM(1)$ to a buffer getting one more item (shift to the next column), and $BM(-1)$ to a buffer loosing one item (shift to the previous column). The modifications of the buffers, i.e. which matrix to use, are deduced from the station transitions and the variables $b_p$ and $b_n$.

Once these matrices defined, the construction of the transition matrix become quite simple. Its pseudo-code is given in Algorithm 1. Each combination of the station matrices, i.e. of the station transitions, is considered. To each combination corresponds one series of $m$ matrices, one for each station. There are thus $6^m$ combinations, as there are 6 station matrices. For each combination, the station matrix corresponding to the $i^{th}$ station is denoted $SM(i)$. It is checked if the combination is compatible. Indeed, in particular cases, some station matrices can be incompatible. For example, a station which unstarves ($MS_1$) may not be preceded by a station which continues to serve the same job ($MC$) or by a station which was starved ($MS_1$ or $MS_S$), as the preceding station would not pass the unstarving item. A working station which becomes blocked ($ME_B$) may not be followed by a
3.1. TANDEM QUEUES

station which starts a new job \((ME_1\) or \(MS_1)\), as the following station inevitably makes some room available in the buffer (or directly takes the item). Then, the algorithm computes the Kronecker product of the station matrices (in the right order), giving the matrix corresponding to this station’s transition combination.

From the matrices of two successive stations, one can easily deduce the impact of the corresponding transitions on the buffer between the stations. For example, if a station continues to serve, and if the next station ends its current job and starts a new job, one item will be removed from the buffer \((b_n(MC(i)) + b_p(ME_1(i)) = 0 - 1)\), and this buffer transition is represented by the matrix \(BM(-1)\). In this way, we deduce the right buffer matrix for each buffer. The algorithm is then able to compute the transition matrix corresponding to the current combination, as a Kronecker product of the previously computed matrix, corresponding to the station transitions, and the matrices representing the corresponding buffer transitions. This part of the transition matrix is added to the global matrix under construction, and a new combination of stations matrices can be considered. When every combination has been considered, we get the transition matrix including each individual station and buffer state combination. Finally, the impossible states are removed from this matrix, to get the correct transition matrix of the Markov chain modelling the system. For this, the impossible states are simply detected by the fact that no other state leads to them (they correspond to zero columns in the matrix).

Stationarity of the Markov Chain

In order to compute the steady-state performance measures, like the cycle time or the WIP for example, the stationary equations of the Markov chain have then to be solved numerically. To be able to compute these probabilities, the Markov chain has to be stationary. The stationarity of the chain, and thus the existence of the stationary probabilities, can be shown quite easily. First, as said in the previous paragraph, when the transition matrix is built, impossible states are removed. This is done by removing the states with an all-zero column. Every state left can thus be reached from at least one other state. Second, there is a single recurrent class in the Markov chain. To show this, we have to see that, from every state, the tandem queue can be filled. The system can always be filled so that every station except the last one is blocked. Then, when the last station ends its job, the system state becomes \(1b_1b_2 \cdots b_{m-1}\). This state can thus be reached from every other state\(^1\). Similarly, every state of the Markov chain can be reached from \(1b_2b_3 \cdots b_{m-1}\), following some sequence of jobs, with appropriate service times. All states thus belong to the same recurrent class, and the Markov chain is stationary. Finally, note that the chain may be cyclic in some very unrealistic cases (e.g. all service times are multiples of \(j\tau\) with some \(j \geq 2\)). In this case, the solutions of the stationary equations may be used instead of the proper stationary probabilities (which do not exist).

\(^1\) This is true for non-degenerate cases. An example of degenerate case, for which \(1b_2b_3 \cdots b_{m-1}\) is never reached, happens when the last station always takes one time unit to do its job while other stations need two or more time units. In such (unrealistic) cases, another state which can be reached from every other state, could be found.
CHAPTER 3. MODELLING METHOD

<table>
<thead>
<tr>
<th>$a$</th>
<th>$m = 2$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
<th>$m = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>42 (0.9%)</td>
<td>186 (2.4%)</td>
<td>664 (0.92%)</td>
<td>2238 (0.37%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.1 sec.</td>
<td>0.3 sec.</td>
<td>4.8 sec.</td>
</tr>
<tr>
<td>6</td>
<td>88 (3.6%)</td>
<td>510 (1.0%)</td>
<td>2221 (0.33%)</td>
<td>8763 (0.12%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.2 sec.</td>
<td>1.2 sec.</td>
<td>41 sec.</td>
</tr>
<tr>
<td>8</td>
<td>150 (2.2%)</td>
<td>1062 (0.53%)</td>
<td>5319 (0.15%)</td>
<td>23475 (0.05%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.3 sec.</td>
<td>4.7 sec.</td>
<td>240 sec.</td>
</tr>
</tbody>
</table>

Table 3.1: Transition matrix size, sparsity (under parenthesis) and computational time (in italic), for tandem queues, using PMF without instant job, and with a total storage space equal to two ($B_S = 2$). The number $m$ of stations and the number $a$ of discretization steps vary.

Complexity of the Method

The computation of the stationary probabilities\(^2\), in order to evaluate the steady-state performance, is the main source of complexity in the method. The size of the Markov chain is, in first approximation, proportional to the number of individual stations and buffer states combinations (i.e. the size of the matrix first build and then reduced). This number is given by $\prod_{i=1}^{m} (a_i + 2) \cdot \prod_{i=2}^{m} (b_i + 1)$, with $m$ the number of stations, $a_i$ the number of discrete values in the discretized service time distribution of station $i$, and $b_i$ the size of buffer $i$. In Figure 3.1, we have $a_1 = 2$, $a_2 = 3$ and $b_1 = 1$. Note that this is a worst case estimation as a non-negligible amount of these individual combinations are impossible. For example, a non-empty buffer is never followed by a starved state and the last station is never blocked. In the example of Figure 3.1, the chain contains eleven real states while forty combinations can be formed ($= 4 \cdot 2 \cdot 5$).

It can be seen from the formula that the complexity quickly increases when the number of discrete values $a$ increases, in other words when the PMF discretization is refined. There is thus a clear trade-off between complexity and accuracy, which can be directly controlled by the parameter $a$. The size of the Markov chain also quickly increases with the complexity of the system configuration (number of stations and buffer sizes).

In tables 3.1 and 3.2, we show the actual matrix size and computational time\(^3\) (construction and inversion of the transition matrix). The sparsity of the transition matrix is also revealed by the fraction of non-zero elements in the matrix. Table 3.1 shows the impact of the number of stations while Table 3.2 shows the impact of the total storage space. In both cases, the matrix size and computational time increases very quickly when the system gets more complex. However, the sparsity is also more and more marked, when the system becomes more complex, what slows down the computational time increase. The matrix size increase is also illustrated on Figure 3.2.

\(^2\)The stationary probabilities $\pi$ are found by resolving the linear system of equation $\pi = \pi P$ where $P$ is the transition matrix of the Markov chain.

\(^3\)Using the Gaussian elimination implemented in MATLAB® on a 2.16 GHz usual PC, 2 GB RAM, Intel® Centrino Duo processor.
3.1. TANDEM QUEUES

<table>
<thead>
<tr>
<th>$a$</th>
<th>$B_S = 2$</th>
<th>$B_S = 4$</th>
<th>$B_S = 6$</th>
<th>$B_S = 8$</th>
<th>$B_S = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>186 (2.4%)</td>
<td>458 (1.0%)</td>
<td>858 (0.57%)</td>
<td>1386 (0.36%)</td>
<td>2042 (0.25%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.1 sec.</td>
<td>0.3 sec.</td>
<td>0.4 sec.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>510 (1.0%)</td>
<td>1357 (0.41%)</td>
<td>2631 (0.22%)</td>
<td>4335 (0.13%)</td>
<td>6471 (0.09%)</td>
</tr>
<tr>
<td></td>
<td>0.2 sec.</td>
<td>0.3 sec.</td>
<td>0.5 sec.</td>
<td>1.4 sec.</td>
<td>2.3 sec.</td>
</tr>
<tr>
<td>8</td>
<td>1062 (0.53%)</td>
<td>2980 (0.20%)</td>
<td>5908 (0.10%)</td>
<td>9851 (0.06%)</td>
<td>14813 (0.04%)</td>
</tr>
<tr>
<td></td>
<td>0.3 sec.</td>
<td>0.6 sec.</td>
<td>2.9 sec.</td>
<td>8.1 sec.</td>
<td>44 sec.</td>
</tr>
</tbody>
</table>

Table 3.2: Transition matrix size, sparsity (under parenthesis) and computational time (in italic), for three station tandem queues ($m = 3$), using PMF without instant job. The total storage space $B_S = \Sigma b_i$ and the number $a$ of discretization steps vary.

The explosion of the state space is the main limitation of the approach, to deal with realistic cases. This weakness can be overcome as usual when exact methods (a state model in our case) become too complex: using approximate methods. Decomposition or expansion models could be applied after probability mass fitting (see Chapter 6 for the application of the decomposition technique). Another source of improvement lies in the sparsity and the structure of the matrix. The construction of the matrix via Kronecker products gives it a very particular structure, which can be exploited to speed up the resolution (see Stewart [78] and Dayar [29] for example).

3.1.2 PMF With Instant Jobs

Using probability mass fitting with instant jobs, the method is slightly complicated by the potential service times of length zero, but the method is globally the same.

First, discrete distributions are built by PMF/IJ, from the data collected about the service times. The parameters $a$ and $\alpha$ are chosen, $\tau$ is inferred, and each distribution is then transformed into a discrete distribution by aggregating the probability mass distributed in the interval $[(j-1)\tau + \alpha, j\tau + \alpha]$ on the point $j\tau$ (where $j = 0, 1, \ldots, a$) (see Figure 2.3). The discrete distributions are then formulated as discrete phase-type and an analytical model, namely a state model, is applied. The evolution of the transformed system is exactly described by a Markov chain whose states are the possible combinations of the stages of the various stations and the contents of the various buffers. The performance measures, transient and steady-state, of the queueing system can then be estimated from the analysis of the Markov chain. The cycle time, for example, is computed as the sum of the average service time (in the discrete time), and the average blocked/starved time, which is easily computed (adding the stationary probabilities of the starved and the blocked states).

Concerning the Markov chain construction, one has to pay attention to the service times of length zero. Suppose the tandem queue is in a given state and a station ends its current job. At that time, this station may proceed one or more jobs instantly and, consequently, the next buffer content can increase by
Figure 3.2: Transition matrix size for tandem queues, using PMF without instant job, with various numbers $a$ of discretization steps. On the left hand-side figure, the number $m$ of stations vary ($B_2 = 2$). On the right hand-side figure, the total storage space $B_2 = \sum b_i$ vary ($m = 3$).

more than one unit in a single period. The Markov chain has thus to include the corresponding transitions. In this case, the transition probabilities can be deduced from the probability of an instant job on the station.

As for the method using PMF/nIJ, the method is well illustrated on an example. We consider the same example of a tandem queue with two stations and a buffer of size one. The information about the real service time distributions (Figure 3.3.a) can be given in the form of histograms (Figure 3.3.b). They are transformed, by probability mass fitting with instant jobs, into the discrete distributions shown in Figure 3.3.c, with the phase-type representation given in Figure 3.3.d. In the illustrated case, the particular PMF/IJ “grouping at the beginning” is used ($\alpha = \tau$, see Section 2.1). The behavior of the complete system can then be modeled by the Markov chain given in Figure 3.3.e, using the same symbols as in Figure 3.1.f (number of stages for the stations and occupation for the buffers).

For example, state B11 means that the first station is blocked, that the buffer is full and that the second station already served the current customer during one time step. When the tandem queue is in state B11, the second station may continue or end its service on the current job. In the first case, the new state will be B12. The probability of this transition is 0.2 (see the PH representation in Figure 3.3.d). In the second case, if the second station ends its job, both stations can begin a new job. Instant jobs may thus occur on both stations. The next system state depends on the number of instant jobs which occur on the first station compared to the number of instant jobs on the second station. If these numbers are equal, the next state will be 111. If one more instant job occurs on the first station, the tandem queue will stay in state B11. If one more instant job occurs on the second station, it will empty the buffer and the next system state will be 101. Finally, if there occurs two more instant jobs on the second station, it will become starved and the next state will be 10S.
3.1. TANDEM QUEUES

The transition probabilities are inferred from the same reasoning. For example, the probability of the transition from B11 to 101 equals the probability for the second station to end its current job, multiplied by the probability of having one more instant job on the second station. Let us denote $p_{i,0}$ the probability of an instant job on station $i$, i.e. $p_{1,0} = 0.4$ and $p_{2,0} = 0.5$, and $p_{i,j\rightarrow j+1}$ the probability, in station $i$, that the service time is greater or equal to $j + 1$ knowing that it is greater or equal to $j$ ($p_{2,1\rightarrow 2} = 0.2$ for example). With these notations we get:

$$P[B11 \rightarrow 101] = (1 - p_{2,1\rightarrow 2}) \sum_{n=0}^{\infty} (1 - p_{1,0})p_{1,0}^n(1 - p_{2,0})p_{2,0}^{n+1}$$

$$= (1 - p_{2,1\rightarrow 2}) \frac{(1 - p_{1,0})(1 - p_{2,0})p_{2,0}}{1 - p_{1,0}p_{2,0}}.$$

Proceeding this way with each state, the transition probabilities can be computed from the discrete distributions and the Markov chain can be built.

Concerning the complexity, the construction of the transition matrix, and its stationarity, the remarks are very similar to those given in Section 3.1.1 for the method using PMF/nIJ. The size of the Markov chain is, in worst case approximation, proportional to $\prod_{i=1}^{m}(a_i + 2) \cdot \prod_{i=2}^{m}(b_i + 1)$. However, the real number of states is larger than with PMF/nIJ. Moreover, as more transitions are possible due to the instant jobs, the transition matrix is less sparse for the method using PMF with instant jobs than for the method using PMF/nIJ and thus harder to solve in practice. This is the main weakness of the method using PMF/IJ compared to the method using PMF/nIJ. The building of the Markov chain transition matrix is very similar to the case of PMF without instant jobs. The matrix is build as a sum of Kronecker products. The stationarity of the Markov chain can also be proved similarly.

The complexity of the method using PMF with instant jobs is illustrated on Table 3.3. It can be seen that the required computational effort increases very quickly with the system size. Compared to PMF without instant job (see Table
CHAPTER 3. MODELLING METHOD

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m = 2$</td>
<td>$m = 3$</td>
<td>$m = 4$</td>
<td>$m = 5$</td>
</tr>
<tr>
<td>4</td>
<td>44 (19%)</td>
<td>226 (10%)</td>
<td>963 (6.2%)</td>
<td>3756 (4.2%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.4 sec.</td>
<td>11 sec.</td>
<td>900 sec.</td>
</tr>
<tr>
<td>6</td>
<td>90 (9.7%)</td>
<td>575 (4.3%)</td>
<td>2880 (2.2%)</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.7 sec.</td>
<td>36 sec.</td>
<td>×</td>
</tr>
<tr>
<td>8</td>
<td>152 (5.9%)</td>
<td>1156 (2.2%)</td>
<td>6522 (1.0%)</td>
<td>×</td>
</tr>
<tr>
<td></td>
<td>0.2 sec.</td>
<td>1.1 sec.</td>
<td>104 sec.</td>
<td>×</td>
</tr>
</tbody>
</table>

Table 3.3: Transition matrix size, sparsity (under parenthesis) and computational time (in italic), for tandem queues, using PMF with instant job, and with a total storage space equal to two. The number $m$ of stations and the number $a$ of discretization steps vary. The symbol × stands for a memory error, i.e. we could not compute the Markov chain transition matrix in these cases.

3.1), the computation is much more time consuming when instant jobs are possible. Checking tables 3.1 and 3.3 more carefully, it can be seen that the transition matrix size does not change very much (the number of states added due to instant jobs is not so large), but the matrix is much sparser in the case without instant job.

The difference between probability mass fitting with or without instant job, when part of the global modelling method, is illustrated on Figure 3.4. It appears, of course, in the service time distributions (the first step of the PMF/nIJ case gathers two intervals of the PMF/IJ case) and, more interestingly, in the Markov chains that model the evolution of the system. The difference comes from the fact that instant jobs make some transitions possible, while they are impossible with PMF/nIJ. For example, using PMF without instant job, four transitions are possible from 101. If PMF with instant jobs is used, eight transitions are possible from state 101 (see Figure 3.4). The chain obtained when instant jobs are possible has more states and is clearly more dense. This affects the speed of the method, in the construction of the chain as well as in its resolution, making the method using PMF/IJ slower (see Table 3.3). The transition matrix of the chain built when instant jobs are not possible is sparser than the matrix built when they are possible and, consequently, the linear system of equation which has to be solved to find the stationary probabilities is easier to solve. In the example of Figure 3.4, PMF without instant jobs leads to a chain with eight states, with two transitions from each state on average (i.e. two elements per line in the matrix). Using PMF with instant jobs, the chain has slightly more states (10), but, most importantly, it has many more transitions (5.4 per states on average).

3.2 Fork-Join QN

We now present the application of the analytical modelling method on more complex queueing networks, and we begin with fork-join queueing networks (FJQN, see Section 1.2.2 for a presentation). We focus on the description of the modelling
3.2. FORK-JOIN QN

Figure 3.4: Steps of the method (using PMF with instant jobs in the upper part and without in the lower part, with $a = 2$ and $\alpha/\tau = 0.5$, i.e. aggregating the probability mass in the middle of the interval) applied to a two station tandem queue with service times (a) collected in the form of histograms (b): construction of the tractable discrete distributions by PMF (c), PH representation (d) and Markov chain modelling the evolution of the system (e).

method using PMF without instant jobs and on the differences brought by FJQN compared to tandem queues. These differences are the same when PMF with instant jobs is used.

As for tandem queues, the method includes two steps. Tractable discrete distributions are first built by probability mass fitting without instant jobs. From them, the evolution of the fork-join queuing system can be modelled by a Markov chain, from which the performance can be evaluated. The states of the chain give the stages of the stations and the content of the buffers.

Compared to tandem queues, the starved and blocked stages of the stations have to be refined. Indeed, a join station can be starved by all its predecessors, but also by one of its predecessors, or by some of them (as we assume independent loading). For example, when a join station with two predecessors ends its job, if the first preceding buffer is empty while the other one is not, the station will take the item from the second buffer but will be starved by the first station (starved state denoted by $S^1$). Similarly, the station could be starved by the second station only ($S^2$), or by both ($S^{12}$). If a station has more than one successor, it can be blocked by one or more successors (as we assume independent loading). For example, when a station with two successors ends its job, if the first buffer is full while the second one is not, the station will pass an item to the second buffer but it will be blocked by the first successor (blocked state denoted by $B^1$). The station could be blocked by the second successor ($B^2$) or by both ($B^{12}$).

The application of the method on a simple example is illustrated on Figure 3.5, which shows a part of the Markov chain modelling the evolution of a join.
CHAPTER 3. MODELLING METHOD

Figure 3.5: A part of the Markov chain (b) modelling the evolution of a three station join system (a), obtained by the method using PMF/nIJ, with $a = 2$.

queueing network. The service time distributions and their transformations are omitted since they do not present any difference compared to previously related examples. Just note that the number of discrete values equals two in the discretized distributions, for every station. Each Markov chain state gives the stages of the stations and then the content of the buffers, sorted according to their numbering. For example, on Figure 3.5, state $12S^200$ means that the first and second stations already served their current unit during one and two time step, respectively, and that the third station is starved by the second station (he got the item from the first station but still need the item from the second station to assemble them). Moreover, in state $12S^200$, the buffer is empty (the second buffer has zero capacity and is thus always empty). From $11100$, state $12S^200$ is reached if the first and third stations end their job while the second one continues to serve its current job. In this case, the first station gives its item to the third one and begins to serve a new unit, while the third one is starved by the second station, which did not pass the required item. From $11100$, if both predecessors of that third station continue their job while the third station ends, the state will become $22S^100$ as the third station will miss both items. The whole chain can be built following this logic. A transition occurs depending on the stations ending or continuing to serve the same job, and the transition probability follows from the probabilities of these events, which are given by the discrete phase-type distributions.

The main difference of the modelling method for FJQN compared to tandem queues thus lies in the refined starved and blocked states. This slightly affects the complexity of the method, as the number of states of the Markov chain is increased. In first approximation, the number of states in the Markov chain is proportional to the number of individual stations and buffer states combinations: $\prod_{i=1}^{m} (a_i + 2^{ne_i} + 2^{nf_i}) \cdot \prod_{i,j} (b(i,j) + 1)$, with $m$ the number of stations, $a_i$ the number of discrete values in the discretized service time distribution of station $i$, $ne_i$ and $nf_i$ the number of predecessors and successors of station $i$, respectively, and $b(i,j)$ are the sizes of the buffers. Despite the slight difference, the evolution in the parameters $a$, and in the system size ($m$ and $b_1$) is the same. The complexity of the method, when applied to fork-join QN, is illustrated on Table 3.4, which can be compared to Table 3.1. The matrix size is slightly increased for FJQN, but, practically, it does not impact the computational effort. The total storage space
3.3. SPLIT-AND-MERGE QN

3.3. SPLIT-AND-MERGE QN

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m = 3</td>
<td>m = 4</td>
<td>m = 5</td>
</tr>
<tr>
<td>4</td>
<td>206 (2.1%)</td>
<td>740 (0.86%)</td>
<td>2337 (0.37%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.7 sec.</td>
<td>6.4 sec.</td>
</tr>
<tr>
<td>6</td>
<td>576 (0.89%)</td>
<td>2612 (0.29%)</td>
<td>9591 (0.11%)</td>
</tr>
<tr>
<td></td>
<td>0.2 sec.</td>
<td>1.4 sec.</td>
<td>37 sec.</td>
</tr>
<tr>
<td>8</td>
<td>1210 (0.46%)</td>
<td>6484 (0.13%)</td>
<td>26387 (0.05%)</td>
</tr>
<tr>
<td></td>
<td>0.3 sec.</td>
<td>5.9 sec.</td>
<td>235 sec.</td>
</tr>
</tbody>
</table>

Table 3.4: Transition matrix size, sparsity (under parenthesis) and computational time (in italic), for fork-join QN, using PMF without instant job, and with a total storage space equal to two. The number m of stations and the number a of discretization steps vary.

has similar influence than for tandem queues (see Table 3.2). The configuration of the network does not impact significantly the computational effort. As for tandem queues, the explosion of state space of the Markov chain is the main limitation of the method.

3.3 Split-and-Merge QN

Finally, in this section, we present the specificities of the modelling method applied to split-and-merge queueing networks (see Section 1.2.2 for a presentation). The first step of the modelling process, probability mass fitting without instant jobs, has already been extensively described. In the second step, i.e. the state model, the Markov chain is slightly different due to the characteristics of split-and-merge networks. Here are the three main differences:

- The blocked state of a station is refined. When a split station ends its current job, the item is routed to one of the successors with some routing probability. If the corresponding following buffer is full, the station will be blocked. If a station has two successors, he can be blocked by the first (state denoted \(B^1\)) or by the second one (\(B^2\)). Note that, unlike for FJQN, a station cannot be blocked by more than one station, and that the starved stage does not need to be refined (there is only one preceding buffer in SMQN).

- When a station has more than one successor, a finished item can be routed to each following buffer, according to the routing probabilities. This thus multiplies the number of transitions corresponding to a station ending its job, in the Markov chain. Accordingly, the transitions probabilities depend on the routing probabilities.

- In the discrete time, two stations can end their jobs exactly at the same time, unlike in the continuous time. So, when several jobs end simultaneously in several predecessors of a station, and if the buffer attached to this merge station has not enough room to store all of them, a priority rule has to be established. Indeed, in these case, some (but not all) of the predecessors
will be blocked. The rule allows to decide which of them will. In our work, we choose a simple rule: priority is given to the first stations, in terms of numbering, i.e. the buffer is primarily filled by the first stations. Other, more elaborated, rules could be fitted to the method (a rule randomly giving priority with the same probability for each buffer, for example).

As an example, the application of the modelling to a three station split network is illustrated on Figure 3.6. The number of values in the discretized distributions equals two. Each Markov chain state gives the stages of the stations and then the content of the buffers, sorted according to their numbering. For example, on Figure 3.6, state $B^21200$ means that the first station is blocked by its second successor (station number 3), that the second and third stations already served during one and two time steps, respectively, and that the first buffer is empty (the second buffer is always empty). This state is reached from $12110$ if the two first stations end their job, if the third station continue to serve the same job and if the item finished by the first station is routed to the third station (which is busy and thus blocks the first station, where $B^2$ stands for blocked by the second successor). If, in the otherwise same situation, the item is routed to the first station, it goes to the buffer and the first station can begin a new job, leading to the transition $12110 \rightarrow 11210$. Note that the probability of these transition equals the product of the probabilities for the stations to end or continue, times the routing probability to the first station ($p$).

Concerning the complexity, the size of the Markov chain is slightly changed compared to the application to other networks. In worst case approximation, the number of states is proportional to the number of individual stations and buffer states combinations: $\prod_{i=1}^{m}(a_i + n f_i + 1) \cdot \prod_{i=2}^{m}(b_i + 1)$, with $m$ the number of stations, $a_i$ the number of discrete values in the discretized service time distribution of serve $i$, $n f_i$ the number of successes of station $i$, respectively, and $b_i$ the size of buffer $i$. The complexity of the method applied on split-and-merge QN is illustrated on Table 3.5. It is similar to the cases of tandem queues (see Table 3.1) and of fork-join QN (see Table 3.4). The transition matrix size is smaller, because some distributions have smaller support (a split station should be quicker than its followers in order for the system to be balanced), and a station can thus be in less
3.3. SPLIT-AND-MERGE QN

<table>
<thead>
<tr>
<th>$a$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
<th>$m = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>167 (3.5%)</td>
<td>372 (1.9%)</td>
<td>1246 (0.76%)</td>
</tr>
<tr>
<td></td>
<td>0.2 sec.</td>
<td>1.2 sec.</td>
<td>15 sec.</td>
</tr>
<tr>
<td>6</td>
<td>478 (1.5%)</td>
<td>1411 (0.66%)</td>
<td>5880 (0.23%)</td>
</tr>
<tr>
<td></td>
<td>0.3 sec.</td>
<td>2.1 sec.</td>
<td>67 sec.</td>
</tr>
<tr>
<td>8</td>
<td>1015 (0.78%)</td>
<td>3603 (0.30%)</td>
<td>17763 (0.09%)</td>
</tr>
<tr>
<td></td>
<td>0.4 sec.</td>
<td>4.9 sec.</td>
<td>340 sec.</td>
</tr>
</tbody>
</table>

Table 3.5: Transition matrix size, sparsity (under parenthesis) and computational time (in italic), for split-and-merge QN, using PMF without instant job, and with a total storage space equal to two. The number $m$ of stations and the number $a$ of discretization steps vary.

stages of service. However, the matrix is less sparse, what tends to increase the computational time for systems with a large number of stations. Again, the main observation is the explosion of the size of the Markov chain, which is the main limitation of the state model.
Chapter 4

Bounds

In our work, we are interested in the modelling of queueing networks with general service time distributions. No exact models exist for such networks. Consequently, bounds are the only exact information available. They offer some certainty in the performance evaluation of complex systems, about the quality of the estimations. It can be used to test approximations or simulation models, or to develop approximations with known accuracy. However, few such bounds have been proposed in the literature (see Section 1.3.4). In this chapter, we propose two distinct bounding methodologies, leading to bounds on the throughput, for fork-join queueing networks (among which tandem queues).

The outline of the chapter is the following. In Section 4.1, we present the concept of critical path, which will show to be very useful in both bounding methodologies. Section 4.2 describes the first bounding methodology, which directly computes the bounds from the discretized system and essentially extends the bounding property of probability mass fitting. Then, in Section 4.3, we present the second bounding methodology, which takes advantage of the critical path computation. Computational results are presented for both methodologies. The bounds are primarily shown for fork-join queueing networks. The extension to split-and-merge queueing networks is discussed in Section 4.4. After these main results, we show that the bounds on the throughput lead to bounds on other performance measures, such as the idle time and the utilization (Section 4.5). Finally, the monotonicity of the computed throughput in the shift parameter $\alpha$ is shown in Section 4.6.

4.1 Critical Path

In this section, we present the notion of critical path. The critical path is a sequence of jobs that covers the running time. In other words, it allows to express the global running time as a sum of the individual service times, and thus to express the cycle time in terms of service times. Consequently, the critical path will allow us to show that the PMF properties (bounds, monotonicity, see sections 2.2.2 and 2.3.2) on the service times extend to the cycle time (see Section 4.2). Moreover, we will show that the critical path can be computed in the discretized time. The same sequence of job is similar (but not equal) to the critical path.
in the original time (which corresponds to the original cycle time). The critical path’s computation thus gets us close to the exact cycle time. This will allow us to prove a tight lower bound on the cycle time (see Section 4.3) and to propose an accurate approximation (see Section 5.1.2). Note that, in this subsection, the subscript $\alpha$ (and not $\alpha'$) is used but the results are valid for the PMF with instant jobs as well as for the PMF without instant job.

### 4.1.1 Synchronization Constraints

The notion of critical path, its existence and its construction, relies on the synchronization constraints of the queueing network, also called evolution equation, or state recursion (see Baccelli and Makowski [12]). This property can be found in Dallery et al. [27] for fork-join queueing networks with blocking (with blocking before service policy). In the following, we first give the synchronization constraints for the particular case of tandem queues, for which the equation is more intuitive. The synchronization constraints are then given for fork-join QN and from them we are able to define the critical path.

#### Tandem Queues

Let us consider a sample run $r$. In the case of a tandem queue, a job $k$ can only be started on station $i$ if:

- Its service in the previous station $i-1$ is ended (4.1).
- The service of the previous job $k-1$ in station $i$ is ended (4.2).
- This previous job $k-1$ is not blocked in station $i$ by some unfinished jobs downstream (4.3).

Furthermore, since there is no reason to wait once all these conditions are satisfied, the job will begin immediately when the last condition becomes satisfied. The moments a job $w_{i,k}$ starts and ends in the run $r$ are denoted $t^r\text{start}(w_{i,k})$ and $t^r\text{end}(w_{i,k})$.

**Lemma 4.1. (Synchronization constraints of a tandem queue)** Given an $m$ station tandem queue including a buffer of size $b_i$ before each station $i$, the moment a job $w_{i,k}$ starts, in a sample run $r$, is given by the following equation:

$$t^r\text{start}(w_{i,k}) = \max \left[ t^r\text{end}(w_{i-1,k}), t^r\text{end}(w_{i,k-1}), t^r\text{start}(w_{i+1,k-b_{i+1}-1}) \right],$$

where, for notation purpose, $t^r\text{end}(w_{0,k}) = 0, \forall k$. Moreover, the starting time of a job always equals the ending time of another job, as we have:

$$t^r\text{start}(w_{i+1,k-b_{i+1}-1}) = \max \left[ t^r\text{end}(w_{i+1,k-b_{i+1}-2}), t^r\text{end}(w_{i+2,k-b_{i+1}-b_{i+2}-3}), \ldots, t^r\text{end}(w_{m,k-\sum_{j=i+1}^{m} b_j-(m-i+1)}) \right].$$
4.1. CRITICAL PATH

Figure 4.1: Jobs which trigger the beginning of $w_{i,k}$, according to the preceding state in station $i$, and corresponding to (4.2), (4.1) and (4.3).

Proof. We first show that $t_{\text{start}}(w_{i,k})$ is always larger than the various terms in the right-hand side. For this, we just have to understand their practical meaning:

(4.1) A station $i$ cannot begin to serve item $k$ before the preceding station $i - 1$ has done its job on this item $k$.

(4.2) A station $i$ cannot begin to serve item $k$ before it has done its job on item $k - 1$.

(4.3) To begin to serve item $k$, station $i$ has to finish its job on item $k - 1$ (4.2) and get rid of it. There is space in the next buffer if station $i + 1$ already began to serve item $k - b_{i+1} - 1$, as, otherwise, items $k - 2$ to $k - b_{i+1} - 1$ are waiting in the buffer and block item $k - 1$ (the service discipline is first come first served).

The presence of a maximum in the equation comes from the fact that a job always begins due to the end of another job. There is no reason to wait once all the conditions are satisfied. In order to prove the first equation, we just have to prove that it gives every possibility. As each possible state preceding $w_{i,k}$ on the same station corresponds to one of the right-hand terms in the lemma, the equation gives all the possibilities:

(4.1) Before $w_{i,k}$, the station was starved, $t_{\text{start}}(w_{i,k})$ is thus given by (4.1): $w_{i,k}$ begins when the previous station pass the item on (see Figure 4.1.b, note that, in Figure 4.1, $W_{i,k}$ is used instead of $w_{i,k}$).

(4.2) The station was already in service previously the station begin to serve item $k$ directly when it ends on $k - 1$ (see Figure 4.1.a).

(4.3) The station was blocked, $t_{\text{start}}(w_{i,k})$ is thus given by (4.3): the station had to pass the blocking item on, what is possible when the start of a job in the next station vacates a space in the buffer between them (see Figure 4.1.c).

The second part of the lemma can be directly deduced from equation (4.1-4.3). Job $k - b_{i+1} - 1$ can only start on station $i + 1$ if the previous job $k - b_{i+1} - 2$ finished (leading to term (4.4)), and if there is room left in the next buffer for station $i + 1$ to get rid of $k - b_{i+1} - 2$ (4.5). The next condition means that item $k - b_{i+1} - b_{i+2} - 2$ should have started on station $i + 2$, which means $k - b_{i+1} - b_{i+2} - 3$ should have ended. Similar conditions can then be found for the following stations, till the end of the tandem queue (station $m$, term (4.6)).

Fork-Join QN

We now give the synchronization constraints for a fork-join queueing network (in a different form than Dallery et al. [27]). They are slightly more complex than for a tandem queue, due to the fact that a station can have several predecessors and successors (sets $E(i)$ and $F(i)$). Similarly to the case of tandem queues, a station $i$ can start a job $k$ if three conditions are satisfied.

- First, each previous station should have finished job $k$ (4.7).
• Second, the previous job on station $i$, job $k - 1$, has to be finished (4.8).

• Third, there should be some room left for this job $k - 1$ in each following buffer (4.9). In other words, the previous job $k - 1$ should not be blocked in station $i$.

Moreover, once all these conditions are satisfied, there is no reason to wait, and so, $w_{i,k}$ starts exactly when the last condition becomes satisfied.

**Lemma 4.2. (Synchronization constraints of a FJQN)** Given a fork-join queueing network, the moment a job $w_{i,k}$ starts, in a sample run $r$, is given by the following equation:

\[
t_{\text{start}}(w_{i,k}) = \max \left[ \max_{j \in E(i)} \left[ t_{\text{end}}(w_{j,k}) \right], t_{\text{end}}(w_{i,k-1}), \max_{j \in F(i)} \left[ t_{\text{start}}(w_{j,k-b(i,j)-1}) \right] \right],
\]

(4.7)

where, for notation purpose, $t_{\text{end}}(w_{0,k}) = 0, \forall k$. Moreover, the starting time of a job always equals the ending time of another job, as we have:

\[
t_{\text{start}}(w_{j,k-b(i,j)-1}) = \max \left[ t_{\text{end}}(w_{j,k-b(i,j)-2}), \max_{h \in F(j)} \left[ t_{\text{end}}(w_{h,k-b(i,j)-b(j,h)-3}) \right] \right].
\]

(4.10)

Proof. Each term corresponds to one of the conditions mentioned previously, and to one state (starved, working, or blocked) preceding $w_{i,k}$ on station $i$.

(4.7) Job $w_{i,k}$ cannot start before each preceding station $j \in E(i)$ gave unit $k$ to station $i$. If this term corresponds to the maximum, i.e. $t_{\text{start}}(w_{i,k}) = \max_{j \in E(i)} t_{\text{end}}(w_{j,k})$, it means station $i$ was starved before beginning to serve unit $k$.

(4.8) The station $i$ cannot begin to serve unit $k$ before it finished serving the previous unit $k - 1$. If $t_{\text{start}}(w_{i,k}) = t_{\text{end}}(w_{i,k-1})$, it means station $i$ was working just before starting job $w_{i,k}$.

(4.9) In fact, before starting $w_{i,k}$, station $i$ should not only finish to serve job $w_{i,k-1}$ but should also get rid of unit $k - 1$. For this, there should be some room left in each following buffer. This is not the case (and station $i$ gets blocked) if units $k - 2$ to $k - b(i,j) - 1$ are waiting in the queue preceding a station $j \in F(i)$ (in this case, the buffer is full). So, $w_{i,k}$ can only start when job $k - b(i,j) - 1$ began on station $j$. In this case, i.e. if $t_{\text{start}}(w_{i,k})$ equals term (4.9), it means station $i$ was blocked before beginning to serve unit $k$.

Once all the above conditions are satisfied, there is no reason to wait. Moreover, equation (4.7-4.9) gathers all conditions as every possible state preceding $w_{i,k}$ is considered (starved, working, or blocked).

The second part of the lemma can be directly deduced from equation (4.7-4.9). Job $k - b(i,j) - 1$ can only start on station $j$ if the previous job $k - b(i,j) - 2$ finished (leading to term (4.10)), and if there is room left in the next buffers for station $j$ to get rid of $k - b(i,j) - 2$ (4.11).

The last condition means that $k - b(i,j) - b(j,h) - 2$ should have started on station $h \in F(j)$, which means $k - b(i,j) - b(j,h) - 3$ should have ended. Similar conditions can then be found for the following stations.

These synchronization constraints are the basis which allows to define, build and compute the critical path.
4.1. CRITICAL PATH

Figure 4.2: Gantt chart of a sample run, for a three station tandem queue, with buffer sizes $b(1,2) = b(2,3) = 1$. The critical path is given in gray.

4.1.2 Definition and Construction

In this section, we give the formal definition of the critical path, show its existence and explain how it can be built. From this point on, we will assume the sample runs to be finite. During a sample run $r$, $n^r$ units are served\footnote{A given station is chosen to be the “reference” station, and given index $m$. The end of the run correspond to the end of the $n^r$th job on this station. In the case of tandem queues, we take the last station, i.e. $n^r$ equals the number of units leaving the system during the run. In the case of FJQN, the reference station is a chosen sink station.}. The critical path of the run $r$, denoted $cp(r)$, is defined as the sequence of jobs that covers the run $r$ without gap and without overlap. By definition, the length of a run $r$ can thus be written as a sum of job lengths:

$$l(r) = \sum_{w_{i,k} \in cp(r)} l'(w_{i,k}).$$

(4.13)

It can be seen from this equation that the critical path offers the way to relate the individual service times $l'(w_{i,k})$ and the length $l(r)$ of the run, i.e. the time to serve the $n^r$ units or, equivalently, $t_{\text{end}}(w_{m,n^r})$.

The critical path can be built quite easily. Starting with the last job that leaves the network (at the end of run $r$), we look which job’s end, in this precise run, has triggered its start (in other words which term among (4.7-4.12) equals $t_{\text{start}}(w_{m,n^r})$). This new job will be part of the critical path and, from it, the next job can be found in the same way. Repeating this process, we can proceed backwards in time until the start of the run. From Lemma 4.1, and as every job’s start is triggered by the end of another job, every run $r$ has at least one critical path.

The notion of critical path is well illustrated on the Gantt chart associated to a particular run. An example is given in Figure 4.2, where we draw the critical path on the run already shown on Figure 2.4 (for which $n^r = 11$). For the run depicted, the critical path, in gray, is made of the following backward sequence:

$$cp(r) = \{w_{3,11}, w_{2,11}, w_{3,8}, w_{3,7}, w_{2,7}, w_{1,7}, w_{1,6}, w_{3,1}\}.$$  

It can be seen on Figure 4.2 that the critical path covers the run and that each couple of successive jobs corresponds to one of the terms (4.7-4.12).

Furthermore, the most perceptive readers may also observe on Figure 4.2 that the course of the critical path among the stations can be related to the station
state before each job of the path. Indeed, the predecessor of a job in the critical path can be deduced from the state of the same station just before this job. If the station is previously working, the predecessor is obviously a job on the same station (and this corresponds to the term (4.8) in Lemma 4.1 and to Figure 4.1.b). If the station is previously starved, the predecessor is a job on the previous station for which the current station was waiting (term (4.7) and Figure 4.1.a). If the station is previously blocked, the predecessor is a job on the following station which was blocking the current station (terms (4.10-4.12) and Figure 4.1.c).

This property can also be checked on Figure 4.2. When the critical path is constructed backward in time, it jumps to the previous station when the current station was previously starved (w3,11 to w2,11 for example), to a following station when the current station was previously blocked (w2,11 to w3,8) and it stays on the same station if the station was previously working (w3,8 to w3,7). We will show in Section 4.1.4 that this characteristic is the key idea that allows us to compute the critical path.

4.1.3 Properties

In this section, we give some useful properties of the critical path. The first property relates the sequence of jobs defining the critical path in the original (resp. discretized) realisation and the same sequence of jobs in the discretized (resp. original) realisation. The other properties relate the number of jobs in the critical path and the number of units served during the sample run.

We begin with the first property. Let us consider the critical path of a sample run r. It is defined as the sequence of jobs, \( cp(r) = \{w_{i,k}\} \), that covers r without gap and without overlap. We would like to know the behavior of this sequence of jobs \( \{w_{i,k}\} \) in another run. The equation of Lemma 4.1 is valid for any run: a job \( w_{i,k} \) cannot be started before all the jobs on the right hand side are finished. This is just a static structural property of the system, independent of the run. Consequently, as two consecutive jobs in the sequence \( cp(r) = \{w_{i,k}\} \) satisfy this structural property, in another run, the same sequence of jobs will not show any overlap neither. The absence of overlap is independent of the run considered. However, which precise job end will trigger the start of job \( w_{i,k} \), i.e. which term of equation (4.7-4.12) will be satisfied at equality, depends on the service times and thus on the particular run we consider. In another run, gaps could thus appear between the jobs of the sequence \( \{w_{i,k}\} \).

In conclusion, while it forms the critical path in the run r, the sequence of jobs \( \{w_{i,k}\} \) will just be a non-overlapping path, maybe with gaps, in another run. In particular, the sequence of jobs making the critical path in the discretized run is just an non-overlapping path in the original run, and vice versa. Obviously, in a given run, the sum of the lengths of the jobs composing a non-overlapping path is always shorter than the length of the critical path, as the first may include gaps.

The second property relates the cardinality \( |cp(r_\alpha)| \) of the critical path of the discretized run \( r_\alpha \), i.e. the number of jobs in it, to the number \( n^r \) of units served during the run \( r_\alpha \). More precisely, it allows to compute the ratio between
4.1. CRITICAL PATH

Both values, in an infinite run. We denote \( c_\alpha \) the cycle time computed in the
 discretized time, and \( l_\alpha(w_{cp}) \) the average length of a job in the critical path in
discrete time.

**Proposition 4.3.** The ratio between the number of jobs in the critical path and
the number of units served can be computed as follows:

\[
\lim_{n^r \to \infty} \frac{|cp(r)|}{n^r} = \frac{c_\alpha}{l_\alpha(w_{cp})}.
\]

**Proof.** This property simply follows from the following equation:

\[
\frac{c_\alpha}{l_\alpha(w_{cp})} = \lim_{n^r \to \infty} \frac{l(r_\alpha)/n^r}{l(r_\alpha)/|cp(r_\alpha)|} = \lim_{n^r \to \infty} \frac{|cp(r_\alpha)|}{n^r}.
\]

The third property also relates the cardinality of the critical path to the
number \( n^r \) of units served. Namely, we show an upper bound on \( |cp(r)| \). It relies
on the following. When the critical path jumps, backward in time, to the previous
station, it goes over an empty buffer (as the station was previously starved) and the
current item is included twice in the critical path. When it jumps to a subsequent
station, it goes over \( n \) full buffer(s) (as the station was previously blocked) and
omits \( \sum b_i + n \) items, with \( b_i \) the size(s) of the buffer(s). In order to find an
upper bound for \( |cp(r)| \), the worst case thus corresponds to a critical path jumping
proportionally a maximal number of times to previous stations and a minimal
number of times to subsequent stations. In Proposition 4.4, this proportion is
given by \( \kappa \) and the additional term is just a correction term.

Before stating the proposition, we need some new notations. We denote \( \nu(i,j) \)
(resp. \( \mu(i,j) \)) the minimum (resp. maximum) number of stations on a way from
station \( i \) to station \( j \), \( i \) and \( j \) included. Moreover, \( bb(i,j) \) denotes the set of indexes
of the buffers between \( i \) and \( j \) following the way in the system leading to this \( \nu(i,j) \),
(i.e. \( |bb(i,j)| = \nu(i,j) - 1 \)). The longest and the shortest path from a station \( i \) to a
station \( j \) only depends on the particular topology of the studied queueing network.
Their computation is an easy problem in our case as the networks are relatively
simple (compared to other applications). We use the Floyd-Warshall algorithm,
which is well-suited for unweighted directed graphs.

**Proposition 4.4.** Let \( r \) be a sample run on a fork-join queueing network. The
number of jobs in the critical path satisfies the following upper bound:

\[
|cp(r)| \leq \kappa n^r + \eta - 1. \tag{4.14}
\]

Where,

\[
\kappa = \max_{i,j} \left[ \frac{\mu(i,j)}{\nu(i,j) + \sum_{k \in bb(i,j)} b_k} \right],
\]

and,

\[
\eta = \max_{i,j} (\mu(i,j)).
\]
**Proof.** Our goal here is to find an upper bound on \(|cp(r)|\) in function of \(n^r\). We know that each item served during the run \(r\) corresponds to one item index \(k\) of the jobs. The last job of the run is \(w_{m,n^r}\). In order to count the number of jobs in \(cp(r)\), we relate them to the underlying items. Three possibilities exist, corresponding to the right-hand terms in Lemma 4.2.

(4.7) The predecessor of a critical path job is a job on the same item, by a previous station (see Figure 4.1.b). The same item is thus included twice in the critical path.

(4.8) The predecessor is a job by the same station, on the previous item (see Figure 4.1.a). In this case, the number of jobs in \(cp\) corresponds to the number of items and to \(n^r\).

(4.9)-(4.12) The predecessor of a \(cp\) job on station \(i\) is a job on a next station \(j\) (see Figure 4.1.c). The items in the stations and in the buffers between stations \(i\) and \(j\) are not included in the critical path. Some items are omitted by the critical path. In this case, the number of jobs in the critical path is smaller than the number of served items. The number of omitted items can be deduced from Lemma 4.1. It equals \(\sum (b_l + 1)\) where \(b_l\) are the sizes of the buffers between stations \(i\) and \(j\).

As we are looking for an upper bound, we study the worst case, i.e. the case where \(|cp(r)|\) is the greatest in comparison to the number of units served \(n^r\). From the possibilities detailed in the previous list, we see that it happens when the critical path jumps upward a lot (4.7) and does not jump downward a lot (4.9)-(4.12). We consider two stations \(i\) and \(j\) with more than one way of stations linking \(i\) to \(j\). The length of the longest way is \(\mu(i,j)\) and the length of the shortest way is \(\nu(i,j)\). The worst case happens when the critical path pass in \(j\), then goes upward, station by station, to \(i\) through the longest way, and then goes back to \(j\) in one jump downward, through the shortest way. In this part of the critical path, \(|cp| = \mu(i,j) + 1\), as there is one \(cp\)'s job in each station of the longest way, plus one. By construction of this part of \(cp\), the items index are the same for the jobs in station \(j\) to \(i\), going upward (see point (4.7)). Let us call this index \(h\). When going back to \(j\), the item index becomes \(h - \sum_{k \in M(i,j)} b_k - \nu(i,j)\) (see Figure 4.1.c). In this part of \(cp\), the number of units served, \(n^r\), thus equals \(\sum_{k \in M(i,j)} b_k + \nu(i,j) + 1\).

If we imagine that the critical path is made of a sequence of such identical parts, the proportion of the number of jobs in \(cp\) to \(n^r\) remains the same. In this (worst) case, we have:

\[
\frac{|cp(r)|}{n^r} = \frac{\mu(i,j)}{\sum_{k \in M(i,j)} b_k + \nu(i,j)}
\]

(4.15)

Where one unit is removed from each term of the fraction because one job lies in both parts for each subsequent pair of \(cp\) parts. The wrongest case corresponds to stations \(i\) and \(j\) for which the fraction is maximal. This leads us to the definition of \(\kappa\).

Finally, we still have to take account of the fact that the critical path begins in a particular station and may end at another station. If the critical path goes upward between these stations, more jobs are included in \(cp\) while the number of units served is not increased (4.7). This justify the additional term, which corresponds to the longest way in the network, in Proposition 4.4 and completes the proof.

In the case of tandem queues, the previous proposition becomes simpler. In this case, there is only one possible path between two stations. When the critical path jumps, backward in time, to the previous station, it goes over an empty buffer and the current item is included twice in \(cp\). When it jumps to a next station, it goes over one or more full buffer(s) and omits \(\sum (b_l + 1)\) items. In order to find an upper bound for \(|cp(r)|\), the worst case thus corresponds to a critical path beginning on the last station and jumping step by step to the first station. In this case, the number of jobs in the critical path is \(n^r + m - 1\). This leads to Corollary 4.6, which is proved using the following lemma.

**Lemma 4.5.** In the case of tandem queues, we have \(\kappa \leq 1\).

**Proof.** In the case of tandem queues, there is only one way between two stations. Consequently, we have \(\mu(i,j) = \nu(i,j)\) and thus \(\kappa \leq 1\) (the term \(\sum b_k\) is a non-negative number).
4.1. CRITICAL PATH

Corollary 4.6. Let \( r \) be a sample run on a \( m \) station tandem queue. The number of jobs in the critical path satisfies the following upper bound:

\[
|cp(r)| \leq n^r + m - 1.
\]

Proof. This result is proved applying Proposition 4.4 and Lemma 4.5. Moreover, \( \mu(i,j) \) (defined in Proposition 4.4) is maximal between the first and the last stations, and \( \nu \) thus equals the number of station in the tandem queue, i.e. \( m \). This completes the proof.

Finally, another interesting particular case is provided by tandem queues with buffers of size zero, in an infinite run. In this case, the number of items counted twice when the critical path goes one station upward is the same as the number of items omitted when \( cp \) goes one station downward. In an infinite run, the number of omitted items equals the number of those counted twice. We get the following property.

Corollary 4.7. For a zero buffer tandem queue, in an infinite run, the number of jobs in the critical path equals the number of units served:

\[
\lim_{n^r \to \infty} \frac{|cp(r)|}{n^r} = 1.
\]

Proof. It can be seen from the proof of Proposition 4.4 that, for a zero buffer tandem queue, the fraction \((4.15)\) equals one. Moreover, in an infinite run, the additional term \((\eta - 1/n^r)\) tends to zero. This completes the proof.

The critical path is now well defined, and its properties are established. To end this section about this notion, we discuss how the critical path can be computed.

4.1.4 Computation

Beside its usefulness to prove bounds on the throughput, we will see below that the critical path also allows to compute bounds and approximations. To do this, the critical path has to be computed. By the computation of the critical path, we mean the computation of the steady-state conditional probabilities \( p_{cp}^\alpha(i,s) \) that, if the system is in state \( s \), the critical path “lies” on station \( i \). For example, in Figure 3.1, \( p_{cp}^\alpha(1,B12) = 0 \) and \( p_{cp}^\alpha(2,B12) = 1 \), as the critical path cannot lie on a blocked station. Note that we use the subscript \( \alpha \) because the critical path is, and can only be, computed in the discretized time.

The question is thus how the probabilities \( p_{cp}^\alpha(i,s) \) can be computed. They can be computed thanks to the fact that the predecessor of a job in the critical path can be deduced from the state of the same station just before this job (see Section 4.1.2). Let us consider a transition from a system state \( s \) to another system state \( s' \) and analyze the behavior of the critical path while this transition is encountered. We can infer on which station the critical path will lie in the system state \( s' \) if we know where \( cp \) lies in \( s \), if we know the individual station status in state \( s \). The critical path will lie in the (working) station \( i \) of \( s \) in three cases. First, going backward in time, \( cp \) will stay on station \( i \) if it was already on \( i \) in state \( s \) (see Figure 4.3.a). Second, \( cp \) will jump, going backward in time, to a preceding station \( i \) in \( s \) if it was on station \( i_f \) in \( s' \), with \( i_f \in F(i) \), and if the station was starved by station \( i \) previously, i.e. \( s_{i_f} = S_i \) (see Figure 4.3.b). Third, \( cp \) will jump, going backward in time, to a following station \( i \) in \( s \) if it was on station \( i_e \) in \( s' \), with \( i_e \in E(i) \), and if the station was blocked by station \( i \) previously, i.e.
CHAPTER 4. BOUNDS

Figure 4.3: The predecessor of a job in the critical path can be deduced from the state of the same station just before this job: (a) working, (b) starved, or (c) blocked.

$s_{i_e} = B^i$ (see Figure 4.3.c). To get the probability $p^{CP}_\alpha(i, s)$ that the critical path lies on station $i$ if the system is in state $s$, we thus simply have to consider these three cases for each possible transition from state $s$ to one of its successors and weight each transition by its probability. We finally get the following equation,

$$\forall i, s: p^{CP}_\alpha(i, s) = \sum_{s' \in \text{Suc}(s)} p[s \to s'] \left( p^{CP}_\alpha(i, s') + p_{\text{fol}} + p_{\text{pred1}} + p_{\text{pred2}} + \ldots \right),$$

with

$$p_{\text{fol}} = \sum_{i_f \in F(i)} \mathbb{1}_{\{s_{i_f} = S\}} p^{CP}_\alpha(i_f, s'),$$

$$p_{\text{pred1}} = \sum_{i_e \in E(i)} \mathbb{1}_{\{s_{i_e} = B^i\}} p^{CP}_\alpha(i_e, s'),$$

$$p_{\text{pred2}} = \sum_{i_e \in E(i)} \sum_{i_d \in E(i_e)} \mathbb{1}_{\{s_{i_e} = B^i, s_{i_d} = B^{i_e}\}} p^{CP}_\alpha(i_d, s'),$$

and where $\mathbb{1}_{\{\text{condition}\}}$ is the indicator function, i.e. it equals one if the condition is satisfied and zero otherwise. $S$ means starved (by any preceding station), $B$ means blocked (by any subsequent station), $p[s \to s']$ is the transition probability between the system states $s$ and $s'$, and Suc($s$) is the set of successors of $s$, i.e. $s' \in \text{Suc}(s)$ if $p[s \to s'] > 0$.

Concerning the complexity, to compute the probabilities $p^{CP}_\alpha(i, s)$, a linear system of equations has thus to be solved. The number of unknows of this system equals the number of stations $m$ (index $i$) times the number of system states (index $s$). The new system is thus $m$ times bigger than the one needed for the basic method (see Section 3.2). In first approximation, for a fork-join QN, its size is proportional to $m \cdot \prod_{i=1}^m (a_i + n e_i! + n f_i!) \cdot \prod_{i,j} (b(i, j) + 1)$. A trade-off between this supplementary computational cost and the accuracy benefit thus appears when the critical path computation is required in the applications, bounds and approximations, proposed further in the document (see subsections 4.3 and 5.1.2).
4.2. BOUNDS FROM THE DISCRETIZED SYSTEM

Table 4.1: Transition matrix size, sparsity (under parenthesis) and computational time (in italic), when computing the critical path, for tandem queues, using PMF without instant job, and with a total storage space equal to two. The number \( m \) of stations and the number \( a \) of discretization steps vary.

<table>
<thead>
<tr>
<th>( a )</th>
<th>( m = 2 )</th>
<th>( m = 3 )</th>
<th>( m = 4 )</th>
<th>( m = 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>84 (3.4%)</td>
<td>558 (0.79%)</td>
<td>2656 (0.23%)</td>
<td>11190 (0.07%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.2 sec.</td>
<td>1 sec.</td>
<td>13 sec.</td>
</tr>
<tr>
<td>6</td>
<td>176 (1.8%)</td>
<td>1530 (0.34%)</td>
<td>8884 (0.08%)</td>
<td>43815 (0.02%)</td>
</tr>
<tr>
<td></td>
<td>0.1 sec.</td>
<td>0.5 sec.</td>
<td>7.7 sec.</td>
<td>250 sec.</td>
</tr>
<tr>
<td>8</td>
<td>300 (1.1%)</td>
<td>3186 (0.17%)</td>
<td>21276 (0.04%)</td>
<td>117375 (0.01%)</td>
</tr>
<tr>
<td></td>
<td>0.2 sec.</td>
<td>1.7 sec.</td>
<td>53 sec.</td>
<td>2000 sec.</td>
</tr>
</tbody>
</table>

The cost of the critical path computation\(^2\) is illustrated on Table 4.1 for tandem queues. The case of fork-join and split-and-merge queueing networks is similar. Compared to the basic method complexity (see Table 3.1), the number of states is multiplied by \( m \) and the number of non-zero elements is divided by \( m \). In other words, the complexity increases but the sparsity of the matrix helps, and slows down the increase of the computational time.

This completes the section about the notion of critical path. It allows to relate the individual service times and the global running time (equation (4.13)), it has various interesting properties and it can be computed. In the next sections, we take advantage of this notion in various ways, in order to show bounds on the cycle time and on related performances, to prove the monotonicity of the computed cycle time, and to propose an approximation of the cycle time.

4.2 Bounds from the Discretized System

In this section, we present our first bounding methodology for fork-join queueing networks. We show that lower and upper bounds on the throughput can be directly computed from the system in the discretized time, i.e. using the basic modelling method presented in Chapter 3. The methodology relies on two ideas. First, it involves an intelligible transformation, i.e. with good knowledge of its effect on the service times. In case of the PMF discretization, it leads to the bounds on the individual service times presented in sections 2.2.2 and 2.3.2. Second, the concept of critical path allows to translate the effect of the transformation to the global time (see equation (4.13)), and to extend bounds on the service time to bounds on the global time.

4.2.1 Bounding Methodology

We first present the bounding methodology in a general form, independent of the chosen PMF discretization. For this, we introduce the following formalism.

\(^2\)Using the Gaussian elimination implemented in MATLAB® on a 2.16 GHz usual PC, 2 GB RAM, INTEL® Centrino Duo processor.
We consider a probability mass fitting which transforms the original service time \( l'(w_{i,k}) \) to the discrete service time \( \tilde{l}'(w_{i,k}) \), such that \( l'(w_{i,k}) \) can be bounded using its discretized value as follows:

\[
\tilde{l}'(w_{i,k}) - \delta_- \leq l'(w_{i,k}) \leq \tilde{l}'(w_{i,k}) + \delta_+, \quad \forall \, r, i, k.
\] (4.16)

The terms \( \delta_- \) and \( \delta_+ \) stands for the maximum value by which a service time can be increased and decreased, respectively, when discretized. Using PMF with instant jobs, we have \( \delta_- = (\tau - \alpha) \) and \( \delta_+ = \alpha \). Using PMF without instant job, we have \( \delta_- = \tau \) and \( \delta_+ = \alpha \).

In a Particular Run

To begin, we give the lemma on which the bounding methodology relies. It shows that, if the service time is bounded, the time needed to serve a given number of units in a particular run can also be bounded. This result will then be extended to the throughput and the cycle time. The run obtained from the PMF discretization of the original run \( r \) is denoted \( \tilde{r} \), and its length is denoted \( l(\tilde{r}) \).

**Lemma 4.8.** Given a PMF discretization of the service times \( l'(w_{i,k}) \) to \( \tilde{l}'(w_{i,k}) \) verifying (4.16), the time a FJQN takes to serve \( n_r \) units in a sample run \( r \) can be bounded as follows:

\[
l(\tilde{r}) - \delta_- (\kappa n^r + \eta - 1) \leq l(r) \leq l(\tilde{r}) + \delta_+ (\kappa n^r + \eta - 1).
\] (4.17)

**Proof.** Using equations (4.16) and (4.13) and the fact that \( cp(r) \) is just a non-overlapping path in the discretized run and is thus smaller than the critical path \( cp(\tilde{r}) \) in this run (see Section 4.1.3), we may write:

\[
l(r) \overset{(4.13)}{=} \sum_{w_{i,k} \in cp(r)} l'(w_{i,k}) \overset{(4.16)}{\leq} \sum_{w_{i,k} \in cp(r)} (\tilde{l}'(w_{i,k}) + \delta_+ ) \\
\leq \sum_{w_{i,k} \in cp(\tilde{r})} \tilde{l}'(w_{i,k}) + \delta_+ |cp(r)| \overset{(4.13)}{=} l(\tilde{r}) + \delta_+ |cp(r)|.
\]

As, by Proposition 4.4, \( |cp(\tilde{r})| \leq \kappa n^r + \eta - 1 \), we get the right inequality of (4.17). For the left inequality, using the same equations and the fact that \( cp(\tilde{r}) \) is a non-overlapping path in the original run \( r \), we get:

\[
l(r) \overset{(4.13)}{=} \sum_{w_{i,k} \in cp(r)} l'(w_{i,k}) \overset{(4.16)}{\geq} \sum_{w_{i,k} \in cp(\tilde{r})} l'(w_{i,k}) \\
\geq \sum_{w_{i,k} \in cp(\tilde{r})} (\tilde{l}'(w_{i,k}) - \delta_- ) \overset{(4.13)}{=} l(\tilde{r}) - \delta_- |cp(\tilde{r})|.
\]

As \( |cp(\tilde{r})| \leq \kappa n^r + \eta - 1 \) (Proposition 4.4), this completes the proof. \( \square \)

Lemma 4.8 provides an interesting result. Considering any sample run, bounds on the individual service times can be extended to bounds on the time it takes to serve a given number of units. Unfortunately, this result cannot yet be directly used since it refers to a sample run. For the results to be useful, we need to be able to say something about an expectation. This point is tackled in the following.
4.2. BOUNDS FROM THE DISCRETIZED SYSTEM

Transient Throughput

We first extend Lemma 4.8 to the expected time $T_P$ necessary to serve $P$ units. In a manufacturing system, $P$ corresponds to the number of products manufactured during the run. By definition, $T_P$ (resp. $\tilde{T}_P$) equals the sum of the lengths of all possible original (resp. discretized) runs $r$ (resp. $\tilde{r}$) during which $n^r = P$ units are served, weighted by their probabilities. Consequently, we simply have to check that each of the three terms of equation (4.17) is weighted in the same way. That comes from the definition of probability mass fitting. As it aggregates the probability masses, the total probability in continuous time of all the runs $r$ which discretize to the same run $\tilde{r}$ is equal to the probability of this run $\tilde{r}$ in discretized time. As this is true for each discrete run and as an original run has only one discrete correspondent, we can extend Lemma 4.8 to the following proposition.

**Proposition 4.9.** Given a PMF discretization of the service times $\bar{v}(w_{i,k})$ to $\bar{v}^d(w_{i,k})$ verifying (4.16), the expected time $T_P$ a FJQN takes to serve $P$ units can be bounded using $\tilde{T}_P$, the expected time to serve $P$ units computed in the discretized time. We have:

$$\tilde{T}_P - \delta_-(\kappa P + \eta - 1) \leq T_P \leq \tilde{T}_P + \delta_+(\kappa P + \eta - 1).$$

(4.18)

**Proof.** First, let us denote $r_P$ a finite sample run during which $P$ units are served, i.e. such that $n^r = P$. By definition, the expected time to serve $P$ units, $T_P$, equals the sum of the lengths of all possible runs $r_P$, weighted by their probabilities. More formally, we have: $T_P = \int f(r_P)l(r_P)dr_P$, where $f(r_P)$ is the density function of the runs $r_P$. To get (4.18) from (4.17), we have to check that each of the three terms of (4.17) is weighted in the same way. We thus relate $r_P$ to its discretized correspondent, $\tilde{r}_P$ (which also serves $P$ units). Let us note $\gamma(\tilde{r}_P)$ the set of original runs $r_P$ (in infinite number) which have the same discretized correspondent $\tilde{r}_P$.

We can decompose the previous integral and use Lemma 4.8, to get:

$$T_P = \sum_{\tilde{r}_P} \int_{r_P \in \gamma(\tilde{r}_P)} f(r_P)l(r_P)dr_P \leq \sum_{\tilde{r}_P} \left(\sum_{\tilde{r}_P} \int_{r_P \in \gamma(\tilde{r}_P)} f(r_P)dr_P\right) \geq \sum_{\tilde{r}_P} \int_{r_P \in \gamma(\tilde{r}_P)} f(r_P)dr_P.$$

As the PMF discretization simply aggregates the probability masses in intervals, $\int_{r_P \in \gamma(\tilde{r}_P)} f(r_P)dr_P$ gives the probability of the run $\tilde{r}_P$, i.e. $P[\tilde{r}_P]$. As $\sum_{\tilde{r}_P} P[\tilde{r}_P] = 1$, we get:

$$T_P \leq \sum_{\tilde{r}_P} l(\tilde{r}_P)P[\tilde{r}_P] + \delta_+(\kappa P + \eta - 1) = \tilde{T}_P + \delta_+(\kappa P + \eta - 1).$$

The way to the lower bound is very similar. Using Lemma 4.8, we get:

$$T_P \geq \sum_{\tilde{r}_P} \left(\sum_{\tilde{r}_P} \int_{r_P \in \gamma(\tilde{r}_P)} f(r_P)dr_P\right) \geq \tilde{T}_P - \delta_-(\kappa P + \eta - 1).$$

If we are interested in a fixed time instead of a fixed number of units served, bounds can straightforwardly be derived from the previous proposition.

**Proposition 4.10.** Given a PMF discretization of the service times $\bar{v}(w_{i,k})$ to $\bar{v}^d(w_{i,k})$ verifying (4.16), the expected number $P_T$ of units served by a FJQN during a fixed time $T$ can be bounded using the information computed in the discretized time:

$$\tilde{P}_T^- \leq P_T \leq \tilde{P}_T^+.$$
Where $\tilde{P}_{T-}$ is the expected number of units served during the discrete time $\tilde{T}_-$, with $\tilde{T}_-$ chosen maximal\(^3\) such that $\tilde{T}_- + \delta_+ (\kappa \tilde{P}_{T-} + \eta - 1) \leq T$. And $\tilde{P}_{T+}$ is the expected number of units served during the discrete time $\tilde{T}_+$, with $\tilde{T}_+$ chosen minimal\(^4\) such that $\tilde{T}_+ - \delta_- (\kappa \tilde{P}_{T+} + \eta - 1) \geq T$.

Proof. The lower bound, $\tilde{P}_{T-} \leq P_T$, follows from the upper bound in Proposition 4.9. If $T_-$ is the expected time, in the original time, to serve $\tilde{P}_{T-}$, we have $T_- \leq \tilde{T}_- + \delta_+ (\kappa \tilde{P}_{T-} + \eta - 1)$. By definition of $\tilde{T}_-$, we get $T_- \leq T$. Consequently, the number of units served in $T_-$, i.e., $\tilde{P}_{T-}$, is smaller than the one in $T$, i.e., $P_T$.

Similarly, the upper bound, $P_T \leq \tilde{P}_{T+}$, comes from the lower bound in Proposition 4.9. If $T_+$ is the expected time, in the original time, to serve $\tilde{P}_{T+}$, we have $T_+ - \delta_- (\kappa \tilde{P}_{T+} + \eta - 1) \leq \tilde{T}_+$. By definition of $\tilde{T}_+$, we get $T \leq \tilde{T}_+$. Consequently, the number of units served in $T$, i.e., $P_T$, is smaller than the one in $T_+$, i.e., $\tilde{P}_{T+}$.

Steady-state Cycle Time

When focusing on the cycle time, the results get simpler. The cycle time $c$ is defined as the average time, in steady-state, between two units leaving the network, i.e., $c = \lim_{P \to \infty} T_P / P$. The cycle time in the discretized time is denoted $\tilde{c}$, i.e., $\tilde{c} = \lim_{P \to \infty} \tilde{T}_P / P$. Note that the cycle time is the same for any station of the network.

**Proposition 4.11.** Given a PMF discretization of the service times $l^r(w_{i,k})$ to $\tilde{l}^r(w_{i,k})$ verifying (4.16), the cycle time $c$ of a FJQN can be bounded using $\tilde{c}$, the cycle time computed in the discretized time:

$$\tilde{c} - \kappa \delta_- \leq c \leq \tilde{c} + \kappa \delta_+,$$

Proof. These bounds straightforwardly follow from Proposition 4.9 by dividing (4.18) by $P$ and making $P \to \infty$.

The particular case of tandem queues leads to simpler bounds. The similarity between bounds on the service time (4.16) and on the cycle time (4.19) is remarkable.

**Corollary 4.12.** Given a PMF discretization of the service times $l^r(w_{i,k})$ to $\tilde{l}^r(w_{i,k})$ verifying (4.16), the cycle time $c$ of a tandem queue can be bounded using $\tilde{c}$, the cycle time computed in the discretized time:

$$\tilde{c} - \delta_- \leq c \leq \tilde{c} + \delta_+.$$ \hspace{1cm} (4.19)

Proof. This corollary follows directly from Proposition 4.11 and Lemma 4.5.

Propositions 4.9, 4.10 and 4.11 show that our methodology allows to extend bounds on the individual service times to prove bounds on the throughput, in transient or in steady-state. We believe that this bounding methodology is an interesting contribution. It could be extended to other service time transformations showing the bounding property (4.16).

\(^3\)In order to find $\tilde{T}_-$ and $\tilde{P}_{T-}$, $\tilde{T}_-$ (and consequently $\tilde{P}_{T-}$) is increased until $\tilde{T}_- + \delta_+ (\kappa \tilde{P}_{T-} + \eta - 1)$ becomes larger than $T$. $\tilde{T}_-$ and $\tilde{P}_{T-}$ are the last ones for which $\tilde{T}_- + \delta_+ (\kappa \tilde{P}_{T-} + \eta - 1) \leq T$.

\(^4\)In order to find $\tilde{T}_+$ and $\tilde{P}_{T+}$, $\tilde{T}_+$ (and consequently $\tilde{P}_{T+}$) is increased until $\tilde{T}_+ - \delta_- (\kappa \tilde{P}_{T+} + \eta - 1)$ becomes larger than $T$. $\tilde{T}_+$ and $\tilde{P}_{T+}$ are the first ones for which $\tilde{T}_+ - \delta_- (\kappa \tilde{P}_{T+} + \eta - 1) \geq T$. 

4.2. BOUNDS FROM THE DISCRETIZED SYSTEM

4.2.2 Application of the Methodology with PMF

In the previous section, we presented the bounding methodology in a general form and show that bounds on the service time can be extended to bounds on the throughput. In this section, the bounding methodology is applied to the modelling method presented in Chapter 3, using probability mass fitting.

PMF Bounds

The results given earlier can readily be instantiated to probability mass fitting with or without instant jobs.

Corollary 4.13. The method using PMF/nIJ allows to compute upper and lower bounds on the expected time \( T_P \) a FJQN takes to serve \( P \) units. With \( T_{P,\alpha'} \) the expected time to serve \( P \) units in the time discretized by PMF/nIJ, we have:

\[
T_{P,\alpha'} - \tau (\kappa P + \eta - 1) \leq T_P \leq T_{P,\alpha'} + \alpha (\kappa P + \eta - 1).
\]

Proof. This corollary is straightforward from propositions 4.9 and 2.4 (\( \delta = \tau \) and \( \delta_+ = \alpha \)).

Corollary 4.14. The method using PMF/IJ allows to compute upper and lower bounds on the expected time \( T_P \) a FJQN takes to serve \( P \) units. With \( T_{P,\alpha} \) the expected time to serve \( P \) units in the time discretized by PMF/IJ, we have:

\[
T_{P,\alpha} - (\tau - \alpha)(\kappa P + \eta - 1) \leq T_P \leq T_{P,\alpha} + \alpha (\kappa P + \eta - 1).
\]

Proof. This corollary is straightforward from propositions 4.9 and 2.1 (\( \delta_0 = \tau - \alpha \) and \( \delta_+ = \alpha \)).

Similar results could easily be shown for the number \( P_T \) of units served in a fixed time \( T \), from Proposition 4.10. As they provide little additional interest, we omit them and go directly to the bounds on the cycle time.

Corollary 4.15. (PMF/nIJ bounds) The method using PMF/nIJ allows to compute upper and lower bounds on the cycle time of a FJQN. With \( c_{\alpha'} \) the cycle time in the time discretized by PMF/nIJ, we have:

\[
c_{\alpha'} - \kappa \tau \leq c \leq c_{\alpha'} + \kappa \alpha.
\]

The width of the interval between both bounds equals \( \kappa (\tau + \alpha) \). Moreover, this equation reduces to \( c_{\alpha'} - \tau \leq c \leq c_{\alpha'} + \alpha \) in the case of tandem queues.

Proof. This corollary is straightforward from propositions 4.11 and 2.4 (\( \delta_0 = \tau \) and \( \delta_+ = \alpha \)), and Lemma 4.5.

Corollary 4.16. (PMF/IJ bounds) The method using PMF/IJ allows to compute upper and lower bounds on the cycle time of a FJQN. With \( c_{\alpha} \) the cycle time in the time discretized by PMF/IJ, we have:

\[
c_{\alpha} - \kappa (\tau - \alpha) \leq c \leq c_{\alpha} + \kappa \alpha.
\]

The width of the interval between both bounds equals \( \kappa \tau \). Moreover, this equation reduces to \( c_{\alpha} - (\tau - \alpha) \leq c \leq c_{\alpha} + \alpha \) in the case of tandem queues.
Proof. This corollary is straightforward from propositions 4.11 and 2.1 (δ_− = (τ − α) and δ_+ = α), and Lemma 4.5.

Note that the estimated cycle time \( c_\alpha \) (or, similarly, \( c_{\alpha'} \)) is easily computed in the discretized system, from the stationary probabilities of the Markov chain (see Figure 3.1).

The particular PMF presented in Section 2.1, grouping at the end and at the beginning, lead to simplified bounds and an intuitive explanation. Grouping at the end corresponds to PMF without instant job where \( \alpha = 0 \) (in this case, we use \( T_P \) and \( \bar{\tau} \) for \( T_{P,\alpha} \) and \( c_{\alpha'} \)). Using it, the discretized service times are always longer than the original ones, and the throughput measured in the discretized time gives an upper bound. Informally said, as every job of the critical path is lengthened, the global running time is also increased. The bounds on the throughput and on the cycle time become (from corollaries 4.13 and 4.15):

\[
T_P - \tau(\kappa P + \eta - 1) \leq T_P \leq T_P, \quad \text{and} \quad \bar{\tau} - \kappa \tau \leq c \leq \bar{\tau}.
\]  

(4.20)

Grouping at the beginning (PMF/IJ with \( \alpha = \tau \)) is the opposite (\( T_p \) and \( \underline{c} \) are used). It decreases each individual service time and thus leads to a direct lower bound on the throughput and on the cycle time. We get (from corollaries 4.14 and 4.16):

\[
T_P \leq T_P \leq T_P + \tau(\kappa P + \eta - 1), \quad \text{and} \quad \underline{c} \leq c \leq \underline{c} + \kappa \tau.
\]

The “direct” bounds, i.e. the upper bound using grouping at the end and the lower bound using grouping at the beginning, can be proved using a different bounding methodology, proposed by Baccelli and Makowski [12]. The latter also use the synchronization constraints of the system, in the form of a recursion equation. The bounding methodology is based on the stochastic ordering of the random variables representing the service times\(^5\). The “direct” bounds are immediate corollaries of results given in Baccelli and Liu [11] (proposition 3.1 and corollary 5.1), which applies the methodology to stochastic decision free Petri nets, which can be used to model fork-join QN with finite buffers. However, direct relations between the service times are required, i.e. the additional terms (\( \delta_- \) and \( \delta_+ \)) are not allowed. Consequently, the other bounds proposed previously cannot be proved using the Baccelli’s methodology.

The above results show that our method allows to bound the throughput of a FJQN, in transient or in steady-state, from below and from above. Furthermore, the accuracy of the bounds is directly related to the selected step size \( \tau \). The bounds thus become tighter, and converge, when the discretization step is decreased, i.e. when the number of discrete values is increased. In other words, the bounds are refinable. Moreover, it allows to a priori choose the desired accuracy of the results. Of course every accuracy improvement will require additional computational efforts caused by the increase of the state space size. But the latter

\(^5\) Using grouping at the end, we have \( l'(w_{i,k}) \leq \overline{P}(w_{i,k}) \implies l'(w_{i,k}) \leq u \overline{P}(w_{i,k}) \Rightarrow l'(w_{i,k}) \leq l_{\text{ex}} \overline{P}(w_{i,k}) \). Using grouping at the beginning, we have \( l'(w_{i,k}) \leq \overline{P}(w_{i,k}) \Rightarrow l'(w_{i,k}) \leq l_{\text{ex}} \overline{P}(w_{i,k}) \).
4.2. BOUNDS FROM THE DISCRETIZED SYSTEM

Figure 4.4: The method is applied on an example, using PMF/IJ. The bounds on the cycle time are computed for various step sizes $\tau$, i.e. various numbers $a$ of steps, with $\alpha/\tau = 0$ (▽), 0.25 (♦), 0.5 (⊙), 0.75 (□), 1 (△).

can be approximated (from the formula given in Section 3.2) as $a$, $b$ and $m$ are known. The trade-off between complexity and accuracy can thus be assessed a priori, from the step size $\tau$.

It can also be seen from these corollaries that PMF with instant jobs offers better bounds than PMF without instant job. The gap between both bounds is smaller with PMF/IJ. In particular, PMF/IJ allows to compute better lower bounds. This is natural since the absence of instant job requires the interval on which the first probability mass is collected to be larger (see Figure 2.6), and thus the lower bound on the service time to be worse. On Figure 4.4, we draw the bounds obtained using PMF with instant jobs, for a three station tandem queue with buffers of size one and service time distributions beta(2,2), uniform(0,1) and triangular(0,1,0.5). It can be seen that the bounds get sharper when the step size $\tau$ decreases. Moreover, the figure reveals that PMF/IJ with the shift parameter $\alpha/\tau = 1$, i.e. aggregating the probability mass in the beginning of the interval, leads to the best bounds. It comes from the fact that the step size $\tau$, and thus the distance between both bounds, decreases when $\alpha$ increases ($\tau = (\text{max} - \alpha)/a$, see equation (2.1)). More precisely, the lower bound shows to be better with $\alpha/\tau = 1$, while nothing can be said about the upper bound. As will be shown in the next section by computational experiments, this observation is not particular to the present example.

Computational Experiments

In this section, we show by computational experiments how the bounds behave. We first illustrate the impact of the shift parameter $\alpha$ and of the instant jobs
on the tightness of the bounds, focusing on three station tandem queues. Then, other configurations are analyzed, with a smaller scope of parameters, to show the behavior of the bounds for various FJQN.

To begin, we study an assortment of 500 three station tandem queues. The buffer configurations vary from \([0 \ 0]\) up to \([2 \ 2]\). They are equally shared out according to the global storage space (sum of both buffer sizes) and the space is uniformly divided, i.e. we analyze 100 \([0 \ 0]\) configurations, 100 either \([1 \ 0]\) or \([0 \ 1]\) configurations, 100 \([1 \ 1]\) configurations, 100 either \([2 \ 1]\) or \([1 \ 2]\) configurations, and 100 \([2 \ 2]\) configurations. The service time distributions are randomly chosen among the 10 following distributions: uniform\((0,1)\), beta\((1.3,1)\), beta\((2,2)\), beta\((4,4)\), beta\((5.5,6)\), beta\((8,8)\), beta\((10,9)\), triangular\((0,1,0.5)\), triangular\((0.2,1,0.3)\) and triangular\((0.1,0.9,0.6)\). In other words, for each new experiment, a service time distribution is randomly chosen for each station, among the ten distributions. These distributions are drawn on Figure 2.5. They have various expectations and various coefficients of variation. The 500 tandem queues are analyzed using both PMF alternatives (with or without instant job). Five different shift parameters \(\alpha\) are used \((\alpha/\tau = 0, 0.25, 0.5, 0.75, 1)\) as well as, with PMF/IJ, the parameters \(\alpha\) that conserve the expectations of the original distributions (see Proposition 2.3, noted “Exp. cons.” in the tables). The number \(a\) of non-zero values in the discretized distributions varies from five up to ten. We thus made a total of 33000 \((500 \text{ tandem queues} \cdot 5 \text{PMF + 6IJ} \cdot 6a)\) experiments. The 500 three station tandem queues were also analyzed by simulation, in order to assess the tightness of the bounds. In the following tables, we give the average relative error, in percents, between the results of our method and the results of the simulation (i.e. \([\text{sim} - \text{bound}] / \text{sim}\)).

For all experiments related throughout this text, the simulation model has been developed using the software ANYLOGIC®. The simulation time has been set equal to \(10^5\) time units, while the service time support is \([0, 1]\). This simulation time is long, so that the cycle time estimated by simulation does not change a lot. To show this we ran ten replications of the simulation for the same three station tandem queue. The relative difference between the maximal and the minimal cycle time estimations equals 0.14%. The average relative error with the average of the ten estimations equals 0.05%. For one particular system, the significance of the relative error between our methodology’s and the simulation’s result may be slightly influenced by the error made by the simulation (compared to the real value). However, throughout this text, we present average relative errors computed from a large number of experiments, and the slight influence of the simulation error thus vanishes. The simulation error sometimes leads to overestimate the error made by our method and sometimes leads to underestimate it, but, when put together, the influence of the simulation error on the estimation of our method’s error is negligible. To illustrate this, we runned twice a test set, leading to the same average relative errors (see Section 5.1.1).

The results of the experiments are summarized in tables 4.2 to 4.5. They give the relative errors of the lower and upper bounds, using PMF with or without instant job. The number \(a\) of non-zero values in the discrete distribution increases from left to right and the shift parameter \(\alpha\) increases from top to bottom. Several observations can be made:
4.2. **BOUNDS FROM THE DISCRETIZED SYSTEM**

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>17.4%</td>
<td>14.5%</td>
<td>12.4%</td>
<td>10.9%</td>
<td>9.7%</td>
<td>8.7%</td>
</tr>
<tr>
<td>0.25</td>
<td>16.3%</td>
<td>13.7%</td>
<td>11.8%</td>
<td>10.4%</td>
<td>9.3%</td>
<td>8.4%</td>
</tr>
<tr>
<td>0.5</td>
<td>15.3%</td>
<td>12.9%</td>
<td>11.2%</td>
<td>9.9%</td>
<td>8.8%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.75</td>
<td>14.3%</td>
<td>12.2%</td>
<td>10.6%</td>
<td>9.4%</td>
<td>8.4%</td>
<td>7.6%</td>
</tr>
<tr>
<td>1</td>
<td>13.4%</td>
<td>11.4%</td>
<td>10.0%</td>
<td>8.9%</td>
<td>8.0%</td>
<td>7.3%</td>
</tr>
</tbody>
</table>

Exp. cons. 17.4% 14.5% 12.4% 10.9% 9.7% 8.7%

Table 4.2: Average relative error reached by the lower bound on the cycle time, using PMF with instant jobs, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16.1%</td>
<td>13.4%</td>
<td>11.5%</td>
<td>10.5%</td>
<td>8.9%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.25</td>
<td>24.2%</td>
<td>20.4%</td>
<td>17.6%</td>
<td>15.4%</td>
<td>13.8%</td>
<td>12.4%</td>
</tr>
<tr>
<td>0.5</td>
<td>30.4%</td>
<td>25.7%</td>
<td>22.3%</td>
<td>19.7%</td>
<td>17.6%</td>
<td>15.9%</td>
</tr>
<tr>
<td>0.75</td>
<td>35.9%</td>
<td>30.6%</td>
<td>26.7%</td>
<td>23.7%</td>
<td>21.3%</td>
<td>19.3%</td>
</tr>
<tr>
<td>1</td>
<td>41.0%</td>
<td>35.2%</td>
<td>30.8%</td>
<td>27.4%</td>
<td>24.7%</td>
<td>22.5%</td>
</tr>
</tbody>
</table>

Exp. cons. 15.5% 13.1% 11.3% 10.0% 8.9% 8.0%

Table 4.3: Average relative error reached by the upper bound on the cycle time, using PMF with instant jobs, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16.1%</td>
<td>13.4%</td>
<td>11.5%</td>
<td>10.5%</td>
<td>8.9%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.25</td>
<td>15.7%</td>
<td>13.1%</td>
<td>11.3%</td>
<td>10.0%</td>
<td>8.9%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.5</td>
<td>15.3%</td>
<td>12.9%</td>
<td>11.2%</td>
<td>9.9%</td>
<td>8.8%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.75</td>
<td>15.1%</td>
<td>12.8%</td>
<td>11.1%</td>
<td>9.8%</td>
<td>8.8%</td>
<td>8.0%</td>
</tr>
<tr>
<td>1</td>
<td>14.9%</td>
<td>12.7%</td>
<td>11.1%</td>
<td>9.8%</td>
<td>8.8%</td>
<td>8.0%</td>
</tr>
</tbody>
</table>

Table 4.4: Average relative error reached by the lower bound on the cycle time, using PMF without instant jobs, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16.1%</td>
<td>13.4%</td>
<td>11.5%</td>
<td>10.5%</td>
<td>8.9%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.25</td>
<td>15.7%</td>
<td>13.1%</td>
<td>11.3%</td>
<td>10.0%</td>
<td>8.9%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.5</td>
<td>15.3%</td>
<td>12.9%</td>
<td>11.2%</td>
<td>9.9%</td>
<td>8.8%</td>
<td>8.0%</td>
</tr>
<tr>
<td>0.75</td>
<td>15.1%</td>
<td>12.8%</td>
<td>11.1%</td>
<td>9.8%</td>
<td>8.8%</td>
<td>8.0%</td>
</tr>
<tr>
<td>1</td>
<td>14.9%</td>
<td>12.7%</td>
<td>11.1%</td>
<td>9.8%</td>
<td>8.8%</td>
<td>8.0%</td>
</tr>
</tbody>
</table>

Table 4.5: Average relative error reached by the upper bound on the cycle time, using PMF without instant jobs, with $a$ and $\alpha$ changing.
The bounds are not really tight. Their accuracy is on the order of ten percent. The gap between both bounds is directly proportional to the step size $\tau$ (equal with PMF/IJ).

However, the tightness of the bounds improves when the number of discretization steps increases, i.e. when the PMF discretization refines. Theoretically, any accuracy could be reached, as it is directly proportional to the size $\tau$ of the intervals. However, the tightness is limited by the computational complexity. Note that the evolution of the upper and lower bounds is also illustrated on the example of a three station tandem queue in Section 5.1.3. It can be seen on Figure 5.1 that the bounds tighten, i.e. the gap between them regularly decreases, when the step size decreases.

Concerning the effect of the shift parameter $\alpha$, the bounds improve when $\alpha$ increases, except for the lower bound using PMF/nIJ (see tables 4.2, 4.3 and 4.5). A shift parameter $\alpha$ equal to one, i.e. aggregating the probability mass in the beginning of the interval, thus leads to the best bounds. This comes from the fact that the step size $\tau$, and thus the distance between both bounds, decreases when $\alpha$ increases. Moreover, the effect of $\alpha$ appears to be stronger on the lower bound. These observations are consistent with those made on the simple example studied in the previous section (see Figure 4.4). When $\alpha$ is chosen in order to conserve the expectation of the distribution, the behavior of the bound is very similar to the case where $\alpha/\tau = 0.5$, unsurprisingly.

The lower bound computed with PMF without instant job is less tight than the others (see Table 4.4). This property is obvious from Corollary 4.15. It is due to the larger size of the first interval on which the probability masses are computed, to avoid instant jobs. In general, the bounds computed using PMF with instant jobs are thus better.

It can also be seen that the first lines of the tables are equal for the lower bounds (tables 4.2 and 4.4) and for the upper bounds (tables 4.3 and 4.5), respectively. This comes from the fact that the case where $\alpha = 0$, i.e. grouping at the end, is the same for PMF/IJ and PMF/nIJ. This is easily checked from the definition of PMF/IJ and PMF/nIJ, or from figures 2.3 and 2.6.

It is also interesting to note that, with PMF/IJ, the middle lines ($\alpha/\tau = 0.5$) are very similar. As we have $c_\alpha - (\tau - \alpha) \leq c \leq c_\alpha + \alpha$, it tends to show that $c_\alpha$, which is right in the middle of the interval between both bounds, could be a good approximation for the cycle time $c$.

Tables 4.2 to 4.5 illustrate the effect of the shift parameter $\alpha$ and of the presence (or not) of instant jobs on the behavior of the bounds. Table 4.6 shows the effect of the buffer sizes on the behavior of the upper bound computed with PMF/nIJ (the other cases are similar). The table shows that the tightness of the bounds deteriorates when the buffer sizes increase. This observation is easily
4.2. **BOUNDS FROM THE DISCRETIZED SYSTEM**

\[ \sum b_i = 5 \]
\[ a = 6 \]
\[ a = 7 \]
\[ a = 8 \]
\[ a = 9 \]
\[ a = 10 \]

<table>
<thead>
<tr>
<th>[ \sum b_i ]</th>
<th>[ a = 5 ]</th>
<th>[ a = 6 ]</th>
<th>[ a = 7 ]</th>
<th>[ a = 8 ]</th>
<th>[ a = 9 ]</th>
<th>[ a = 10 ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>13.4%</td>
<td>11.4%</td>
<td>9.9%</td>
<td>8.7%</td>
<td>7.8%</td>
<td>7.1%</td>
</tr>
<tr>
<td>1</td>
<td>14.3%</td>
<td>12.1%</td>
<td>10.5%</td>
<td>9.3%</td>
<td>8.3%</td>
<td>7.5%</td>
</tr>
<tr>
<td>2</td>
<td>15.9%</td>
<td>13.4%</td>
<td>11.6%</td>
<td>10.2%</td>
<td>9.2%</td>
<td>8.3%</td>
</tr>
<tr>
<td>3</td>
<td>16.4%</td>
<td>13.8%</td>
<td>11.9%</td>
<td>10.5%</td>
<td>9.4%</td>
<td>8.5%</td>
</tr>
<tr>
<td>4</td>
<td>16.6%</td>
<td>14.0%</td>
<td>12.1%</td>
<td>10.6%</td>
<td>9.5%</td>
<td>8.6%</td>
</tr>
</tbody>
</table>

Table 4.6: Average relative error reached by the upper bound on the cycle time, using PMF without instant jobs \((\alpha/\tau = 0.5)\), with \(a\) changing and for various total storage spaces.

understood from the following: when the storage space increases, the cycle time decreases, while the absolute error made by the bound tends to be constant (directly proportional to the step size \(\tau\)), the relative error thus increases.

One parameter has not yet been considered: the **configuration** of the network. The previous experiments focused on three station tandem queues, aiming to illustrate the effect of the shift parameter \(\alpha\), the instant jobs and the buffer sizes. We now study a broader range of configurations, but with a smaller scope of parameters. We analyse tandem queues with two, three or four stations, and fork and join networks with three or four stations, i.e. 9 different network topologies\(^6\). The global storage space of a network goes from zero to four and is balanced among the buffers. The service time distributions are chosen randomly among the ten distributions listed earlier. In total, 250 networks (5 total storage space, 50 distribution mixes) of each of the 9 topologies were analysed, making 2250 networks. Concerning the method parameters, the number \(a\) of discretization steps equals 4, 6 or 8 and PMF without instant jobs, with \(\alpha/\tau = 0.5\), is used. The results were compared to simulation results and the average relative errors are given in Table 4.7.

First, we see on Table 4.7 that the tightness of the bound improves when the number \(m\) of stations increases. Similarly to the effect of buffer sizes, this comes from the fact that, when the number of stations increases, the cycle time increases, while the error made by the bound tends to be constant. Second, Table 4.7 shows that the tightness of the bound is very similar for all configurations (tandem, fork, join). This comes from the fact that, in any case, the bound error is directly proportional to the time step \(\tau\). Moreover, for the networks studied here (up to four stations), there is never more than one way between two stations, so that \(\kappa \leq 1\) (see the proof of Proposition 4.4). The factor \(\kappa\) can thus be removed from the bounds given in corollaries 4.15 and 4.16, similarly to the case of tandem queues.

\(^6\)For fork and join networks with four stations, we consider two possibilities. For fork networks, the fourth station can precede the disassembly station (with two subsequent stations), or come after one of the following stations. For join networks, the fourth station can follow the assembly station (with two preceding stations), or precede one of the two preceding stations.
4.3 Bound from the Critical Path Computation

In this section, we propose a second bounding methodology (which is also presented in Tancrez et al. [82]). It allows to compute a tight lower bound on the cycle time of fork-join queueing networks with blocking, and with general service time distributions. We show that we can take advantage of the critical path computation (see Section 4.1.4) in order to find a tighter lower bound on the cycle time of a queueing network. We could simply compute $|cp(\tilde{r}_P)|$ rather than using the upper bound on it, i.e. $(\kappa_P + \eta - 1)$ (see proof of Lemma 4.8). This would offer a slightly better lower bound. On the contrary, such a tighter upper bound is unreachable as it would require to compute $|cp(r_P)|$. However, in order to get a better lower bound, a far better option is available, as we show below. Note that, in this subsection, the subscript $\alpha$ (and not $\alpha'$) is used but the results are valid for the PMF with instant jobs as well as for the PMF without instant job.

4.3.1 Bounding Methodology

The core idea of the bounding methodology comes from the first property given in Section 4.1.3. The sequence of jobs which constitutes the critical path in the discretized time form a non-overlapping path in the original time, thus shorter than the critical path in the original time, and thus shorter than the length of the original sample run. The critical path in the discretized time thus leads to a lower bound on the original running time, and, importantly, this bound can be computed (see Section 4.1.4). Moreover, it can be supposed that this non-overlapping path is close to the critical path in the original time, i.e. the bound is tight.

Before rigorously proving this result, let us illustrate it on Figure 4.5. The left-hand side shows a run $r$ in the original time and the right-hand side depicts the corresponding discretized run $r_\alpha$. On the left hand-side, the critical path is colored in light gray. Its length gives the real time to serve three units. On the right-hand side, the critical path is colored in dark gray. The corresponding non-overlapping path is given in dark gray on the left-hand side. It can be seen that it offers a lower bound on the real time to serve three units. In other words, in the original time (left), the dark sequence of jobs is smaller than the light critical path. Moreover, it is reasonable to think that the critical path in the discretized time leads to a non-overlapping path (dark sequence) which is close to the critical path in the original time (light sequence), and thus to a tight lower bound. In the next
4.3. BOUND FROM THE CRITICAL PATH COMPUTATION

Figure 4.5: Critical paths and corresponding non-overlapping paths, in an original run (left) and in the discretized run (right, with \( \alpha = 0 \)), for a tandem queue made of four stations.

proposition, we show that the length of the non-overlapping path in the original time can be computed, and thus, that a good lower bound can be computed. The proof relies on two ideas. First, the probability for a job of the dark sequence to be on a given station and to have its length in a given interval can be deduced from the discrete critical path computation. Second, the expected length of such a job (knowing its discretized length) is independent of the fact that it belongs to the dark sequence and can thus be computed.

**Proposition 4.17. (Lower bound by the critical path computation)** The modelling method, and the critical path computation, allows to compute the following lower bound on the cycle time \( c \) of a fork-join queueing network:

\[
c \geq \frac{c_\alpha}{l_\alpha(w_{cp})} \sum_{i=1}^{m} \sum_{j=1}^{a} P[l_\alpha w_{i,k} = j\tau \mid w_{i,k} \in cp(r_\alpha)] \cdot E[l(w_{i,k}) \mid l_\alpha w_{i,k} = j\tau].
\]

(4.21)

**Proof.** Let us suppose that we choose a given \( w_{i,k} \) in the discrete time (right-hand side of Figure 4.5) and that its length equals \( j\tau \). The only thing we know about the original length of \( w_{i,k} \) (left-hand side of Figure 4.5) is that it lies in the interval for which the probability mass is aggregated on \( j\tau \), i.e., between \((j-1)\tau + \alpha\) (0 if \( j = 1 \)) and \( j\tau + \alpha \). If \( w_{i,k} \) is chosen in the discretized time, independently of its length in the original time, we have no clue about the position of the original length in the interval. The original length (known to be in a given interval \([ (j-1)\tau + \alpha, j\tau + \alpha ] \)) of a job which is chosen because it belongs to the critical path in the discretized time is thus independent of the fact that it belongs to this critical path \( cp(r_\alpha) \). We have:

\[
E[l(w_{i,k}) \mid l_\alpha w_{i,k} = j\tau \& w_{i,k} \in cp(r_\alpha)] = E[l(w_{i,k}) \mid l_\alpha w_{i,k} = j\tau].
\]

Moreover using Proposition 4.3, the right-hand side term of inequality (4.21) can be rewritten as follows:

\[
\lim_{n' \to -\infty} \frac{|cp(r_\alpha)|}{n'} \sum_{i=1}^{m} \sum_{j=1}^{a} P[l_\alpha w_{i,k} = j\tau \mid w_{i,k} \in cp(r_\alpha)]
\]

\[
\cdot E[l(w_{i,k}) \mid l_\alpha w_{i,k} = j\tau \& w_{i,k} \in cp(r_\alpha)]
\]

\[
= \lim_{n' \to -\infty} \frac{|cp(r_\alpha)|}{n'} E[l(w_{i,k}) \mid w_{i,k} \in cp(r_\alpha)].
\]

The expression \( |cp(r_\alpha)| \cdot E[l(w_{i,k}) \mid w_{i,k} \in cp(r_\alpha)] \) gives the length, in the original time, of the sequence of jobs making the critical path in the discrete time, i.e. the dark sequence on the
left-hand side of Figure 4.5. As it is only a non-overlapping path in the original time, it is
shorter than the critical path (in light gray), and thus than the global real time to serve $n^r$
units. Formally, this can be written as follows, and it completes the proof.

$$\lim_{n^r \to \infty} \frac{|cp(r_\alpha)|}{n^r} E[l(w_{i,k}) \mid w_{i,k} \in cp(r_\alpha)] \leq \lim_{n^r \to \infty} \frac{|cp(r)|}{n^r} E[l(w_{i,k}) \mid w_{i,k} \in cp(r)] = c.$$ 

It is essential to see that this lower bound is computable. Each term can
be computed. The two terms implying the critical path $cp(r_\alpha)$ ($l_\alpha(w_{cp})$ and $P[l_\alpha(w_{i,k}) = j\tau \mid w_{i,k} \in cp(r_\alpha)]$) are easily inferred from the critical path computation, i.e. from the probabilities $p^\alpha_{l,s}(i, s)$ (see Section 4.1.4). The cycle time $c_\alpha$ is
deduced from the steady-state probabilities of the states of the Markov chain (see
Chapter 3). Finally, $E[l(w_{i,k}) \mid l_\alpha(w_{i,k}) = j\tau]$ can be computed from the original
service time distributions, which are known.

It can be argued that the proposed lower bound is tight because the critical
path does not differ much from the discretized to the original time. Moreover,
when the number of discretization steps $\alpha$ increases, the length of a discretized job
becomes closer to the length of the original job. Consequently, the lower bound
becomes tighter when the PMF discretization is refined. At the limit, when $\alpha$ goes
to infinity, the discretized time tends to be equivalent to the original time, and
the bound thus converges to the exact cycle time.

Moreover, it is reasonable to think that this new lower bound brings a sig-
nificant improvement compared to the lower bounds proposed in corollaries 4.15
and 4.16. Informally said, the new lower bound computes the dark sequence in
the left hand side of Figure 4.5 (original time), while the previous lower bound
computes the dark sequence in the right hand side part of the figure (discretized
time), and then removes a correction constant to each job in order to be sure that
every corrected job is smaller than the original job. Obviously, this constant is
frequently too large. In other words, the corrected job is frequently much smaller
than the original job in the dark sequence on the left hand side, leading to a less
good lower bound.

However, the critical path computation suffers from a high complexity. As
explained in Section 4.1.4, the critical path computation is more expensive than
the basic method. The new lower bound is thus more expensive to compute than
the previous one. Nevertheless, our experiments show that the improvement is
clearly worth the additional cost. At constant computational cost, we can choose
a smaller $\alpha$, i.e. less discrete values, and get a more accurate lower bound. The
tightness of the new lower bound is shown and discussed in the next section.

This new lower bound is another interesting contribution. It can be seen as
a second bounding methodology. It highlights two originalities of our research.
First, the critical path, and its computation, is obviously essential here. Second,
the lower bound also relies on the idea of probability mass fitting as it uses the
fact that the probability mass in an interval is aggregated in one value $j\tau$. The
extension of this bounding methodology to split and merge networks is discussed
in Section 4.4.
### 4.3. Bound from the Critical Path Computation

The shift parameter $\alpha$ and the number $a$ of discretization steps vary.

#### 4.3.2 Computational Experiments

In order to assess the tightness of the bound computed by the second bounding methodology, and to study its behavior, we compared it to simulation results. Various networks configurations have been tested: tandem, fork and join, with two, three, or four stations (various possibilities in this case). The global storage space of a network goes from zero to four and is supposed to be balanced among the buffers. Each configuration has been tested with various service time distributions, arbitrary chosen among the ten distributions previously mentioned: uniform(0,1), beta(1.3,1), beta(2,2), beta(5.5,6), beta(8,8), beta(10,9), triangular(0,1,0.5), triangular(0.2,1,0.3) and triangular(0.1,0.9,0.6) (see Figure 2.5).

In total, 900 FJQN have been analyzed. Moreover, the parameters of the modelling method have also been varied aiming to understand their respective influence: the number $a$ of discretization steps (4, 6 and 8 steps) and the shift parameter $\alpha$ ($\alpha/\tau = 0, 0.25, 0.5, 0.75, 1$). In total, we made 13500 experiments. Tables 4.8 to 4.10 illustrate the levels of accuracy reached by the lower bound on the cycle time. They give the average relative error, in percent, between the bound and the result of the simulation (i.e. $(\text{sim} - \text{bound})/\text{sim}$).

Table 4.8 gives the average relative errors obtained for all the network configurations (tandem, fork, join, with various numbers of stations and various storage spaces). First of all, it shows the tightness of the bound. On average, the level of accuracy reached is 0.4% with eight discretization steps, 0.7% with six and 1.4% with only four steps. We remind that this accuracy, as well as the bound characteristic, is valid for general distributions, i.e. for the global modelling process, including the building of tractable distributions. In Table 4.8, the influence of parameters $a$ and $\alpha$ is also illustrated. It can be seen that the tightness of the bound increases significantly when the discretization is refined. This is not surprising as the approximation of the distributions improves, and so does the approximation of the critical path. The shift parameter $\alpha$ has less impact. In the next tables, we choose to focus on the case where the mass is grouped in the middle of the interval ($\alpha/\tau = 0.5$), which is the most natural. Other parameters $\alpha$ lead to the same behavior.

Table 4.9 aims to illustrate the influence of the topology of the network on the tightness of the bound. It can be seen that the accuracy reached for tandem, fork or join networks is very similar. The number of stations composing the network has a negative influence. Nevertheless, the bound stays tight for larger networks.

Table 4.10 shows that the bound tends to tighten when the storage space of the network is increased. In summary, concerning the configuration of the network,
Table 4.9: Average bound tightness reached for various FJQN network topologies, with various numbers of stations $m$. The number $a$ of discretization steps varies, and $\alpha/\tau = 0.5$.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$m = 2$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.83%</td>
<td>1.36%</td>
<td>1.64%</td>
<td>1.27%</td>
<td>1.55%</td>
<td>1.30%</td>
<td>1.62%</td>
</tr>
<tr>
<td>6</td>
<td>0.38%</td>
<td>0.61%</td>
<td>0.80%</td>
<td>0.61%</td>
<td>0.75%</td>
<td>0.64%</td>
<td>0.79%</td>
</tr>
<tr>
<td>8</td>
<td>0.21%</td>
<td>0.36%</td>
<td>0.51%</td>
<td>0.36%</td>
<td>0.48%</td>
<td>0.38%</td>
<td>0.51%</td>
</tr>
</tbody>
</table>

Table 4.10: Average bound tightness reached for various storage spaces ($B_{\Sigma}$ stands for $\sum b(i,j)$). The number $a$ of discretization steps varies, and $\alpha/\tau = 0.5$.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$B_{\Sigma} = 0$</th>
<th>$B_{\Sigma} = 1$</th>
<th>$B_{\Sigma} = 2$</th>
<th>$B_{\Sigma} = 3$</th>
<th>$B_{\Sigma} = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.98%</td>
<td>1.78%</td>
<td>1.38%</td>
<td>1.03%</td>
<td>0.87%</td>
</tr>
<tr>
<td>6</td>
<td>0.98%</td>
<td>0.87%</td>
<td>0.66%</td>
<td>0.49%</td>
<td>0.40%</td>
</tr>
<tr>
<td>8</td>
<td>0.58%</td>
<td>0.51%</td>
<td>0.39%</td>
<td>0.29%</td>
<td>0.17%</td>
</tr>
</tbody>
</table>

while the topology does not seem to matter, the tightness of the bound seems to deteriorate when the number of stations increases, and it seems to improve when the storage space increases.

The tightness reached with this second bounding methodology, taking advantage of the critical path computation, is clearly better, by one order of magnitude, than the one reached with the first bounding methodology, computed in the discretized system (see Section 4.2). Tables 4.7 and 4.9 can be compared for example (see tables 3.1 and 4.1 for the computational cost). For three station tandem queues (with various buffer sizes), with $a = 8$ and $\alpha/\tau = 0.5$, the average relative error equals 9.9% (see Table 4.2). This can be compared to the 0.36% tightness reached with the present bound (see Table 4.9). The additional computational cost due to the computation of the critical path is clearly worthwhile (1.7 compared to 0.3 seconds). At equal computational cost (a smaller for the second methodology), the accuracy of the bound taking advantage of the critical path is clearly better. For example, for four station tandem queues, the first bounding methodology leads to an average relative error of 24.5% (14.5% using PMF/IJ) in 1.2 second ($a = 6$, see Table 3.1), while the second bounding methodology leads to an error of 1.64% in 1 second ($a = 6$, see tables 4.9 and 4.1). These observations confirm the informal argument given in Section 4.3.1, concerning the quality of the second bounding methodology. Moreover, note that the upper bound (by the first methodology) can be computed without additional cost.

Ideally, the results obtained by our bounding methodologies should be compared to those of concurrent methods in the literature. However, as explained in the introduction (Section 1.3.4), few bounding methods exist for networks with general service time distributions. The effect of the distribution fitting on the global modelling process is poorly studied, and bounds exist essentially for queue-
ing networks with phase-type distributions (often exponential). The contribution of Baccelli and Makowski [12] is mainly theoretical. To the best of our knowledge, the accuracy of the bounds has not been investigated. Nakade [63] proposed upper and lower bounds on the cycle time of tandem queues with general service time distributions and zero buffer sizes. On some numerical examples, Nakade [63] shows that “the bounds give at most 8% relative errors”, and compare favorably to the bounds shown in van Dijk and Lamond [94] and Shanthikumar and Jafari [76] (for systems with exponentially distributed service times). On systems with blocking-after-service and uniform distributions, Nakade [63] gets a 5.6% average relative error for tandem queues with zero buffer sizes. The second bounding methodology thus compares favorably in terms of accuracy (less than 1%) and both bounding methodologies compare favorably in terms of the scope of application (more configurations and buffer sizes).

4.4 Split-and-Merge QN

In the previous sections, we described two new methodologies to compute bounds on the cycle time of fork-join queueing networks. However, the bounding methodologies could be applied or extended to other queueing networks. In this section, we are interested in split-and-merge queueing networks with blocking (presented in Section 1.2.2). The bounds mainly rely two ideas: the probability mass fitting and the critical path. Obviously, the former is independent of the network configuration. The latter can be extended to many networks as its existence basically follows from the fact that a job start is always triggered by another job end.

The existence of the critical path relies on the synchronization constraints of the queueing network. We give them for split-and-merge queueing networks in the next lemma. For this, we first need some notations. As already said, each job is given an index \( k = 1, \ldots \). However, in split-and-merge QN, every job does not pass through every station (but jobs do not pass twice by the same station, as we assume acyclic networks). We thus need a way to consider the sequence of jobs passing by a given station. We denote \( \chi_i(k, l) \) the index of the job which is \( l \) ranks before job \( k \) in station \( i \). It means that \( \chi_i(k, 1) \) is the index of the job which directly precedes job \( k \) on station \( i \), that \( \chi_i(k, 2) \) directly precedes \( \chi_i(k, 1) \), and so on. Moreover, we denote \( \psi(i, k) \) the station to which item \( k \) on station \( i \) is routed.

The synchronization constraints of split-and-merge QN are similar to those given in Lemma 4.2 for fork-join QN. However, in a SMQN, a merge station needs only one unit from one of its predecessor to begin (see Figure 4.6), as opposed to a join station which needs units from all its predecessors ((4.7) becomes (4.22)). Moreover, a split station just needs one succeeding buffer to have some free space, i.e. the buffer in front of the station to which the unit is routed (see Figure 4.6). This is different from a fork station which needs space in every succeeding buffer (compare (4.9) to (4.24)).
Lemma 4.18. (Synchronization constraints of a SMQN) Given a split-and-merge queuing network, the moment a job \( w_{i,k} \) starts, in a sample run \( r \), is given by the following equation:

\[
t^r_{\text{start}}(w_{i,k}) = \max \left[ t^r_{\text{end}}(w_{j,k}) \mathrel{|} \psi(j,k) = i, \right. \\
\left. t^r_{\text{end}}(w_{i,\chi_i(k,1)}), \right. \\
\left. t^r_{\text{start}}(w_{\psi(i,\chi_i(k,1),\xi(i,k)}) \right] .
\]  

(4.22)  

Where \( \xi(i,k) \) is the index of the job whose start unblocks job \( k \) from station \( i \), 

\[
\xi(i,k) = \chi_{\psi(i,\chi_i(k,1))}(\chi_i(k,1), \psi(i, \chi_i(k,1))).
\]  

Moreover, the starting time of a job always equals the ending time of another job, as we have:

\[
t^r_{\text{start}}(w_{\psi(i,\chi_i(k,1),\xi(i,k)}) = \max \left[ t^r_{\text{end}}(w_{\psi(i,\chi_i(k,1),\xi(i,k)}) \mathrel{|} \psi(i, \chi_i(k,1)), \right. \\
\left. t^r_{\text{end}}(w_{\psi^2(i,k),\chi_{\psi^2(i,k)}}(\xi^2(i,k),1)) \right], \\
\left. \ldots \right] .
\]  

(4.25)  

(4.26)  

(4.27)  

Where \( \psi^2(i,k) \) is the station to which item \( \xi(i,k) \) on station \( \psi(i, \chi_i(k,1)) \) is routed, and \( \xi^2(i,k) \) is the index of the job whose start unblocks job \( \xi(i,k) \) from station \( \psi(i, \chi_i(k,1)) \),

\[
\psi^2(i,k) = \psi(i, \chi_i(k,1), 1), \\
\xi^2(i,k) = \chi_{\psi^2(i,k)}(\chi_{\psi^2(i,k)}(\xi(i,k), 1), \psi^2(i,k)).
\]

Proof. Each term corresponds to one condition, and to one state (starved, working, or blocked) preceding \( w_{i,k} \) on station \( i \).

(4.22) Job \( w_{i,k} \) cannot start before the right preceding station \( j \in E(i) \) with \( \psi(j,k) = i \) passed job \( k \) to station \( i \). If this term corresponds to the maximum, it means station \( i \) was starved before beginning to serve unit \( k \).

(4.23) The station \( i \) cannot begin to serve unit \( k \) before it finished serving the previous unit \( \chi_i(k,1) \). If \( t^r_{\text{start}}(w_{i,k}) = t^r_{\text{end}}(w_{i,\chi_i(k,1)}) \), it means station \( i \) was working just before starting job \( w_{i,k} \).

(4.24) In fact, before starting \( w_{i,k} \), station \( i \) should not only finish to serve job \( w_{i,\chi_i(k,1)} \) but should also get rid of unit \( \chi_i(k,1) \). For this, there should be some room left in the right following buffer (of station \( \psi(i, \chi_i(k,1)) \)). This is not the case (and station \( i \) gets blocked) if units \( \psi(i, \chi_i(k,1)) \) to \( \chi_{\psi(i,k)}(\chi_i(k,1), 1), \psi(i, \chi_i(k,1))) \) are waiting in this buffer. Indeed, in this case, the buffer is full. So, \( w_{i,k} \) can only start when job \( \chi_{\psi(i, \chi_i(k,1))}(\chi_i(k,1), \psi(i, \chi_i(k,1))) \) began on station \( \psi(i,k) \). In this case, it means station \( i \) was blocked before beginning to serve unit \( k \).
4.4. SPLIT-AND-MERGE QN

Once all the above conditions are satisfied, there is no reason to wait (and $t'_{\text{start}}(w_{i,k})$ thus equals the maximum). Moreover, equation (4.22-4.24) gathers all conditions as every possible state preceding $w_{i,k}$ is considered (starved, working, or blocked).

The second part of the lemma can be directly deduced from equation (4.22-4.24). Job $\xi(i,k)$ can only start on station $\psi(i, \chi_{i}(k,1))$ if the previous job $\chi_{\psi(i, \chi_{i}(k,1))}(\xi(i,k),1)$ finished (leading to term (4.25)), and if there is room left in the next buffers for station $\psi(i, \chi_{i}(k,1))$ to get rid of it (4.26). The last condition means that $\xi^2(i,k) = \chi_{\psi(i, \chi_{i}(k,1))}(\xi^1(i,k), 1)$, i.e. the job whose start unblocks job $\xi(i,k)$ from station $\psi(i, \chi_{i}(k,1))$, should have started on $\psi^2(i,k)$ the right successor of $\psi(i, \chi_{i}(k,1))$, which means the preceding job $\chi_{\psi^2(i,k)}(\xi^2(i,k), 1)$ should have ended. Similar conditions can then be found for the following stations.

From these synchronization constraints, it is clear that a critical path, i.e. the sequence of jobs that covers a run, can be found for split-and-merge QN. We can go over the properties of the critical path presented in Section 4.1.3 for fork-join QN to see if they stay true in the case of split-and-merge QN. We do not give the proofs for SMQN as they are extremely similar to those for FJQN. In fact, all properties except the first one (see below) extend. Proposition 4.3 (which allows to compute the ratio between the cardinality of the critical path and the number of units served) stays true as it is a simple consequence of the definition of the critical path. Proposition 4.4 (giving an upper bound on the cardinality of the critical path) also stays true as it is proved from the definition of the critical path and its way among the stations (going backward in time, it jumps to a previous station if the current station is currently starved, and to a next station if blocked). Moreover, for split-and-merge QN as for fork-join QN, the critical path can be built, going backward in time, thanks to the fact that the predecessor of a job in the critical path can be deduced from the state of the same station just before this job (see Section 4.1.2 and Figure 4.2).

However, the first property, saying that a sequence of job making the critical path in a given run will make a non-overlapping path in another run, is not true for split-and-merge QN. For fork-join QN, the synchronization constraints (Lemma 4.2) and the constant job ordering guarantee that no overlap could appear. This is not the case anymore for split-and-merge QN.

Let us review the three terms of the synchronization constraints, corresponding to the three possible states preceding the current job when building the critical path.

- If the station is previously working (terms (4.8) and (4.23)), the next critical path job will be on the same station. Obviously, no overlap can appear between two jobs on the same station, neither in FJQN nor in SMQN.

- If the station is previously blocked, the next critical path job will be on the unblocking successor (see Figure 4.1.c). In the FJQN case, if there is more than one successor, i.e. in a fork station, the unblocking station $j \in F(i)$ will be the last, among the blocking successors, to begin a new job (term (4.9)). In another run, another successor could be the last, but the station $j$ will definitely end before it (as, otherwise, the other station would not unblock), i.e. a gap may appear but no overlap. In the SMQN case, in a
split station, the unblocking station can only be the station to which the
blocked job is routed (term (4.24)). So, no overlap may appear between \( w_{i,k} \)
and the unblocking job \( w_{\psi(i,\chi_1(k,1)),\xi(i,k)} \) in another run (as the same job is
the unblocking job in any run).

- If the station is previously \textbf{starved}, the next critical path job will be on
  the unstarving predecessor. In the FJQN case, if there are more than one
  predecessor, i.e. in a join station, the unstarving station \( j \in E(i) \) will be the
  last, among the starving predecessors, to give the missing item to assemble
  (term (4.7)). In another run, another predecessor could be the last, but the
  station \( j \) will definitely end before it (as, otherwise, the other station would
  not unstarve), i.e. a gap may appear but no overlap. In the SMQN case,
  when a merge station is starved, it is unstarved by the \textit{first} predecessor
  \( j \) to put a unit in the buffer. In another run, another predecessor could end
  before and unstarve the merge station (and the ordering of the jobs is thus
  changed, what is impossible in FJQN). In this case, station \( j \) ends its job
  after the merge station starts and an overlap appears in this sequence of
  jobs. This is illustrated in Figure 4.7: the critical path in the discretized
  run (right-hand side) leads to an overlapping sequence of jobs in the original
  run (left-hand side). In the discretized run, the first station unstarves the
  third station while in the original run the second station unstarves the third
  station. The first station thus ends its job later on, and an overlap appears
  in the sequence of jobs.

Consequently, when \textbf{merge} stations are included in the network, the property
of the critical path does not hold anymore: the sequence of jobs which makes
the critical path in the discrete run is not necessarily non-overlapping in the original
run. Unfortunately, this property is essential for both bounding methodologies. It
is used in the proof of the first methodology, to pass from the original run to the
discretized run (see the proof of Lemma 4.8). Furthermore, the property is the
core idea of the second methodology: the critical path in the discretized run leads
to a non-overlapping, shorter, path in the original time and thus to a lower bound.
In conclusion, the bounds presented in sections 4.2 and 4.3.1 do not extend to
merge systems.
4.4. SPLIT-AND-MERGE QN

<table>
<thead>
<tr>
<th>a</th>
<th>First methodo. m = 3</th>
<th>First methodo. m = 4</th>
<th>Second methodo. m = 3</th>
<th>Second methodo. m = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>33.4%</td>
<td>30.9%</td>
<td>1.81%</td>
<td>2.32%</td>
</tr>
<tr>
<td>6</td>
<td>22.8%</td>
<td>21.2%</td>
<td>0.70%</td>
<td>0.98%</td>
</tr>
<tr>
<td>8</td>
<td>17.2%</td>
<td>16.1%</td>
<td>0.38%</td>
<td>0.57%</td>
</tr>
</tbody>
</table>

Table 4.11: Average relative error, in percent, reached by the upper bound by the first bounding methodology (left), and on the lower bound by the second bounding methodology (right), for the split configurations, with various numbers of stations $m$. The number $a$ of discretization steps varies, and PMF without instant jobs is used ($\alpha/\tau = 0.5$).

On the opposite, as can be seen from the above reasoning, this particular property extends to split networks (a station never has more than one predecessor), as well as the other properties (see Figure 4.6). The bounds, given in corollaries 4.15 and 4.16 and in Proposition 4.17, thus straightforwardly extend to split systems. They can be proved using exactly the same procedure as for fork-join queueing networks.

Similarly to the case of fork-join QN, we now test the tightness of the bounds for split systems. The bounds computed by the first bounding methodology (see corollaries 4.15 and 4.16) were tested with a set of experiments similar to the second one described in Section 4.2.2. We analyse split systems with three or four stations (two topologies in this case) and with a balanced storage space going from zero to four. The distributions are randomly chosen among the ten distributions previously mentioned (see Figure 2.5). Note that, in order to keep the network balanced, some distributions, for split stations notably, were shrunk to a domain which is half of the previous one ([0 0.5] instead of [0 1]). In total, we analysed 750 split systems. Table 4.11 present the average tightness of the upper bounds with 4, 6 or 8 discretization steps, $\alpha/\tau = 0.5$, and using PMF without instant jobs. It can be compared to Table 4.11. As for fork-join QN, the tightness of the bound improves when the discretization refines ($a$ increases) and when the number of stations increases. However, the accuracy of the upper bound for split systems is less good.

The bounds computed by the second bounding methodology (see Proposition 4.17) were tested with a set of experiments similar the one described in Section 4.3.2. We compute the tightness of the lower bound for 300 split configurations (three or four stations, storage space up to four, distributions randomly chosen in the previously described set). The results are summarized in Table 4.11. The average accuracy reached are similar, the bound is tight, but slightly less good than the one reached on FJQN (see Table 4.9). This can be explained by the fact that some distributions are shrunk and thus less finely discretized (these distributions are discretized to 3 values when $a = 6$ for example).
4.5 Other Performance Measures

In sections 4.2 and 4.3, we proposed two bounding methodologies, giving bounds on the cycle time. These bounds easily lead to bounds on other, related, performance measures. The basic idea here lies in the fact that the cycle time can be divided up into two components: the service time and the idle (blocked/starved) time. If \( w_i = E[l(w_{i,k})] \) denotes the expected service time on station \( i \), and \( bs_i \) the expected idle time on station \( i \), we have:

\[
c = w_i + bs_i, \quad \forall i.
\]

As the original expected service time is known, bounds on the cycle time easily lead to bounds on the blocked/starved time, on the utilization \( (U_i = w_i/c) \) and on the proportion of idle time \( (I_i = bs_i/c) \) of each station \( i \). These bounds are given in the following proposition.

**Proposition 4.19.** Let us suppose that the cycle time is bounded as follows:

\[
LB_c \leq c \leq UB_c.
\]

Then, the blocked/starved time, the utilization and the proportion of idle time can be bounded as follows:

\[
LB_c - w_i \leq bs_i \leq UB_c - w_i, \quad (4.30)
\]

\[
\frac{w_i}{UB_c} \leq U_i \leq \frac{w_i}{LB_c}, \quad (4.31)
\]

\[
\frac{LB_c - w_i}{UB_c} \leq I_i \leq \frac{UB_c - w_i}{LB_c}, \quad (4.32)
\]

\[
Proof. \text{ Inequality (}4.30\text{) follows straightforwardly from equations (}4.28\text{) and (}4.29\text{), (}4.31\text{) from (}4.29\text{) (}U_i = w_i/c\text{), and (}4.32\text{) from (}4.29\text{) and (}4.30\text{) (}I_i = bs_i/c\text{).}
\]

These bounds can be instantiated according to the bounds given in corollaries 4.15 and 4.16, or in Proposition 4.17. As for the cycle time, the bounds are refinable, i.e. they become tighter when the number of discrete values \( a \) is increased.

Valuable bounds on other performance measures such as the work-in-progress or the flow time is far more difficult to establish using our method, if not impossible. At least, we did not succeed in finding such bounds.

4.6 Monotonicity

In Section 4.2, we extended the bounding property of probability mass fitting (propositions 2.1 and 2.4) to the global modelling method, leading to bounds on the cycle time. Similarly, the monotonicity in \( \alpha \) of the discretized service times (propositions 2.2 and 2.5) can also be extended. It can be shown using the critical path as well but it is more easily shown as a corollary of a result given in Baccelli and Liu [11], which uses the fact that the discretized service times can be stochastically ordered, and follows the bounding methodology proposed in Baccelli and Makowski [12]. As previously, we focus here on fork-join queueing networks.
4.6. MONOTONICITY

Figure 4.8: The method is applied on an example, using PMF/IJ. The cycle time $c_{\alpha}$ is computed for various parameters $\alpha/\tau$ and with 5 (◇), 10 (○), 15 (□), or 20 (△) time steps. The horizontal line gives the cycle time estimated by simulation.

Proposition 4.20. Discretizing by PMF/IJ (PMF/nIJ) with two parameters $\alpha$ such that $\alpha_1 \leq \alpha_2$, the first expected time a FJQN takes to serve $P$ units, $T_{P,\alpha_1}$ ($T_{P,\alpha_1'}$), will always be larger than the second one, $T_{P,\alpha_2}$ ($T_{P,\alpha_2'}$), and similarly for the cycle times, $c_{\alpha_1}$ and $c_{\alpha_2}$ ($c_{\alpha_1'}$ and $c_{\alpha_2'}$). We have:

$$T_{P,\alpha_1} \geq T_{P,\alpha_2}, \quad \text{and} \quad c_{\alpha_1} \geq c_{\alpha_2},$$

$$T_{P,\alpha_1'} \geq T_{P,\alpha_2'}, \quad \text{and} \quad c_{\alpha_1'} \geq c_{\alpha_2'}.$$  

Proof. Both discretized random variables can be compared stochastically. Using PMF/IJ, Proposition 2.2 tells that $l_{\alpha_1}(w_{i,k})$ is always larger than $l_{\alpha_2}(w_{i,k})$. We thus have a strong stochastic ordering and, consequently, a stochastic convex increasing ordering:

$$l_{\alpha_1}(w_{i,k}) \geq l_{\alpha_2}(w_{i,k}), \quad \Rightarrow \quad l_{\alpha_1}(w_{i,k}) \succeq_{\text{st}} l_{\alpha_2}(w_{i,k}).$$

Consequently, and as fork-join queueing networks with finite buffers can be modeled by stochastic decision free Petri nets, Proposition 3.1 and Corollary 5.1 of Baccelli and Liu [11] apply and prove the PMF/IJ part of the proposition. The proof is exactly the same for the PMF/nIJ case. From Proposition 2.5, we have $l_{\alpha_1}(w_{i,k}) \succeq_{\text{st}} l_{\alpha_2}(w_{i,k})$ and the results of Baccelli and Liu [11] can thus be applied.

This result reveals the evolution of the throughput as a function of the shift parameter $\alpha$. When $\alpha$ is increased, the cycle time estimation $c_{\alpha}$ decreases. In Figure 4.8, we illustrate the evolution of the cycle time $c_{\alpha}$ in $\alpha$, for the example of a three station tandem queue with buffers of size one and service time distributions beta(2,2), uniform(0,1) and triangular(0,1,0.5). The decreasing characteristic can be clearly observed. Moreover, it can be seen that there exist an $\alpha$ for which the cycle time is exactly estimated. This alpha seems to be close to $\alpha/\tau = 0.5$ for any number of discretization steps.

Unfortunately, such a monotonic property cannot be proved on the bounds given by the first methodology (corollaries 4.15 and 4.16). However, an interesting
result comes from the comparison of the bounds computed using the two particular PMF discretizations presented in Section 2.1: grouping at the end and grouping at the beginning \((\alpha = 0 \text{ and } \alpha = \tau, \text{ also see Section 4.2.2})\). We compare them so that the number of discrete values (zero included) is the same, or, in other words, so that the step size \(\tau\) is the same. It is quite natural that the upper bound obtained using grouping at the end is better, while, consequently, the lower bound obtained using grouping at the beginning is better. We prove this in the next proposition, again using the critical path.

**Proposition 4.21.** The grouping at the beginning discretization (with \(a\) discretization steps) leads to a better lower bound, while the grouping at the end discretization (with \(a + 1\) discretization steps) leads to a better upper bound:

\[
T_P - \tau(\kappa P + \eta - 1) \leq \underline{T_P} \leq \overline{T_P} \leq T_P \leq \overline{T_P} + \tau(\kappa P + \eta - 1), \quad (4.33)
\]

\[
\overline{c} - \kappa \tau \leq \overline{c} \leq c \leq \overline{c} \leq \overline{c} + \kappa \tau. \quad (4.34)
\]

**Proof.** The inequalities \(\underline{T_P} \leq T_P \leq \overline{T_P}\) and \(\overline{c} \leq c \leq \overline{c}\) follow straightforwardly from corollaries 4.13 to 4.16. Obviously, for each equation (4.33) and (4.34), the two other inequalities are equivalent. We prove the first inequality for each. As \(l'(w_{i,k}) = l'(w_{i,k}) - \tau\) (the steps sizes \(\tau\) are equal and the service time is carried to its end or to its beginning, see figures 2.1 and 2.2) and as \(c_p(\tau_p)\) is just a non-overlapping path in the run discretized by grouping at the beginning, we get:

\[
l(\underline{\mathcal{L}}) \overset{(4.13)}{=} \sum_{w_{i,k} \in \mathcal{L}} l'(w_{i,k}) \geq \sum_{w_{i,k} \in \mathcal{L}} l'(w_{i,k}) = \sum_{w_{i,k} \in \mathcal{L}} (l'(w_{i,k}) - \tau) \geq l(\mathcal{L}) - \tau|c_p(\mathcal{L})| \overset{(4.14)}{=} l(\mathcal{L}) - \tau(\kappa P + \eta - 1).
\]

This inequality can be compared to Lemma 4.8. It can be extended to an average run, leading to a result similar to Proposition 4.9. As the PMF discretization simply aggregates the probability masses in intervals, and as these intervals are the same for grouping at the end (with \(a + 1\) discretization steps) and grouping at the beginning (with \(a\) discretization steps), we have \(P[\mathcal{R}] = P[\mathcal{R}]\), and get:

\[
\underline{T_P} = \sum_P P[\mathcal{L}]l(\mathcal{L}) \geq \sum_P P[\mathcal{L}](l(\mathcal{L}) - \tau(\kappa P + \eta - 1)) = T_P - \tau(\kappa P + \eta - 1).
\]

Dividing this inequality by \(P\) and making \(P \to \infty\), we get \(\tau - \tau \leq \overline{c}\) and this completes the proof.

Beside the main results, we gathered throughout the chapter clues about how the cycle time should be approximated. It seems that the middle of the interval between the upper and lower bounds (by the first methodology) is a good approximation of the cycle time (see Figure 4.4 and tables 4.2 and 4.3). Moreover, the PMF “grouping at the middle”, i.e. with \(\alpha/\tau = 0.5\), should lead to the more accurate estimations (see Figure 4.8 and tables 4.2 and 4.3). These clues are of interest in the next chapter, which focus on the approximate estimation of the performance measures.
Chapter 5
Performance Evaluation

In the previous chapter, we showed bounds on the cycle time of fork-join queueing networks and split networks. Once bounds are proved, the natural next step is to look for approximations of the performance measures. Our method, using probability mass fitting, offers a complete modelling of the queueing network, and clearly allows to evaluate its performance.

In Section 5.1, we focus on the cycle time evaluation, and propose some approximations. The other performance measures, the work-in-progress and the flow time in particular, are studied in Section 5.2. The accuracy of the performance evaluation is shown by extensive computational experiments. These two first sections evaluate the average performance. In Section 5.3, we show that our method also allows to approximate the distribution of the cycle time. Finally, in Section 5.4, we compare our method, and particularly probability mass fitting, to concurrent methods, essentially other distribution fittings. Note that, in this chapter, the subscript $\alpha$ (and not $\alpha'$) is used but the results are valid for the PMF with instant jobs as well as for the PMF without instant job.

5.1 Cycle Time Evaluation

In this section, we focus on the evaluation of the cycle time of queueing networks. The cycle time is the average time between two units leaving the system, i.e. the average interdeparture time. It equals the inverse of the throughput and reveals the “productivity” of the system. As such, it is one of the most important performance measures, if not the most important.

We first give two approximations which essentially aim to remove the bias of PMF (PMF does not conserve the mean of the service time). Then, we present a more subtle approximation which takes advantage of the critical path computation.
5.1.1 Simple Approximations

Definitions

To begin with, it is quite natural to think that PMF aggregating the probability masses in the middle of the interval ($\alpha/\tau = 0.5$) leads to the best approximation of the distributions, and should thus lead to the best approximations of the performance measures. Indeed, as already said (see Section 2.2.2), probability mass fitting does not conserve the mean of the distribution in general. It is reasonable to think that the bias is larger when the probability mass is aggregated on a discrete value far from the middle of the interval, and that removing this bias is a good idea. This is what is done by the first approximation proposed below. It can also be seen as the middle of the interval between the two bounds given in Corollary 4.16.

Approximation 5.1. The cycle time of a queueing network can be approximated by:

$$c \approx c_\alpha + \alpha - \tau/2.$$

A smarter approximation comes from the following. The cycle time can be divided up into two components: the average service time $w_i$ and the average idle time $bs_{\alpha,i}$ (see Section 4.5). In general, the distribution discretized by PMF does not have the same expectation as the original service time distribution. But the error on the average service time is known, as the expectations of both original and discretized distributions are known. A simple approximation of the cycle time can thus be obtained by removing the discretization error, made by the PMF, on the average service time. In other words, one can just add the estimation of the idle time to the known mean of the original service time distribution. The only remaining error made by this approximation is thus the error on the idle (blocked/starved) time.

Approximation 5.2. The cycle time of a queueing network can be approximated, for each station $i$, by:

$$c \approx w_i + bs_{\alpha,i},$$

with $w_i$ the mean of the original service time distribution of station $i$, $w_i = E[l(w_{i,k})]$, and $bs_{\alpha,i}$ the estimated idle time on station $i$.

The idle time can easily be estimated in the discretized system. The probability for a station $i$ to be idle at a given time is given by the sum of the stationary probabilities of all the states where the station is idle (blocked or starved). The idle time then equals this probability multiplied by the cycle time. Approximation 5.2 has the disadvantage that it depends on the station index $i$, unlike Approximation 5.1. Note that both approximations converge when the number of discrete values is increased, as they lie between the converging bounds.

Finally, note that one could think of other approximations. Approximation 5.1 could be refined to take into account the larger first interval in the case of PMF without instant job. Similarly to Approximation 5.2, one could easily devise an approximation which supposes that the proportion of idle time is conserved, instead of its absolute value. However, these approximations are more complicated.
and experiments reveal that they do not bring any benefit in terms of accuracy. In Tancrez et al. [81], we proposed another approximation, combining the two probability mass fitting introduced in Section 2.1, namely grouping at the end and at the beginning. The approximation consists in taking the average between the Approximation 5.2 for both PMF. It appears to offer accurate estimations (0.1% with 7 discretization steps, on an example) but they require to run the method twice (with both PMF).

Computational Experiments

We now show how approximations 5.1 and 5.2 behave and test their accuracy by extensive numerical experiments. To begin with, we focus on three station tandem queues, as we did for the bounds in Section 4.2. The goal is to study the impact of the shift parameter $\alpha$ and of the presence of instant jobs (or not). The experiments set is the same as in Section 4.2. It is made of an assortment of 500 three station tandem queues. The total storage space varies from zero up to four. The buffer configurations are balanced and equally shared out. The service time distributions are randomly chosen among the 10 distributions already listed: uniform(0,1), beta(1,3,1), beta(2,2), beta(4,4), beta(5.5,6), beta(8,8), beta(10,9), triangular(0,1,0.5), triangular(0.2,1,0.3) and triangular(0.1,0.9,0.6) (see Figure 2.5). The 500 tandem queues are analyzed using both PMF alternatives (with or without instant job). Five different shift parameters $\alpha$ are used ($\alpha/\tau = 0, 0.25, 0.5, 0.75, 1$) as well as, with PMF/IJ, the parameters $\alpha$ that conserves the expectations of the original distributions (see Proposition 2.3, noted “Exp. cons.” in the tables). The number $a$ of non-zero values in the discretized distributions varies from five up to ten. We thus made a total of 33000 experiments. In the following tables, we give the average relative errors, in percents, between the results of our method and the results of a simulation. Note that they basically reveal the error brought by the probability mass fitting (in the distribution fitting stage), as the analytical modelling, by a state model, is exact.

First, tables 5.1 to 5.4 show that our method leads to accurate estimations of the cycle time, at least for three station tandem queues. The error made by the method amounts to a few tenths of a percent. Of course, the accuracy improves when the number of discretization steps increases. A 0.1 percent accuracy is reached with $a = 10$, while a 0.3 percent accuracy is already obtained with five non-zero values in the discretized distribution. The best accuracy is reached by Approximation 5.2, using PMF/IJ and with $\alpha$ chosen to conserve the expectation of the service time distribution (see Table 5.3).

Furthermore, tables 5.1 to 5.4 show the effect of the shift parameter $\alpha$. Even if approximations 5.1 and 5.2 aim to correct the bias caused by PMF when the probability mass is not aggregated in the middle of the interval ($\alpha/\tau$ differs from 0.5), the best results are obtained with $\alpha/\tau = 0.5$ or when $\alpha$ is chosen to conserve the expectation. These choices lead to a better estimation of the cycle time, which means a better estimation of the idle time. This is quite natural: aggregating the probability masses in the middle of the interval offers a better approximation of the distribution. Among both choices, $\alpha/\tau = 0.5$ and $\alpha$ conserving the expectation, the improvement brought by the second is quite limited. Aggregating the proba-
### Table 5.1: Average relative error reached by Approximation 5.1 (middle of the interval) using PMF with instant jobs, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.69%</td>
<td>0.59%</td>
<td>0.48%</td>
<td>0.44%</td>
<td>0.38%</td>
<td>0.35%</td>
</tr>
<tr>
<td>0.25</td>
<td>0.41%</td>
<td>0.34%</td>
<td>0.28%</td>
<td>0.24%</td>
<td>0.21%</td>
<td>0.19%</td>
</tr>
<tr>
<td>0.5</td>
<td>0.32%</td>
<td>0.23%</td>
<td>0.17%</td>
<td>0.14%</td>
<td>0.12%</td>
<td>0.10%</td>
</tr>
<tr>
<td>0.75</td>
<td>0.43%</td>
<td>0.36%</td>
<td>0.29%</td>
<td>0.25%</td>
<td>0.23%</td>
<td>0.20%</td>
</tr>
<tr>
<td>1</td>
<td>0.73%</td>
<td>0.62%</td>
<td>0.52%</td>
<td>0.45%</td>
<td>0.40%</td>
<td>0.36%</td>
</tr>
</tbody>
</table>

### Table 5.2: Average relative error reached by Approximation 5.1 (middle of the interval) using PMF without instant job, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.63%</td>
<td>0.53%</td>
<td>0.46%</td>
<td>0.42%</td>
<td>0.37%</td>
<td>0.33%</td>
</tr>
<tr>
<td>0.25</td>
<td>0.35%</td>
<td>0.26%</td>
<td>0.22%</td>
<td>0.20%</td>
<td>0.18%</td>
<td>0.16%</td>
</tr>
<tr>
<td>0.5</td>
<td>0.31%</td>
<td>0.21%</td>
<td>0.16%</td>
<td>0.13%</td>
<td>0.11%</td>
<td>0.09%</td>
</tr>
<tr>
<td>0.75</td>
<td>0.64%</td>
<td>0.49%</td>
<td>0.38%</td>
<td>0.32%</td>
<td>0.28%</td>
<td>0.24%</td>
</tr>
<tr>
<td>1</td>
<td>1.03%</td>
<td>0.81%</td>
<td>0.65%</td>
<td>0.55%</td>
<td>0.48%</td>
<td>0.42%</td>
</tr>
</tbody>
</table>

### Table 5.3: Average relative error reached by Approximation 5.2 (idle time estimation) using PMF with instant jobs, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.63%</td>
<td>0.53%</td>
<td>0.46%</td>
<td>0.42%</td>
<td>0.37%</td>
<td>0.33%</td>
</tr>
<tr>
<td>0.25</td>
<td>0.52%</td>
<td>0.38%</td>
<td>0.31%</td>
<td>0.27%</td>
<td>0.23%</td>
<td>0.21%</td>
</tr>
<tr>
<td>0.5</td>
<td>0.57%</td>
<td>0.39%</td>
<td>0.30%</td>
<td>0.23%</td>
<td>0.19%</td>
<td>0.16%</td>
</tr>
<tr>
<td>0.75</td>
<td>0.86%</td>
<td>0.60%</td>
<td>0.47%</td>
<td>0.38%</td>
<td>0.31%</td>
<td>0.27%</td>
</tr>
<tr>
<td>1</td>
<td>1.21%</td>
<td>0.87%</td>
<td>0.68%</td>
<td>0.56%</td>
<td>0.48%</td>
<td>0.41%</td>
</tr>
</tbody>
</table>

### Table 5.4: Average relative error reached by Approximation 5.2 (idle time estimation) using PMF without instant job, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 5$</th>
<th>$a = 6$</th>
<th>$a = 7$</th>
<th>$a = 8$</th>
<th>$a = 9$</th>
<th>$a = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.63%</td>
<td>0.53%</td>
<td>0.46%</td>
<td>0.42%</td>
<td>0.37%</td>
<td>0.33%</td>
</tr>
<tr>
<td>0.25</td>
<td>0.52%</td>
<td>0.38%</td>
<td>0.31%</td>
<td>0.27%</td>
<td>0.23%</td>
<td>0.21%</td>
</tr>
<tr>
<td>0.5</td>
<td>0.57%</td>
<td>0.39%</td>
<td>0.30%</td>
<td>0.23%</td>
<td>0.19%</td>
<td>0.16%</td>
</tr>
<tr>
<td>0.75</td>
<td>0.86%</td>
<td>0.60%</td>
<td>0.47%</td>
<td>0.38%</td>
<td>0.31%</td>
<td>0.27%</td>
</tr>
<tr>
<td>1</td>
<td>1.21%</td>
<td>0.87%</td>
<td>0.68%</td>
<td>0.56%</td>
<td>0.48%</td>
<td>0.41%</td>
</tr>
</tbody>
</table>
5.1. CYCLE TIME EVALUATION

...bility masses in the middle of the interval brings very little bias. This is consistent
with Figure 4.8, where we observed that the parameter \( \alpha \) leading to the exact
cycle time estimation seems very close to \( \alpha/\tau = 0.5 \). The adequacy of \( \alpha/\tau = 0.5 \)
was also foreseen from tables 4.2 and 4.3. We noted that, with \( \alpha/\tau = 0.5 \), the
exact cycle time seems to be very close to the middle of the interval between both
bounds.

Concerning the presence of **instant jobs** (or not), it has little influence on
the accuracy of the approximations. It can be seen that, the larger parameter \( \alpha \),
the greater the influence of instant jobs. When \( \alpha = 0 \), i.e. grouping at the
end, PMF/IJ and PMF/nIJ are identical and when \( \alpha \) increases the first interval
becomes large using PMF/nIJ. Comparing the four tables 5.1 to 5.4, they are quite
similar. All in all, approximations 5.1 and 5.2 thus perform similarly and instant
jobs have little influence, with the notable exception of Approximation 5.2 using
PMF/nIJ. Indeed, it can be seen on Table 5.4 that this estimation of the cycle
time is less accurate.

In summary, the analysis of the set of three station tandem queues offers two
main conclusions. First, even on the corrected approximations, \( \alpha \) has a quite
strong influence, and \( \alpha/\tau = 0.5 \) or the expectation conservation are the best
choice. Second, the influence of the instant jobs and the difference between both
approximations is small, but Approximation 5.2 using PMF/nIJ is less good.

To further test the quality of the approximations, we study a larger set of
queueing networks, fork-join as well as split-and merge QN, with various topolo-
gies. The set of experiments has already been used in sections 4.2.2 and 4.4. We
analyse tandem queues with two, three or four stations, fork-join QN and split-
and-merge QN with three or four stations. This leads to 15 different network
topologies. The global storage space of a network goes from zero to four and is
balanced among the buffers. The service time distributions are chosen randomly
from the ten distributions listed earlier. In total, 250 networks (50 for each stor-
age space) of each of the 15 topologies were analysed, making 3750 networks. The
method was run with various parameters. The number \( a \) of discretization steps
equals 4, 6 or 8. Concerning the presence of instant jobs and the shift parameter,
we choose the best options as seen in the previous experiments set, and which
are quite natural options: PMF/IJ and PMF/nIJ with \( \alpha/\tau = 0.5 \), and PMF/IJ
conserving the expectation. The results were then compared to simulation results.
The obtained average relative errors are given in tables 5.5 to 5.11.

Tables 5.5 and 5.6 are similar to tables 5.1 to 5.4\(^1\), except that they give
the accuracy obtained for tandem queues with two, three or four stations (not
only three). The relative relative errors are quite similar but two options tend
to isolate as the best ones: Approximation 5.1 using PMF/nIJ with \( \alpha/\tau = 0.5 \)
and Approximation 5.2 using PMF/IJ with \( \alpha \) conserving the expectation. In
Section 3.1.2, we stretched out the fact that PMF with instant jobs leads to a
more complex modelling, and a larger and denser Markov chain (see Figure 3.4).
In terms of computational cost, the method using PMF with instant jobs is thus
sensibly more expensive, as illustrated on tables 3.1 and 3.3.

\(^1\)Table 5.15 in Section 5.1.2 also illustrates the impact of the shift parameter, for Approxima-
tion 5.1.
Table 5.5: Average relative error reached by Approximation 5.1 (middle of the interval) using PMF/nIJ or PMF/IJ, with \( a \) and \( \alpha \) changing, for tandem queues with 2, 3 or 4 stations.

<table>
<thead>
<tr>
<th>Method</th>
<th>( a = 4 )</th>
<th>( a = 6 )</th>
<th>( a = 8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMF/IJ, ( \alpha/\tau = 0.5 )</td>
<td>0.74%</td>
<td>0.31%</td>
<td>0.17%</td>
</tr>
<tr>
<td>PMF/IJ, Exp. Cons.</td>
<td>0.77%</td>
<td>0.31%</td>
<td>0.16%</td>
</tr>
<tr>
<td>PMF/nIJ, ( \alpha/\tau = 0.5 )</td>
<td>0.49%</td>
<td>0.21%</td>
<td>0.12%</td>
</tr>
</tbody>
</table>

Table 5.6: Average relative error reached by Approximation 5.2 (idle time estimation) using PMF/nIJ or PMF/IJ, with \( a \) and \( \alpha \) changing, for tandem queues with 2, 3 or 4 stations.

<table>
<thead>
<tr>
<th>Method</th>
<th>( a = 4 )</th>
<th>( a = 6 )</th>
<th>( a = 8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMF/IJ, ( \alpha/\tau = 0.5 )</td>
<td>0.75%</td>
<td>0.29%</td>
<td>0.16%</td>
</tr>
<tr>
<td>PMF/IJ, Exp. Cons.</td>
<td>0.50%</td>
<td>0.20%</td>
<td>0.12%</td>
</tr>
<tr>
<td>PMF/nIJ, ( \alpha/\tau = 0.5 )</td>
<td>0.86%</td>
<td>0.37%</td>
<td>0.22%</td>
</tr>
</tbody>
</table>

Table 5.7: Average relative error reached by Approximation 5.1 (middle of the interval) on fork-join QN and on split-and-merge QN, using PMF without instant job with \( \alpha/\tau = 0.5 \), the number of steps \( a \) changing.

<table>
<thead>
<tr>
<th>Method</th>
<th>( a = 4 )</th>
<th>( a = 6 )</th>
<th>( a = 8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>FJQN</td>
<td>0.48%</td>
<td>0.21%</td>
<td>0.13%</td>
</tr>
<tr>
<td>SMQN</td>
<td>1.66%</td>
<td>0.66%</td>
<td>0.33%</td>
</tr>
</tbody>
</table>

All in all, when the computational time is taken into account, as PMF without instant job can lead to similar accuracy, it appears clearly as the best choice. More precisely, from the observations made on tables 5.1 to 5.6, we may conclude that Approximation 5.1 using PMF/nIJ with \( \alpha/\tau = 0.5 \) is the best option. In other words, the natural choice is the good choice, i.e. the cycle time is directly measured on the system transformed by PMF/nIJ aggregating the probability masses in the middle of the intervals. In the following, we thus focus on this option. Note that the other approximations and parameter choices present similar behavior. Their choice can be justified in particular cases, for example when the prior objective is to get a bound as tight as possible (upper or lower).

Table 5.7 shows the average accuracy reached on all fork-join queueing networks and on all split-and-merge queueing networks (for any number of stations and any storage space). First of all, it shows the good accuracy of the approximation for both types of queueing network. For example, for FJQN, with only four discretization steps, the average level of accuracy is 0.5%. However, it appears...
5.1. CYCLE TIME EVALUATION

Table 5.8: Average relative error reached by Approximation 5.1 (middle of the interval) on fork-join QN, using PMF without instant job with $\alpha/\tau = 0.5$, the number of steps $a$ and the number of stations changing.

<table>
<thead>
<tr>
<th></th>
<th>Tandem</th>
<th>Fork</th>
<th>Join</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m = 2$</td>
<td>$m = 3$</td>
<td>$m = 4$</td>
</tr>
<tr>
<td>4</td>
<td>0.57%</td>
<td>0.47%</td>
<td>0.46%</td>
</tr>
<tr>
<td>6</td>
<td>0.22%</td>
<td>0.20%</td>
<td>0.21%</td>
</tr>
<tr>
<td>8</td>
<td>0.12%</td>
<td>0.12%</td>
<td>0.13%</td>
</tr>
</tbody>
</table>

Table 5.9: Average relative error reached by Approximation 5.1 (middle of the interval) on split-and-merge QN using PMF without instant job with $\alpha/\tau = 0.5$, the number of steps $a$ and the number of stations changing.

<table>
<thead>
<tr>
<th></th>
<th>Split</th>
<th>Merge</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$m = 3$</td>
<td>$m = 4$</td>
</tr>
<tr>
<td>4</td>
<td>1.55%</td>
<td>1.20%</td>
</tr>
<tr>
<td>6</td>
<td>0.56%</td>
<td>0.47%</td>
</tr>
<tr>
<td>8</td>
<td>0.28%</td>
<td>0.24%</td>
</tr>
</tbody>
</table>

clearly that the method performs worse when modelling SMQN. Roughly speaking, the average relative error is three times larger, but, even so, it stays quite low.

Tables 5.8 and 5.9 aim to illustrate the impact of the topology of the network and the impact of the number of stations. Concerning fork-join queueing networks (Table 5.8), the accuracy of the approximation reveals to be remarkably stable with the configuration and the number of stations. With eight discretization steps for example, the observed average relative error reveals to be nearby constant ($0.13\% \pm 0.01\%$). Again, the method behaves less nicely for split-and-merge networks. It can be seen on Table 5.9 that the approximation is significantly less accurate on merge systems compared to split systems. For each kind of topology, the accuracy seems to be quite stable in the number of stations.

We believe that the worse behavior of the method for SMQN is linked to the main disadvantage, as noticed by Bobbio et al. [18], of discrete phase-type distributions in distribution fitting: they generate coincident events (see Section 1.3.2). In merge networks in particular, coincident events have a significant effect. Indeed, with discrete service time distributions, two predecessors of merge station can end their job at the same time (this is impossible with continuous distributions). In this case, a priority rule has to be decided to choose which job will be the first in the buffer preceding the merge station (see the right-hand side of Figure 4.6).

---

2To check the robustness of the presented results (average relative errors), according to the simulation error and the distributions, we tested our method on another experiments set of 250 three station tandem queues. The computed average relative errors are very similar: 0.51\% 0.22\% 0.13\% with $a = 4, 6, 8$, which can be compared to the second column of Table 5.8. The impact of the service time distributions is further studied in the following.
When there is space for both job in the buffer, this has no influence on the cycle time. On the opposite, if, at the moment both jobs end, there is only space available for one item in the buffer, then the priority rule basically choose which preceding station will get rid of its job and which station will be blocked. In other words, the priority rule will lead the unprioritized station to be often blocked, or, more precisely, its blocking probability will be overestimated. In our model, we choose to apply the simplest rule: we give priority to the first station (in the sense of their numbering). More evolved priority rules could be considered and could improve the accuracy of the estimations for merge systems. For example, we could think of a priority rule which fifty-fifty chance to give priority to the first or the second station.

Tables 5.10 and 5.11 illustrate the impact of the storage space on the accuracy of the approximations, for fork-join and split-and-merge queueing networks. Concerning FJQN, Table 5.10 shows that the accuracy is also stable in the storage space. However, the accuracy of the approximation deteriorates for SMQN, as can be seen on Table 5.11.

Obviously, the approximations are better than the bounds in terms of accuracy. But, of course, the bounding property is lost in the approximations. Concerning the second bounding methodology, Tables 5.8 and 5.9, showing the impact of the topology, can be compared to tables 4.9 and 4.11. Table 5.10, showing the impact of the storage space, can be compared to Table 4.10. It clearly confirms that the approximation is more accurate than the bounds. However, for FJQN, the difference between the bound and the approximation tends to become less significant when the storage space increases. When the buffer sizes increase, the bound seems to become close to the approximation (0.17% compared to 0.13% with $a = 8$ and $\sum b(i, j) = 4$, see tables 4.10 and 5.10). In other words, the lower bound proposed

<table>
<thead>
<tr>
<th>$a$</th>
<th>$B_0$</th>
<th>$B_1$</th>
<th>$B_2$</th>
<th>$B_3$</th>
<th>$B_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.48%</td>
<td>0.41%</td>
<td>0.45%</td>
<td>0.53%</td>
<td>0.53%</td>
</tr>
<tr>
<td>6</td>
<td>0.23%</td>
<td>0.19%</td>
<td>0.20%</td>
<td>0.23%</td>
<td>0.23%</td>
</tr>
<tr>
<td>8</td>
<td>0.14%</td>
<td>0.12%</td>
<td>0.12%</td>
<td>0.14%</td>
<td>0.13%</td>
</tr>
</tbody>
</table>

Table 5.10: Average relative error reached by Approximation 5.1 (middle of the interval) on fork-join QN, using PMF without instant job with $a/\tau = 0.5$, the number of steps $a$ and the storage space changing ($B_\Sigma$ stands for $\sum b(i, j)$).

<table>
<thead>
<tr>
<th>$a$</th>
<th>$B_0$</th>
<th>$B_1$</th>
<th>$B_2$</th>
<th>$B_3$</th>
<th>$B_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.15%</td>
<td>1.41%</td>
<td>1.61%</td>
<td>1.97%</td>
<td>2.16%</td>
</tr>
<tr>
<td>6</td>
<td>0.50%</td>
<td>0.56%</td>
<td>0.66%</td>
<td>0.78%</td>
<td>0.79%</td>
</tr>
<tr>
<td>8</td>
<td>0.29%</td>
<td>0.28%</td>
<td>0.32%</td>
<td>0.39%</td>
<td>0.38%</td>
</tr>
</tbody>
</table>

Table 5.11: Average relative error reached by Approximation 5.1 (middle of the interval) on split-and-merge QN, using PMF without instant job with $a/\tau = 0.5$, the number of steps $a$ and the storage space changing ($B_\Sigma$ stands for $\sum b(i, j)$).
5.1. CYCLE TIME EVALUATION

Tandem Fork Join

<table>
<thead>
<tr>
<th>a</th>
<th>m = 2</th>
<th>m = 3</th>
<th>m = 4</th>
<th>m = 3</th>
<th>m = 4</th>
<th>m = 3</th>
<th>m = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.31%</td>
<td>0.24%</td>
<td>0.27%</td>
<td>0.24%</td>
<td>0.26%</td>
<td>0.25%</td>
<td>0.28%</td>
</tr>
<tr>
<td></td>
<td>1.26%</td>
<td>0.98%</td>
<td>1.01%</td>
<td>1.07%</td>
<td>1.09%</td>
<td>0.99%</td>
<td>1.10%</td>
</tr>
<tr>
<td>6</td>
<td>0.15%</td>
<td>0.12%</td>
<td>0.13%</td>
<td>0.12%</td>
<td>0.13%</td>
<td>0.13%</td>
<td>0.13%</td>
</tr>
<tr>
<td></td>
<td>0.57%</td>
<td>0.45%</td>
<td>0.51%</td>
<td>0.48%</td>
<td>0.55%</td>
<td>0.49%</td>
<td>0.54%</td>
</tr>
<tr>
<td>8</td>
<td>0.09%</td>
<td>0.08%</td>
<td>0.08%</td>
<td>0.07%</td>
<td>0.08%</td>
<td>0.08%</td>
<td>0.08%</td>
</tr>
<tr>
<td></td>
<td>0.34%</td>
<td>0.29%</td>
<td>0.33%</td>
<td>0.30%</td>
<td>0.33%</td>
<td>0.33%</td>
<td>0.33%</td>
</tr>
</tbody>
</table>

Table 5.12: Standard deviation of the relative error and maximum relative error (in italic), for Approximation 5.1 (middle of the interval) on fork-join QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the number of stations vary.

<table>
<thead>
<tr>
<th>a</th>
<th>m = 3</th>
<th>m = 4</th>
<th>m = 3</th>
<th>m = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.04%</td>
<td>0.77%</td>
<td>2.72%</td>
<td>1.86%</td>
</tr>
<tr>
<td>3.91%</td>
<td>2.98%</td>
<td>9.93%</td>
<td>6.84%</td>
<td></td>
</tr>
<tr>
<td>0.39%</td>
<td>0.30%</td>
<td>0.97%</td>
<td>0.58%</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.60%</td>
<td>1.23%</td>
<td>3.67%</td>
<td>2.48%</td>
</tr>
<tr>
<td>0.21%</td>
<td>0.17%</td>
<td>0.51%</td>
<td>0.30%</td>
<td></td>
</tr>
<tr>
<td>0.91%</td>
<td>0.70%</td>
<td>1.89%</td>
<td>1.34%</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.13: Standard deviation of the relative error and maximum relative error (in italic), for Approximation 5.1 (middle of the interval) on split-and-merge QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the number of stations vary.

in Section 4.3 reaches an “approximation like” accuracy level, particularly when the storage space increases. Note however that the complexity of the bound is higher (it requires the critical path computation).

Up to this point, we focused on average relative errors. However, it is also interesting to know how the accuracy varies from one experiment to the other, i.e. how the service time distributions impact the quality of the performance evaluation. As previously said, fifty distributions combinations (randomly chosen) were tested for each network configuration, i.e. with a given topology, number of stations and storage space. For each configuration, we computed the standard deviation of the relative error as well as the maximum error, in order to reflect the variability of the accuracy according to the service time distributions. Tables 5.12 and 5.13 give the average of these measures for a given topology and a given number of stations (but on various storage spaces). For fork-join queueing networks, the service time distribution has little effect, the accuracy varies very slightly when from one experiment to another (the standard deviation is small). Moreover, this is true independently of the number of stations and of the topology of the network.
(as well as of the storage space). With eight discretization steps for example, the maximum relative error is close to 0.33% while the average relative error is 0.13%. Again, the behavior of the method is less good for split-and-merge QN. Like the average error, the standard deviation and the maximum error are higher, and particularly for merge networks.

In summary, interesting observations can be drawn from the computational experiments. First, they reveal that the natural solution, consisting in the direct computation with PMF/nIJ and $\alpha/\tau = 0.5$, is the best choice. Then, the conclusions are quite distinct for fork-join queueing networks and for split-and-merge queueing networks.

**FJQN:** Most importantly, our method computes accurate estimations of the performance measures. Average relative errors are in the order of a few tenths of a percent. Even with six non-zero values in the discretized distributions, i.e. with six phases in the phase-type distributions, excellent accuracy levels are reached (0.2%). Moreover, the accuracy level reached by the estimations shows to be remarkably stable when the system’s configuration changes, i.e. in its topology, its number of stations, its buffers sizes, and its service time distributions.

**SMQN:** The method performs less well, notably because of the coincident events due to discrete service time distributions. However, the approximation is still accurate, particularly for split networks (0.5% for split, 1.25% for merge, with $a = 6$). The accuracy tends to deteriorate when the system becomes more complex (number of stations and storage space), except for split networks for which the number of stations has little impact. The accuracy is also more sensitive to the service time distributions (than for FJQN).

### 5.1.2 Approximation from the Critical Path Computation

In order to derive a more subtle approximation of the cycle time, we take advantage of the critical path computation (see Section 4.1.4). As previously said, in the previous approximations, the error on the service time is removed while the error on the idle time remains. We have no control on the error made on the idle time, while the error on the service time is fairly well known. Consequently, it would be advantageous to state the cycle time as a function of service times only, avoiding idle times. That is exactly what the critical path allows us to do (primarily for the global running time and then for the cycle time).

Let us come back to the Gantt chart of Figure 4.2. The probability mass fitting modifies each (gray) job of the critical path, by an amount which can be approximated by $\alpha - \tau/2$ (as in Approximation 5.1, the distribution of the lengths of the jobs in the critical path is not known). The modification of the global service time can thus be assessed. However, the number of jobs in the critical path (eight in the example) does not equal the number of jobs served during this run (eleven). To get an approximation of the cycle time, the critical path jobs error ($\alpha - \tau/2$) has thus to be multiplied by the ratio of the former numbers,
5.1. CYCLE TIME EVALUATION

which can be approximated by $c_\alpha / l_\alpha (w_{cp})$, as shown in Proposition 4.3. In the case given in Figure 4.2, where grouping at the end is used ($\alpha = 0$), we may write: $T_{11}/11 \approx T_{11}/11 - (8/11)(\tau/2)$.

**Approximation 5.3.** The cycle time of a queueing network can be approximated by:

$$c \approx c_\alpha + \frac{c_\alpha}{l_\alpha (w_{cp})} (\alpha - \frac{\tau}{2}).$$

The strength of this approximation comes from the fact that the critical path allows to deal with service times only, and not with idle times. It allows to remove the right number of errors on the service times, compared to Approximations 5.1 and 5.2. Let us illustrate this in the example of Figure 4.2. The running time can be computed in two ways. It can be computed as the sum of the job lengths and of the idle times on the third station. Alternatively, it can be computed as the sum of the lengths of the critical path's jobs. The error made on the job lengths can be (approximately) removed, while the error on the idle time is unknown. In Figure 4.2, it is thus smarter to remove the errors on the eight jobs of the critical path (which are approximatively known), while the errors on the eleven jobs in the last station could counterbalanced by the errors on the two starved times.

The weakness of this approximation comes from its complexity. As shown in Section 4.1.4, to compute the critical path, and thus Approximation 5.3, a new linear system of equations has to be solved. The complexity is thus $m$ times larger than for approximations 5.1. It is illustrated on Table 4.1, which can be compared to Table 3.1. Consequently, a trade-off appears between this supplementary computational cost and the accuracy benefit, and thus the potential decrease of the number $a$ of discretization steps, compared to the other approximations. However, note that the critical path computation is also required for the tight lower bound proposed in Section 4.3.1. Approximation 5.3 is thus “free” when this lower bound has already been computed. If one is interested by the tight bound, the approximation can be computed without additional computational cost. Alternatively, the cost of computing the critical path can be seen as shared between both applications.

From the formula, it can be seen straightforwardly that Approximation 5.3 reduces to $c \approx c_\alpha$ in the case where $\alpha/\tau = 0.5$, i.e. when the probability mass is aggregated in the middle of the interval. In this case, the computation of the critical path for Approximation 5.3 makes no sense. The approximation makes sense in case one choose $\alpha/\tau \neq 0.5$. This can be justified in particular cases. When one gives priority to the bounds for example, other shift parameters $\alpha$ can lead to better tightness, compared to $\alpha/\tau = 0.5$ (see tables 4.5 and 4.8). Also note that in the case of a tandem queue with buffers of size zero, Approximation 5.3 reduces to Approximation 5.1 as $c_\alpha / l_\alpha (w_{cp}) = 1$ in this case (see Proposition 4.3 and Corollary 4.7).

We now illustrate the behavior of Approximation 5.3 by numerical experiments. We use the same set of experiments as in Section 4.3.2 (on fork-join QN), and add experiments on split-and-merge queueing networks. The number of stations goes from two to four, the global storage space from zero to four. Each configuration has been tested with various service time distributions, arbitrary
chosen among the ten distributions used previously (in all our tests). Moreover, the parameters of the modelling method have also been varied: the number $a$ of discretization steps (4, 6 and 8 steps, see Figure 2.5) and the shift parameter $\alpha$ ($\alpha/\tau = 0, 0.25, 0.5, 0.75, 1$). In total, we analyzed 1500 different queueing networks and made 22500 experiments. Tables 5.14 and 5.15 give the average relative error, in percent, between Approximation 5.3 and the result of a simulation.

Earlier, we argued that Approximation 5.3 only makes sense when $\alpha/\tau \neq 0.5$. Indeed, in the usual case $\alpha/\tau = 0.5$, Approximation 5.3 reduces to Approximation 5.1, whose accuracy has been studied previously. We thus focus on the impact of the shift parameter $\alpha$. Table 5.14 illustrates this impact for FJQN and SMQN. First, it shows that the accuracy is good for any shift parameter $\alpha$ and that the latter has a quite small impact on the accuracy of Approximation 5.3. Again, this is particularly true for fork-join QN, while the method behaves less nicely on split-and-merge QN. On the opposite, in Section 5.1.1, we showed that approximations 5.1 and 5.2 show a strong dependence on parameter $\alpha$ (see tables 5.1 to 5.6). Table 5.15 further illustrate the impact of $\alpha$ on Approximation 5.1, on the current experiments set. Comparing tables 5.14 and 5.15, it can be seen that Approximation 5.3 performs significantly better when $\alpha/\tau \neq 0.5$, for FJQN as

### Table 5.14: Average relative error reached by Approximation 5.3 (critical path computation) on fork-join QN and on split-and-merge QN, using PMF without instant job, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 4$</th>
<th>$a = 6$</th>
<th>$a = 8$</th>
<th>$a = 4$</th>
<th>$a = 6$</th>
<th>$a = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.60%</td>
<td>0.25%</td>
<td>0.15%</td>
<td>2.14%</td>
<td>1.42%</td>
<td>0.83%</td>
</tr>
<tr>
<td>0.25</td>
<td>0.50%</td>
<td>0.22%</td>
<td>0.13%</td>
<td>0.88%</td>
<td>0.45%</td>
<td>0.30%</td>
</tr>
<tr>
<td>0.75</td>
<td>0.48%</td>
<td>0.21%</td>
<td>0.13%</td>
<td>1.51%</td>
<td>0.55%</td>
<td>0.28%</td>
</tr>
<tr>
<td>1</td>
<td>0.59%</td>
<td>0.25%</td>
<td>0.14%</td>
<td>2.68%</td>
<td>0.80%</td>
<td>0.37%</td>
</tr>
</tbody>
</table>

### Table 5.15: Average relative error reached by Approximation 5.1 (middle of the interval) on fork-join QN and on split-and-merge QN, using PMF without instant job, with $a$ and $\alpha$ changing.

<table>
<thead>
<tr>
<th>$\alpha/\tau$</th>
<th>$a = 4$</th>
<th>$a = 6$</th>
<th>$a = 8$</th>
<th>$a = 4$</th>
<th>$a = 6$</th>
<th>$a = 8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.79%</td>
<td>0.54%</td>
<td>0.35%</td>
<td>4.22%</td>
<td>3.51%</td>
<td>2.65%</td>
</tr>
<tr>
<td>0.25</td>
<td>0.52%</td>
<td>0.29%</td>
<td>0.19%</td>
<td>2.28%</td>
<td>1.72%</td>
<td>1.35%</td>
</tr>
<tr>
<td>0.75</td>
<td>0.48%</td>
<td>0.21%</td>
<td>0.13%</td>
<td>1.51%</td>
<td>0.55%</td>
<td>0.28%</td>
</tr>
<tr>
<td>1</td>
<td>0.84%</td>
<td>0.44%</td>
<td>0.26%</td>
<td>5.10%</td>
<td>2.82%</td>
<td>1.83%</td>
</tr>
<tr>
<td></td>
<td>1.30%</td>
<td>0.72%</td>
<td>0.44%</td>
<td>9.39%</td>
<td>5.30%</td>
<td>3.62%</td>
</tr>
</tbody>
</table>
well as for SMQN, while the average relative error is the same when \( \alpha/\tau = 0.5 \). In conclusion, Approximation 5.3 reveals to be interesting in the particular cases where one chooses \( \alpha/\tau \neq 0.5 \) (and only in these cases).

### 5.1.3 Managerial Example

In this section, we illustrate the use of the method by a simple managerial example, given in Tancrez et al. [85]. The method, using grouping at the end (\( \alpha = 0 \)), is applied to a tandem queue with three stations and two buffers of size one. We suppose an analyst is asked to assess the profit of a storage space enlargement. He thus studies the two buffers configurations with one more buffer space: \([2 \ 1]\) and \([1 \ 2]\). In the data collection stage, the service times are measured and given in the form of histograms with a bin width of five seconds (see top of Figure 5.1). The maximum service time is ten minutes. The minimization of the cycle time is the main objective.

The chart of Figure 5.1 (lower part) gives the upper and lower bounds (first methodology, Section 4.2) on the cycle time for the three buffers configurations, as well as approximations 5.2 (with \( i = 3 \)) and 5.3. We see that the bounds and the approximations regularly converge when the step size decreases. The interval between the upper and the lower bound equals the discretization interval size. The accuracy of the bounds is thus known a priori (1 minute with 10 steps, for example).

Furthermore, the accuracy of the approximations can be assessed by comparing them to simulation results. From Figure 5.1, it can be seen that both approximations give good results. Nevertheless, we observe that Approximation 5.3 brings a significant improvement (in this case, \( \alpha = 0 \)). We give the numerical results for the \([1 \ 1]\) buffer configuration, the accuracy is similar for the other configurations. The \([1 \ 1]\) configuration leads to a cycle time of 5.964. Approximation 5.2 makes an error of 1.14\% (0.54\%) if the step size for the probability mass fitting is chosen equal to 100 seconds, \( a = 6 \) (40 seconds, \( a = 15 \)). Approximation 5.3 leads to an error of 0.27\% with a step size of 100 seconds. The advantage of Approximation 5.3, using the critical path computation, is thus obvious in this application.

It can be seen on Figure 5.1 that this approximation leads to a better accuracy with four discretization steps (of size 2.5 minutes) than Approximation 5.2 with fifteen steps (40 seconds). With four discretization steps, Approximation 5.3 has to solve a system of 558 linear equations, which takes approximately 0.2 second\(^3\). While, with fifteen steps, Approximation 5.2 has to solve a system of 5598 linear equations, which takes approximately 3.2 seconds, for a less accurate estimation.

As expected, the analyst finds out that the configuration with a buffer of size two in first position is better, as the beginning of the tandem queue is more variable. The benefit in terms of productivity can be estimated at 3.2\% (the cycle time goes from 5.948 to 5.757). Opting for this storage space enlargement (\([1 \ 1]\) to \([2 \ 1]\)) thus seems to be a good advice. However, the analyst would like to study

\(^3\)Using the Gaussian elimination implemented in MATLAB\textsuperscript{\textregistered} on a 2.16 GHz usual PC, 2 GB RAM, Intel\textsuperscript{\textregistered} Centrino Duo processor.
Figure 5.1: Cycle time estimations for a three station tandem queue with various buffers configurations and service times histograms shown in the upper part. Continuous lines stands for two buffers of size one, [1 1], dashed lines for [2 1], and dash-dotted lines for [1 2].

From Figure 5.1, one could get the idea to use extrapolation in order to find a accurate estimation of the cycle time. Indeed, the measures presented on the figure (the bounds in particular) seem to converge regularly when the step size decreases, and, at the limit ($\tau = 0$), they converge. We could thus extrapolate the measure made with various step sizes $\tau$, in $\tau = 0$. However, we tested the idea on the experiments set presented previously and the results were not satisfactory. The extrapolation is not more accurate than Approximation 5.1 using PMF/nIJ with $\alpha/\tau = 0.5$, while it is more expensive as several estimations, with various parameters $\tau$, are required.

5.2 Other Performance Measures

In the previous section, we focused on the estimation of the cycle time. However, our method offers a quite complete modelling of the system. The Markov chain presented in Chapter 3 describes the evolution of the queueing network, and other performance measures can be estimated from it. Here, we give results for two
5.2. OTHER PERFORMANCE MEASURES

Table 5.16: Average relative error reached by the work-in-progress estimation, on fork-join QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the number of stations vary.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$m = 2$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.73%</td>
<td>0.53%</td>
<td>0.42%</td>
<td>0.87%</td>
<td>0.61%</td>
<td>0.64%</td>
<td>0.35%</td>
</tr>
<tr>
<td>6</td>
<td>0.72%</td>
<td>0.25%</td>
<td>0.22%</td>
<td>0.43%</td>
<td>0.40%</td>
<td>0.28%</td>
<td>0.17%</td>
</tr>
<tr>
<td>8</td>
<td>0.43%</td>
<td>0.18%</td>
<td>0.15%</td>
<td>0.33%</td>
<td>0.36%</td>
<td>0.18%</td>
<td>0.11%</td>
</tr>
</tbody>
</table>

Table 5.17: Average relative error reached by the work-in-progress estimation, on split-and-merge QN using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the number of stations vary.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
<th>$m = 3$</th>
<th>$m = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2.63%</td>
<td>1.51%</td>
<td>7.51%</td>
<td>3.58%</td>
</tr>
<tr>
<td>6</td>
<td>0.86%</td>
<td>0.50%</td>
<td>2.24%</td>
<td>1.23%</td>
</tr>
<tr>
<td>8</td>
<td>0.44%</td>
<td>0.26%</td>
<td>1.17%</td>
<td>0.78%</td>
</tr>
</tbody>
</table>

important measures, the WIP and the flow time, but others could be estimated. First, the work-in-progress (WIP) gives the average number of units in the system, i.e., in the stations as well as in the buffers. In production systems, estimations of the WIP allow to determine the working capital requirement. It also reflects the ability to improve the production process (see Bitran and Dasu [14]). The WIP is easily computed from the occupations of the buffers and of the stations. These occupations are included in the definition of each state of the Markov chain. Consequently, to compute them, one just has to weight them by the corresponding stationary probabilities. Second, the flow time gives the average time a unit spends in the system, from its entrance in a source station to its leaving from a sink. The flow time can be computed from the Markov chain, but it is more accurately estimated using Little’s law and the estimations of the cycle time and of the WIP.

We test the accuracy of the estimations of the work-in-progress and of the flow time on the experiments set previously used (see sections 4.2.2, 4.4 and 5.1.1). We analyse tandem queues, fork-join QN and split-and-merge QN with two, three or four stations (15 different network topologies). The global storage space of a network goes from zero to four and is balanced among the buffers. The service time distributions are chosen randomly among the ten distributions used previously (see Figure 2.5). In total, 3750 networks were analysed. The method was run using PMF/nIJ aggregating the probability masses in the middle of the interval, with a number of discretization steps $a$ equal to 4, 6 or 8. The results were then compared to simulation results.

Tables 5.16 to 5.19 give the average relative errors obtained for the work-in-progress estimation. They show that the method also evaluates the WIP accurately, and more and more accurately when the PMF discretization is refined.
The average relative error is in the order of some tenths of a percent, except for merge networks, for which the method is again less accurate. Tables 5.16 and 5.17 illustrate the impact of the topology and of the number of stations. Unlike the approximation of the cycle time, the accuracy of the estimation of the WIP varies significantly according to the topology of the FJQN network (see Table 5.16). In particular, the evaluation appears to be less accurate for fork networks. Furthermore, for any network topology, the accuracy of the evaluation seems to improve when the number of stations increases, it is a positive characteristic of the estimation.

Tables 5.18 and 5.19 show the influence of the storage space on the work-in-progress evaluation. For fork-join queueing networks, the impact of the buffer sizes is not clear from Table 5.18. For split-and-merge queueing networks, Table 5.19 shows that the accuracy of the WIP estimation deteriorates when the storage space increases, it is a negative characteristic. All in all, the latter observation and the positive effect of the number of stations tend to show that, when the portion of units in the buffers among the WIP is large, the quality of the WIP evaluation is less good. This tends to show that the approximation of the occupation of the buffers is less accurate than the approximation of the occupation of the stations.

Tables 5.20 to 5.23 give the average relative errors obtained by the flow time estimation. They are quite similar to the corresponding tables for the WIP estimation (tables 5.16 to 5.19). The main difference concerns merge networks. The estimation of the flow time is less good, and is quite bad when the number of discretization steps is chosen too low (see Table 5.21). Otherwise, the conclusions about the flow time evaluation are the same than about the WIP evaluation (we

---

4The estimation is also less accurate for two station tandem queues. Note that, in this case, the estimation of the WIP is significantly more accurate when $\alpha$ is chosen to conserve the expectation (Proposition 2.6): 0.79%, 0.40%, 0.29% with $a = 4, 6, 8$. 

---

<table>
<thead>
<tr>
<th>$a$</th>
<th>$B_{\Sigma} = 0$</th>
<th>$B_{\Sigma} = 1$</th>
<th>$B_{\Sigma} = 2$</th>
<th>$B_{\Sigma} = 3$</th>
<th>$B_{\Sigma} = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.73%</td>
<td>0.56%</td>
<td>0.59%</td>
<td>0.74%</td>
<td>1.03%</td>
</tr>
<tr>
<td>6</td>
<td>0.67%</td>
<td>0.32%</td>
<td>0.27%</td>
<td>0.38%</td>
<td>0.47%</td>
</tr>
<tr>
<td>8</td>
<td>0.66%</td>
<td>0.26%</td>
<td>0.20%</td>
<td>0.27%</td>
<td>0.32%</td>
</tr>
</tbody>
</table>

Table 5.18: Average relative error reached by the work-in-progress estimation, on fork-join QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the storage space vary ($B_{\Sigma}$ stands for $\sum b(i, j)$).

<table>
<thead>
<tr>
<th>$a$</th>
<th>$B_{\Sigma} = 0$</th>
<th>$B_{\Sigma} = 1$</th>
<th>$B_{\Sigma} = 2$</th>
<th>$B_{\Sigma} = 3$</th>
<th>$B_{\Sigma} = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.84%</td>
<td>1.66%</td>
<td>2.26%</td>
<td>3.52%</td>
<td>4.51%</td>
</tr>
<tr>
<td>6</td>
<td>0.34%</td>
<td>0.61%</td>
<td>0.71%</td>
<td>1.18%</td>
<td>1.46%</td>
</tr>
<tr>
<td>8</td>
<td>0.21%</td>
<td>0.37%</td>
<td>0.40%</td>
<td>0.67%</td>
<td>0.82%</td>
</tr>
</tbody>
</table>

Table 5.19: Average relative error reached by the work-in-progress estimation, on split-and-merge QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the storage space vary ($B_{\Sigma}$ stands for $\sum b(i, j)$).
5.2. OTHER PERFORMANCE MEASURES

<table>
<thead>
<tr>
<th></th>
<th>Tandem</th>
<th></th>
<th>Fork</th>
<th></th>
<th>Join</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>m = 2</td>
<td>m = 3</td>
<td>m = 4</td>
<td>m = 3</td>
<td>m = 4</td>
</tr>
<tr>
<td>4</td>
<td>1.69%</td>
<td>0.63%</td>
<td>0.52%</td>
<td>0.93%</td>
<td>0.67%</td>
</tr>
<tr>
<td>6</td>
<td>0.71%</td>
<td>0.29%</td>
<td>0.25%</td>
<td>0.44%</td>
<td>0.41%</td>
</tr>
<tr>
<td>8</td>
<td>0.42%</td>
<td>0.20%</td>
<td>0.16%</td>
<td>0.32%</td>
<td>0.36%</td>
</tr>
</tbody>
</table>

Table 5.20: Average relative error reached by the flow time estimation, on fork-join QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the number of stations vary.

<table>
<thead>
<tr>
<th></th>
<th>Split</th>
<th></th>
<th>Merge</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>m = 3</td>
<td>m = 4</td>
<td>m = 3</td>
</tr>
<tr>
<td>4</td>
<td>1.65%</td>
<td>1.27%</td>
<td>11.28%</td>
</tr>
<tr>
<td>6</td>
<td>0.57%</td>
<td>0.41%</td>
<td>3.43%</td>
</tr>
<tr>
<td>8</td>
<td>0.32%</td>
<td>0.23%</td>
<td>1.63%</td>
</tr>
</tbody>
</table>

Table 5.21: Average relative error reached by the flow time estimation, on split-and-merge QN using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the number of stations vary.

<table>
<thead>
<tr>
<th></th>
<th>$B_\Sigma = 0$</th>
<th>$B_\Sigma = 1$</th>
<th>$B_\Sigma = 2$</th>
<th>$B_\Sigma = 3$</th>
<th>$B_\Sigma = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0.73%</td>
<td>0.68%</td>
<td>0.67%</td>
<td>0.66%</td>
<td>0.52%</td>
</tr>
<tr>
<td></td>
<td>0.56%</td>
<td>0.33%</td>
<td>0.56%</td>
<td>0.66%</td>
<td>0.52%</td>
</tr>
<tr>
<td></td>
<td>0.66%</td>
<td>0.30%</td>
<td>0.28%</td>
<td>0.21%</td>
<td>0.28%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.40%</td>
<td>0.28%</td>
<td>0.21%</td>
<td>0.33%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.84%</td>
<td>0.40%</td>
<td>0.52%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.17%</td>
<td>0.40%</td>
<td>0.52%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5.62%</td>
<td>0.40%</td>
<td>0.52%</td>
</tr>
</tbody>
</table>

Table 5.22: Average relative error reached by the flow time estimation, on fork-join QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the storage space vary ($B_\Sigma$ stands for $\sum b(i,j)$).

<table>
<thead>
<tr>
<th></th>
<th>$B_\Sigma = 0$</th>
<th>$B_\Sigma = 1$</th>
<th>$B_\Sigma = 2$</th>
<th>$B_\Sigma = 3$</th>
<th>$B_\Sigma = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1.36%</td>
<td>0.52%</td>
<td>0.26%</td>
<td>2.22%</td>
<td>0.80%</td>
</tr>
<tr>
<td></td>
<td>2.22%</td>
<td>1.02%</td>
<td>0.42%</td>
<td>3.15%</td>
<td>1.02%</td>
</tr>
<tr>
<td></td>
<td>3.15%</td>
<td>1.44%</td>
<td>0.42%</td>
<td>4.47%</td>
<td>1.44%</td>
</tr>
<tr>
<td></td>
<td>4.47%</td>
<td>1.76%</td>
<td>0.42%</td>
<td>5.62%</td>
<td>1.76%</td>
</tr>
<tr>
<td></td>
<td>5.62%</td>
<td></td>
<td>0.76%</td>
<td>0.91%</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.23: Average relative error reached by the flow time estimation, on split-and-merge QN, using PMF without instant job with $\alpha/\tau = 0.5$. The number of steps $a$ and the storage space vary ($B_\Sigma$ stands for $\sum b(i,j)$).
use Little’s law). The approximation of the flow time is accurate (except for merge networks when $a$ is small) and improves when the PMF is refined. In the case of fork-join queueing networks, the topology of the network and the storage space have no clear impact (except that the estimation is less accurate for fork networks, see tables 5.20 and 5.22). In the case of split-and-merge networks, the estimation of the flow time is much better for split networks and the quality of the evaluation deteriorates when the storage space increases (see tables 5.21 and 5.23). For both kinds of queueing network, the accuracy of the flow time estimation improves when the number of stations increases (see tables 5.20 and 5.21).

5.3 Cycle Time Distributions

In the previous sections, we focus on the estimation of average performance measures. This is also the case for the vast majority of the literature on the performance evaluation of queueing networks. However, complementary information can be very useful, in order to reflect the stochastic nature of the performance of a queueing network. The stochasticity can be partially shown by the variability or the percentiles, for example. Ideally, one would like to estimate the distribution of the performance, which fully represents the stochasticity, and from which the variance and the percentiles can be computed.

In the extension section of their reference paper on production lines, Dallery and Gershwin [26] claim that the calculation of the variance of the output “is a tremendously important area because manufacturers must deliver products on a daily or weekly basis”, and that the “prediction of this variability is no less important than that of the prediction of the mean; in fact, it may be more important”. “The firm might insist that [the production line] be able to deliver a certain amount of finished product each week with a certain probability. In order to do this, the research community must be able to provide statistical information on the number of parts produced during a given time interval other than just the mean: ideally, the probability distribution; more realistically, it can provide the standard deviation”.

In Section 2.4, we showed that probability mass fitting has the advantage to preserve the shape of the original service time distribution (see Figure 2.7). This characteristic can even be thought as a motivation of PMF. Like we did for other properties of PMF (the bounding property and the monotonicity, see sections 4.2 and 4.6), we now extend this PMF property, the shape conservation, to the global modelling method.

5.3.1 Computation

To begin, we briefly explain how the cycle time distribution can be computed (using PMF without instant job, to begin). By cycle time distribution, we mean the distribution of the time between two jobs exiting a given station. In the discretized time, the cycle time distribution is of phase type, i.e. it can be modelled as the time until absorption of a discrete Markov chain with one absorbing state. It is fully characterized by the transition matrix of the Markov chain and the initial probabilities of starting in any of the states of the chain (see Neuts [64]).
Figure 5.2: Example on a two station tandem queue with a buffer of size one. The left-hand side shows the Markov chain modelling the evolution of the system (using PMF/NIJ, see Figure 3.1). Dashed transition are from $[1, 2, \ldots, a \text{ or } B]$ to $[S \text{ or } 1]$, in the second station. The right-hand side shows the Markov chain associated with the phase-type distribution of the cycle time of the second station.

The transition matrix characterizing the distribution of the cycle time is quite easily deduced from the Markov chain modelling the evolution of the system (see Figure 3.1). In the latter Markov chain, a transition from a state where station $i$ is working or blocked to a state where station $i$ is starved or in its first stage of service (symbol 1) corresponds to one unit leaving station $i$. When station $i$ is working or blocked, one unit is in the station. When station $i$ becomes starved, the station thus finished to serve, or unblocked, and put the unit in the next buffer (but did not find any new item to serve). Similarly, when station $i$ becomes “in first stage of service”, it means the unit left and the station began to serve a new job. To get the transition matrix of the cycle time PH distribution of station $i$, we just have to modify these transitions (from $[1, 2, \ldots, a \text{ or } B]$ to $[S \text{ or } 1]$, see left-hand side of Figure 5.2). The departure state of the transition stays the same but the transition goes to the absorbing state of the Markov chain characterizing the PH distribution. Informally said, reaching the absorbing state means that one item leaves the station. The Markov chain of the PH cycle time distribution (of the second station) is given in Figure 5.2 for a two station tandem queue. It illustrates how the Markov chain modelling the evolution of the system, given in Figure 3.1, is modified.

Besides the transition matrix, the only other information needed to characterize a phase-type distribution is the initial probabilities $p_{\text{init}}$ of starting in any of the states of the chain. In our case, the initial probabilities are non-zero for the destination states of the transitions previously mentioned (from $[1, 2, \ldots, a \text{ or } B]$ to $[S \text{ or } 1]$, on station $i$). Indeed, these transitions correspond to the end of one cycle time, and thus the beginning of the next one. The initial probability of a state is thus given by the probability to make one of these transitions, to this particular state, weighted by the stationary probability of the departure state of the transition. The vector of initial probabilities $p_{\text{init}}$ can thus be computed as follows:

$$p_{\text{init}} = \frac{\pi P_{\text{out}}}{\pi P_{\text{out}} C},$$
where $P_{out}$ is the matrix of the transitions from $[1, 2, \ldots, a$ or $B]$ to $[S$ or $1]$ (on station $i$), i.e. corresponding to the end of a cycle time (and the beginning of the next one), $\pi$ is the vector of the stationary probabilities of the states of the Markov chain modelling the evolution of the system, and $e$ is a column vector of ones.

The cycle time distribution of any station can thus be computed in the modified system. It is a discrete phase-type distribution, characterized by its transition matrix and its initial probabilities. From the latter characteristic, thanks to a mature theory (see Neuts [64]), the probability density function can be formulated, and the moments can be computed, for example. Note that the computation does not require any matrix inversion (such as for the stationary probabilities computation).

In the case where PMF/IJ is used, the computation of the cycle time distribution is similar but instant jobs make it more complex as several job can end instantaneously. Notably, the probability to start on the absorbing state of the Markov chain of the PH distribution is non-zero.

5.3.2 Shape Conservation

In the previous section, we show that the cycle time distribution of any station can be computed in the modified system. We now argue that it is a good approximation of the cycle time distribution in the original system. In other words, the computed distribution has a shape close to the real (unknown) cycle time distribution. The shape conservation of the service time distributions (by the PMF) extends to the cycle time distributions. We do not prove this result but give convincing arguments. The many examples we tested give strong evidence. In the following, we give some examples, others show the same behavior.

Let us have a look at the example queueing network analyzed in sections 4.2 and 4.6: a three station tandem queue with buffers of size one and service time distributions beta(2, 2), uniform(0, 1) and triangular(0, 1, 0.5). The PMF/IJ discretization of the beta(2, 2) and triangular(0, 1, 0.5) distributions is shown on Figure 2.7, to illustrate the shape conservation of the PMF transformation (the discretized uniform merges with the original one). Figure 5.3 depicts, for each station, the computed discrete cycle time distribution. The three graphs show how starving and blocking impact the cycle time. The distributions are different from the beta, uniform and triangular service time distributions (see Figure 2.7). On station 2 for example, the uniform distribution is sensibly modified: as the blocking and starving times are added, the cycle time is longer than the service time, there is a smaller probability for a cycle time to be on the left, and a larger probability to be on the right hand side.

The cycle time distributions are computed with 5, 10 and 20 steps. As our method refines and converges when the number of discretization steps increases, it can be supposed that the cycle time distribution computed with 20 steps is accurate. Figure 5.3 reveals that the shape of a cycle time distribution appears to be independent of the number of discretization steps used. The distribution is of course more detailed with 20 steps but the distributions computed with 5 or 10 steps show the same shape. It tends to show that the cycle time distribution estimations with 5 or 10 steps are already good.
5.3. CYCLE TIME DISTRIBUTIONS

Figure 5.3: Cycle time distributions computed, using PMF/nIJ and $\alpha/\tau = 0.5$, for the three stations of a tandem queue (beta(2,2), uniform(0,1) and triangular(0,1,0.5) service time distributions), with 5 (dashed), 10 (dotted) and 20 (solid) discretization steps.

Consequently, we may say that the cycle time distributions computed by our modelling method have a shape close to the real one (also see the comparison with simulation results in the next section, figures 5.5 to 5.7). Even with 5 or 10 steps, the approximation of the cycle time distribution is already good. The computation of a good cycle time distribution estimation is a significant advantage of our method, compared to other fitting methods which are unable to do so, such as moments fitting in particular (see Pearson et al. [70]). The distribution offers more detailed information on the behavior of the system, compared to the isolated expectation. It allows to estimate measures such as the variance or the percentiles. In practice, the percentiles are crucial information in order to help managers to evaluate the risk and the service level.

5.3.3 Examples

In this section, we give several examples aiming to illustrate the use of the cycle time distribution estimation. First, we exemplify how the buffers sizes influence the cycle time distributions (note that PMF with instant jobs is used). We consider another three station tandem queue with uniform service time distributions on each station. Uniform distributions allow to clearly see the effect of blocking and starving.

On Figure 5.4, we show the system with no storage space (solid line). As the first station is never starved, the cycle time on it equals the service time plus some blocking time, and it can never be larger than the maximum service time\(^5\). That

---

\(^5\) Suppose the first station works on item $k$ and gets blocked. It means the buffer between the first and second stations is full during this time, i.e. from the beginning of the job $w_{1,k}$ to the end of the blocking time. The job $w_{2,k'}$ which the second station is serving during this time cannot have begun after job $w_{1,k}$ (otherwise the buffer would not be full). As the end of $w_{2,k'}$ unblocks item $k$ from the first station, the service time plus the blocking time in the first station is smaller than $w_{2,k'}$. \]
CHAPTER 5. PERFORMANCE EVALUATION

Figure 5.4: Cycle time distributions computed, using PMF/IJ and $\alpha/\tau = 0.5$, for the three stations of a tandem queue (uniform service time distributions), with 10 discretization steps. Three different buffer configurations are analyzed: [0 0] (solid line), [1 0] (dashed) and [0 1] (dash-dotted).

is the reason why the cycle time never exceeds one in the first station on Figure 5.4. On the opposite, in the last station, the cycle time is composed of the service time and the starved time (no blocking time), and it may exceed one. On the cycle time distribution, the blocking thus tend to increase the probability of the higher values of the support of the service time distribution (see first station). The starving give rise to a “tail” in values higher than the maximum possible service time (see third station).

In Figure 5.4, the cycle time distributions are also drawn for the cases where the storage space is increased by one, in the first or in the second buffer. If one focus on the average cycle time, i.e. the mean of the cycle time distributions$^6$, the [0 1] and [1 0] buffer configurations cannot be distinguished. The average cycle time equals 0.7297 without storage space, and 0.6781 in both [0 1] and [1 0] cases. However, the cycle time distributions, and particularly the one of the second station, are clearly different. The cycle time distributions of the first and third station are slightly affected by the fact that the storage space is allocated to the first or second buffer. The cycle time distribution of the second station is higher in the maximum service time region for the [1 0] buffer configuration, as the first buffer prevents some starving but no blocking. On the opposite, the distribution shows a larger tail in the [0 1] case as the second buffer prevents some blocking but no starving.

To further convince of the accuracy of the cycle time distribution approximation, we compare the computation to the result of a simulation, on the same example (uniform distributions, no storage space), for the third station. The results are given on Figure 5.5. The distribution, its shape and values, show to be remarkably similar.

$^6$The means of the three cycle time distributions are equal.
5.3. CYCLE TIME DISTRIBUTIONS

Figure 5.5: Cycle time distribution of the third and last station of a tandem queue (uniform service time distributions, no storage space). The left-hand side gives the distribution computed by the method using PMF/IJ ($\alpha/\tau = 0.5$) with 10 discretization steps. The right-hand side gives the distribution computed via simulation.

In the previous examples, we considered three station tandem queues which suit well for a clear explanation. The computation of the cycle time can of course be done for tandem queues as well as for fork-join or split-and-merge queueing networks. The approximation is also accurate in these cases. Let us consider a join (assembly) system with four stations. The two first stations are in series, and the fourth station assembles the units coming from the second and third stations. The buffer sizes equal one except for the buffer between the third and fourth station, whose size is zero. The service time distributions are beta(2,2), uniform(0,1), triangular(0.1,0.9,0.6) and beta(5.5,6) (according to the ordering of the stations, also see Figure 2.5). The cycle time distribution is computed using grouping at the end ($\alpha/\tau = 0$). Figure 5.6 shows that the estimation is very close to the simulation result. Similarly, Figure 5.7 illustrates the accuracy of the estimation of the cycle time distribution for a split system, with three stations (see Figure 4.6, with $p = 0.5$), two buffers of size two, and service time distributions uniform(0,1), beta(4,4) and triangular(0.2,1,0.3) (ordered). Many other examples could be presented, and would show similar accuracy.

5.3.4 Managerial Example

In this section, we pursue the managerial example given in Section 5.1.3. An analyst studies the opportunity of enlarging the storage space in a tandem queue with three stations and two buffers of size one. The service times are collected in the form of histograms (see top of Figure 5.1). In Section 5.1.3, we observed that increasing the size of the first buffer is the best solution in terms of productivity, leading to a benefit of approximately 3.2% (the cycle time goes from 5.948 to 5.757).
Figure 5.6: Cycle time distribution of the fourth and last station of an assembly system (various service time distributions). The left-hand side gives the distribution computed by the method using PMF/nIJ ($\alpha/\tau = 0$) with 10 discretization steps. The right-hand side gives the distribution computed via simulation.

Figure 5.7: Cycle time distribution of the third and last station of a split system (various service time distributions). The left-hand side gives the distribution computed by the method using PMF/nIJ ($\alpha/\tau = 0$) with 10 discretization steps. The right-hand side gives the distribution computed via simulation.
5.3. CYCLE TIME DISTRIBUTIONS

Figure 5.8: Cycle time distributions for a three station tandem queue with various buffers configurations. Continuous lines stands for two buffers of size one, [1 1], dashed lines for [2 1], and dash-dotted lines for [1 2].

In Section 5.2, we showed that the method offers a quite complete modelling of the system. Other performance measures can be approximated, such as the work-in-progress and the flow time. Increasing the storage capacity is always beneficial in terms of the productivity of the system, as it increases the decoupling between the stations and thus reduces the productivity losses caused by the blocking and starving phenomena. However, enlarging the buffer sizes has the drawback to increase the work-in-progress. Increasing the WIP implies a larger working capital requirement and it also lowers the ability of the production process to be improved. The proposed approach allows to estimate the work-in-progress. Going from the [1 1] to the [2 1] buffer configuration, the WIP increases from approximately 3.6 to 4.4 units in the system on average. Moreover, the analyst may evaluate the flow time, which reflects the delay to answer a client demand, or the ability of the process to react quickly. The flow time also increases when the buffer sizes go from [1 1] to [2 1]: it increases from 21.4 to 25.1 minutes, approximately. The WIP and the flow time represent a cost that should be balanced with the productivity benefit.

Furthermore, we showed in Section 5.3 that the cycle time distributions can be approximated. They offer more detailed information than the average. On Figure 5.8, we depict the computed distributions for the various buffer configurations. It can notably be observed that the tail (beyond 10 minutes) of the cycle time distributions of the last two servers is decreased with the [2 1] buffer configuration. In other words, the probability to have an unusually long delay between two items leaving a station is decreased, what is a good characteristic brought by this buffer enlargement. From the distributions, the variance can also be approximated. For the third server, i.e. the output of the system, the variance is decreased from 0.031 to 0.025, when going from the [1 1] to the [2 1] configuration (coefficient of variation: from 0.295 to 0.276). This is another profit brought by the storage space enlargement. It is important for the service level to the clients.

From his study, the analyst may then optimize the buffer configuration, thanks to the performance evaluation, and in function of the production costs, the inven-
tory costs, the delay costs and so on. Finally, he is able to present these results to the manager, to help him to take a decision with good knowledge of the implications.

5.4 Comparison with Other Distribution Fittings

Ideally, the performance evaluation should be compared to the results of other concurrent methods. However, this reveals to be difficult as no results are available in the literature to directly compare the accuracy levels we obtain.

As explained in Section 1.3, the modelling process is made of three steps: data collection, distribution fitting and analytical modelling. As the problem is otherwise analytically intractable with general distributions, the approximation of the distributions is actually unavoidable, before the proper analytical modelling. Our main originality lies in the distributions fitting stage, for which we propose probability mass fitting. The relative errors presented in the previous sections essentially reflect the error brought, in the global modelling process, by the distribution fitting (i.e. the PMF), as the analytical modelling (i.e. the state model) is exact. Concurrent methods in this matter, methods from which results should be compared, are thus distribution fitting methods. The main distribution fitting methods are presented in Section 1.3.2. The most popular one, in the literature and in practice, is moments fitting.

However, in the litterature, the distributions fitting error is very rarely considered in the global modelling process. Papers presenting new distribution fitting methods do not study their effect on the following analytical modelling (and most of them suppose infinite support distributions). When new analytical methods are presented, phase-type distributions are supposed to be given, and the distribution fitting, and its approximation error, is skipped. A direct comparison with other methods is thus difficult. Nevertheless, in the following, we give some insights (also see sections 1.3.2 and 2.4).

Some results achieved thanks to probability mass fitting are just unreachable by other methods. First, it allows to compute bounds, and particularly a tight lower bound which was shown to compare favorably to other bounding methodologies (see Section 4.3.2). These bounds offers some certainty and an interesting complement to the approximations. Second, as shown in Section 5.3, PMF allows to approximate the cycle time distributions, which give more detailed information than just the average cycle time. To the extent of our knowledge, no other methods can accurately approximate cycle time distributions (supposing general service time distributions). Third, with our method, the accuracy of the performance is refinable. With PMF, the accuracy level can be chosen a priori, by selecting the number of discrete values accordingly. Of course, in practice, the accuracy of the results depends on the computational cost one can afford. Moments fitting is most often limited to the two or three first moments, and is less flexible.

The number of phases in the transformed distribution is the main source of complexity in the subsequent analysis. In the previous experiments, we choose probability mass fittings leading to 4, 6 or 8 phases (number of discrete probabilities $a$). This choice is consistent with the state of the art. Lang and Arthur
[55] made an extensive study of the existing distribution fitting methods, moments fitting as well as maximum likelihood methods and others, and compared their results using the standard Aalborg benchmark (see Bobbio and Telek [16]). They argue that state of the art methods using 4 to 8 phases lead to satisfactory results for “continuous” phase-type behaved” distributions while sensibly more phases (16) are required for other distributions. Neuts [64], in his reference book, Asmussen [9], in his review paper, and Bobbio and Telek [16], for their own maximum likelihood method, state similar conclusions. Among the distributions we used in our tests, beta distributions can be considered as “phase-type behaved” while uniform and triangular cannot.

Concerning moments fitting in particular, a classical formula exists for two moments fitting, discovered by Aldous and Shepp [2]. It links the moments and the required number of phases in the fitted distribution as follows: 

\[ \frac{1}{k} \leq cv^2 \leq \frac{1}{k-1} \]

where \( k \) is the number of phases and \( cv^2 \) is the squared coefficient of variation. The 10 distributions used in our tests would need 3, 5, 5, 9, 12, 17, 23, 6, 8 and 11 phases, in the same listing order (see Figure 2.5). If three moments are fitted, the number of required phases is of course still increased (see Osogami and Harchol-Balter [67]). These observations show that the number of phases used in our experiments (and leading to accurate results) is reasonable.

Concerning other fitting methods, Johnson [46] proposed a method combining moments fitting with distance minimization and tested it on some GI/M/1 queues, getting WIP relative errors ranging from nearly zero to five percent. Thummller et al. [91] developed a maximum likelihood distribution fitting and tested it on G/M/1/K queues. They notably get a relative error of 8% on the WIP (with 15 phases). Of course, these observations on single queues (and with exponential service time) do not allow any conclusions on queueing networks, but they tend to show that distribution fitting (by moments or other state of the art methods)
can lead to significant errors in the global modelling process. According to Altiok [5], “there is always an error in this type of approximation [i.e. distribution approximation], and it is important to invest every effort to minimize it”.

To further assess the quality of our results, we now apply a basic method, which is widely used in practice, presented in Hopp and Spearman [43]. It is a closed-form (and thus quick) analytical method for two station tandem queues, which just needs the two first moments of the service time distributions as parameters. We apply the method on 2500 tandem queues using the same test parameters as previously: buffer sizes ranging from 0 to 4 and service time distributions chosen randomly from the same ten distributions (see Figure 2.5). We get a 5.7% average relative absolute error on the cycle time. This can be compared with the accuracy of the proposed cycle time estimation, given on the first column of Table 5.8 (0.24% with $a = 6$ for example). Moreover, the cycle time is sometimes overevaluated and sometimes underevaluated by the method presented in Hopp and Spearman [43]. On the opposite, the bounds that PMF allows to compute provides some certainty about the computed estimation. In particular, we know that the cycle time will not be smaller than the lower bound (see the first column of Table 4.9, 0.38% with $a = 6$). Also note that a 60.3% maximum relative error has been observed with the closed-form method. Our method leads to 1% (resp. 1.41%) maximum relative error for the approximation (resp. bound), with $a = 6$, on the reported tests.

Finally, in order to further illustrate the difference between probability mass fitting and moments fitting, we compare them on a simple example. Let us consider a pair of tandem queues with three stations and no buffer. The exact service time distributions are triangular. They are “mirrored” from one tandem queue to the other in order to obtain the same mean and same variance (see top of Figure 5.9). Consequently, methods based on the fitting of the two first moments would not see any difference between both tandem queues. A simulation shows that the cycle times of these two tandem queues differ by 1.62%, from 0.6354 to 0.6458. While models based on a two first moments fitting would not make any difference, our method clearly distinguishes both systems cycle times, even with a small number of phases.

First, the bounds computed by our method allow to mathematically prove the difference. Corollary 4.15 (or equation (4.20)) allows to a priori size the discretization step needed in order for the bounds to cross each other. The step size is chosen equal to 0.01, i.e. leading to 100 and 80 discrete values. In this case, our method computes a lower bound of 0.6407 for one tandem queue (in 60 seconds\footnote{Using the Gaussian elimination implemented in MATLAB® on a 2.16 GHz usual PC, 2 GB RAM, Intel® Centrino Duo processor.}, 14636 states) and an upper bound of 0.6403 for the other tandem queue (in 40 seconds, 13036 states). The difference between the cycle time of the systems is thus proved rigorously. It can be seen on Figure 5.9 that the lower bound for one tandem queue intersects the upper bound for the other one.
5.4. COMPARISON WITH OTHER DISTRIBUTION FITTINGS

Figure 5.9: Bounds and approximations of the cycle time for three station tandem queues without buffers, using grouping at the end (PMF/nIJ with \( \alpha = 0 \)). Their triangular service time distributions, shown on upper part, have the same two first moments. The dashed (resp. solid) results correspond to the dashed (resp. solid) distributions.

Second, the approximations offer accurate estimations of the two different cycle times and allow to quickly be convinced of the difference between them. Note that Approximation 5.3 is not useful here as no buffer is included in the system (the correction ratio \( c_\alpha/l_\alpha(w_{cp}) \) equals one in this case, see Proposition 4.3 and Corollary 4.7). It can be seen on Figure 5.9 that its results are not better than those by Approximation 5.2. For both tandem queues, the method reaches a 0.15% accuracy with \( a = 5 \) discretization steps (in 0.2 second, 61 states) and computes the right four-digit results with \( a = 10 \) (in 0.3 second, 226 states). For these triangular distributions, a two first moments fitting would lead to PH distribution including at least 8 phases (according to the equation given by Aldous and Shepp [2]). The complexity would thus be equivalent to the one of our method with \( a = 8 \), while the moments fitting would not see the difference (1.62%) between the systems.

In conclusion, this comparison to the results available in the literature as well as our experiments and examples allow us to say that probability mass fitting leads to good accuracy in the evaluation of the performance of queueing networks, using a reasonable number of phases. PMF moreover offer some certainty thanks to
bounds. Together with the bounds, the approximations allow to get a good grasp on the exact measure with certainty. PMF is refinable, so that the accuracy can be chosen according to the affordable computational cost. It also offer a more detailed information as it allows to approximate the distribution of the cycle time (and thus the variance and percentiles for example). All in all, we believe probability mass fitting can be thought as a valuable alternative to fit distributions, in the first step of modelling process, preceding state of the art analytical methods.
Chapter 6

Decomposition

In Chapter 3, we presented a modelling method coupling probability mass fitting with a state model. It allows us to compute bounds and approximations on the performance measures of queueing networks in chapters 4 and 5. However, as explained throughout these chapters, this modelling method suffers from high complexity. The size of the Markov chain to be resolved increases quickly with the size of the modelled queueing system. We argued that the complexity comes from the state model, and not from the PMF (the number of stages in the fitted distributions is reasonable). The state model is an exact analytical model. The natural way to lower the complexity, is to thus turn to an approximate model. The most popular approximate analytical model is the decomposition (see Section 1.3.3).

In this chapter, we apply a decomposition method in the analytical modelling stage of the global modelling process, while still using PMF for distribution fitting. Our goal is to improve the applicability of our method, i.e. to be able to evaluate the performance of larger queueing networks. Of course, as we use an approximate model in the second stage of the method, the performance evaluation will be less accurate and the bounds will be lost. However, the decomposition has been extensively studied in the literature and has been shown to lead to accurate performance evaluation. Moreover, the technique is based on the decomposition of the system into several subsystems, and on the modification of the service time distributions of the the stations to approximate the impact of the rest of the system, in terms of blocking and starving. The ability to reliably approximate the cycle time distribution (and thus the blocking and starving time distributions) is thus important in the application of the decomposition. This ability has been shown earlier for probability mass fitting (see Section 5.3). This leads to think that PMF is well suited to the application of the decomposition method, and would provide good results.

In the following, in Section 6.1, we briefly present the decomposition technique we implemented, and its application using probability mass fitting. Two network types are considered: tandem queues and fork-join queueing networks. Split-and-merge QN are not studied in this chapter, even if the decomposition could as
easily be applied to them (see Perros [71], Altiok and Perros [6] or Lee et al. [57] for example). In Section 6.2, we illustrate, by computational experiments and examples, how the modelling method coupling decomposition and PMF behaves.

6.1 Modelling Method

In this section, we present the decomposition technique we implemented, coupled with PMF. We first explain it for tandem queues and then extend it to fork-join queueing networks.

6.1.1 Tandem Queues

We first present the decomposition technique applied to tandem queues, when it follows PMF, i.e. with discrete phase-type service time distributions. For more information on the tandem queue decomposition, the reader is referred to the good review paper by Dallery and Frein [25], which gives an unified view on decomposition for tandem queues with exponentially distributed service times, or to the reference books by Perros [71], Gershwin [36] or Altiok [5]. Concerning the decomposition of tandem queues with phase-type service time distributions, founding papers are those by Altiok [4] (continuous PH) and Gun and Makowski [41] (discrete PH).

The main idea is to decompose the system into smaller subsystems, in order to decrease the computational cost of the modelling. Solving more, but much easier, subproblems, allows to approximately analyze the global system much more quickly and with good accuracy. In this work, we choose to decompose the tandem queue into two station subsystems (tandem queues). The equivalence with decomposition into single station queues with an arrival process has been shown (see Dallery and Gershwin [26]). Other decomposition methods decomposing the systems into three station subsystems also exist, and other various options exist. In other words, other decomposition methods could have been applied, the method presented here is one way of doing it.

The decomposition technique we implemented is illustrated on Figure 6.1, where a four station tandem queue is decomposed into three two station tandem queues. The original system is denoted $T$ and the subsystems are denoted $T(i)$ ($i = 1, 2, \ldots, m_b$, with $m_b$ the number of buffers). A subsystem $T(i)$ is made of two “virtual stations”: an upstream station $S_u(i)$ and a downstream station $S_d(i)$. There is one subsystem for each buffer in the original system. The buffer sizes of the subsystems are kept equal to the corresponding original buffer sizes. The service times of the virtual stations are modified so that the flow of items through the subsystem buffer is close to the flow in the original buffer (almost the same arrivals and departures, starvations, blockages, and buffer levels). In other words, upstream and downstream stations of each two station tandem queue summarize the effects of the entire upstream portion of the original system and the entire downstream portion of the original system, on the buffer (Dallery and Gershwin [26]). For example, on Figure 6.1, the service time of the virtual station $S_u(2)$
Figure 6.1: Decomposition of a four station tandem queue.

should be modified in order to aggregate the effect of the upstream portion of the system, i.e. from \( S_1 \) to \( S_2 \), while the service time of \( S_d(2) \) should aggregate the effect of the end of the system, i.e. from \( S_3 \) to \( S_4 \).

The goal is thus to smartly modify the service time distributions of the stations, so that they model the behavior of the upstream or downstream part of the system. Compared to an isolated two station subsystem, in the original system, the upstream part may cause starvation in a station. If the upstream part of the system has been slow, it may be unable to feed the station. Consequently, in order to include the effect of the upstream portion of the system, one has to modify the service time distribution of the virtual station \( S_u(i) \) to include the omitted starvation time. Similarly, compared to an isolated two station subsystem, in the original system, the downstream portion may cause blocking on a station, if this portion has been comparatively slow. The service time distribution of \( S_d(i) \) has thus to be modified in order to include the blocking time that would be caused by the downstream portion of the original system.

The starving time to be added to the original service time of station \( S_i \), in order to find the service time distribution of the virtual station \( S_u(i) \), can be deduced from the previous subsystem \( T(i - 1) \). Indeed, a starvation of station \( S_i \) in the original system corresponds to a starvation of the virtual station \( S_u(i) \) in the previous subsystem \( T(i - 1) \). The service time distributions of stations \( S_u(i) \) are thus iteratively updated, starting from the beginning of the system to its end \( (i = 1, 2, \ldots, m_b) \). The service time of \( S_u(i) \) is modified from the results computed on subsystem \( T(i - 1) \), i.e. it includes the starvation time computed on station \( S_d(i - 1) \), the subsystem \( T(i) \) is updated, it then allows to compute the starvation time on \( S_d(i) \), to update the service time of \( S_u(i + 1) \), and so on.

Similarly, the blocking time to be included in the service time of a virtual station \( S_d(i) \) corresponds to the blocking time on station \( S_u(i + 1) \) of the next subsystem \( T(i + 1) \). The service time distributions of stations \( S_d(i) \) are thus iteratively updated, starting from the end of the system \( (i = m_b, m_b - 1, \ldots, 1) \). The service time of \( S_d(i) \) is modified from the results of the computation of the
subsystem $T(i+1)$, i.e. it includes the blocking time computed on station $S_u(i+1)$. All in all, the service time distributions of the virtual stations are thus updated iteratively, according to a loop. The algorithm first goes forward and updates the service times of the upstream stations $S_u(i)$ (from the starving time of $S_d(i-1)$). It then goes backward and updates the service times of the downstream stations $S_d(i)$ (from the blocking time of $S_u(i+1)$). The subsystems $T(i)$ are thus progressively updated, in several loops, and then finally converge to an approximate model of the original system $T$. The throughput of the original tandem queue, for example, can be approximated by the throughput of a subsystem (when the algorithm converged, the throughput is equal for each of the subsystems).

In our case, the service time distributions are phase-type (while the decomposition method has originally been developed for exponential distributions). The phase-type distribution allows to compute finer estimations of the starving and blocking times to be added in the virtual station service times. We are able to compute a discrete distribution of the starving (resp. blocking) time, which basically gives the probabilities for the starving (resp. blocking) time to equal $n\tau$ ($n = 1, 2, \ldots, a$). Furthermore, the estimation can still be refined, as we can in fact compute the conditional probability of a discrete starving (blocking) time, knowing the value of the preceding service time. In other words, we are able to compute the probability for the starving time to equal $n\tau$, knowing that the preceding service time equals $n'\tau$. These conditional distributions of the starving time are computed on $S_d(2)$ for example, and consistently added to the original service time distribution of station $S_3$, to get the service time distribution of $S_u(3)$ by adding the blocking time estimated on $S_u(3)$, and so on.

We argued in Section 5.3 that the cycle time distribution, including the service time plus the blocking/starving time, can be accurately approximated by our modelling method, thanks to probability mass fitting. This modelling method, involving a state model, is used to analyze the subsystems and thus to estimate the conditional distributions of the starving and blocking time (to be included in the virtual station service times). This estimation is crucial in the decomposition technique and should thus be accurate.

6.1.2 Fork-Join Queueing Networks

The decomposition of fork-join QN has been less studied than the decomposition of tandem queues. Models with similar assumptions (finite buffers, phase-type distributions, reliable stations) are given by Altiok [5] or van Vuuren and Adan [95].

A fork-join queueing network can be decomposed in subsystems in a similar way than a tandem queue. A subsystem $T(i)$ is associated to each buffer, and made of two virtual stations: the upstream and downstream stations, $S_u(i)$ and $S_d(i)$. The service time distributions of the upstream and downstream stations are modified so that they summarize the effects of the entire upstream portion of the original system and the entire downstream portion of the original system, on the buffer. Going forward, the algorithm updates the service time distributions of the stations
6.1. MODELLING METHOD

Figure 6.2: Decomposition of a four station join (assembly) system.

$S_u(i)$, adding an estimation of the starving time due to the upstream portion of the system, estimation which has been computed on the previous subsystem $T(i-1)$ (station $S_d(i-1)$). Going backward, the service time distribution of stations $S_d(i)$ are modified in order to include the blocking time due to the downstream portion of the system, which is estimated on the subsystem $T(i+1)$ (station $S_u(i+1)$). The algorithm updates the service time distributions of the virtual stations in several loops until convergence.

The difference between the decomposition of a tandem queue and a fork-join QN comes from the following. In a FJQN, a join (assembly) station can be starved by several stations (each of its predecessors). When the algorithm goes forward, to update the service time of the stations $S_u(i)$, the omitted starving time to be added is thus deduced from several subsystems, namely the subsystems corresponding to the preceding buffers. An example is given on Figure 6.2. On this example, the algorithm first analyse subsystems $T(1)$ and $T(2)$. This allows to evaluate the starving caused by $S_1$ and $S_2$ on the join station $S_3$. Subsequently, the starving caused by both predecessors can be consistently added to the service time of the virtual station $S_u(3)$. The subsystem $T(3)$ can then be analysed, and so on.

Similarly, a fork (disassembly) station can be blocked by each of its successors. When the algorithm goes backward, to update the service time of the stations $S_d(i)$, the omitted blocking time to be added is thus deduced from several subsystems, i.e. the subsystems corresponding to the succeeding buffers. On Figure 6.3 for example, the algorithm analyse subsystems $T(2)$ and $T(3)$ in order to estimate the blocking caused by $S_3$ and $S_4$ on the fork station $S_2$. Subsequently, the block-

\footnote{If $S_3$ is starved by its two preceding stations, the starving will end when each station will have ended its job (see the synchronization constraints, term (4.7) in Lemma 4.2). The starving time to be added to $S_u(3)$ is thus given by the maximum of the starving times computed on $S_d(1)$ and $S_d(2)$.}
ing caused by both successors is consistently\(^2\) added to the service time of the virtual station \(S_d(1)\). The subsystem \(T(1)\) can then be analysed, the algorithm can go forward again, and so on.

Like for tandem queues, phase-type service time distributions allow to get more detailed estimations of the blocking and starving time distributions (compared to exponential service time distributions), while these distributions are crucial in the decomposition technique. Moreover, probability mass fitting promises accurate estimations of these distributions.

### 6.1.3 Complexity

The motivation behind the decomposition technique is to decrease the computational cost of the modelling of queueing networks. On the opposite, we showed in Chapter 3 that the complexity is the main limitation of a state model. Using decomposition, only small subsystems, i.e. two station tandem queues, are analysed. They are analysed several times in the iterative loop, but are much quicker to analyse than the original system.

In Section 3.1.1, we show that the computational cost of the analysis of a two station tandem queue \(T(i)\) is proportional to \((b_i + 1)(a_{ui} + 2)(a_{di} + 2)\), where \(a_{ui}\) (resp. \(a_{di}\)) is the number of discrete values in the distributions of the upstream (resp. downstream) station, and \(b_i\) is the buffer size. One subsystem corresponds to each buffer, and each subsystem is analysed \(2n_{iter}\) times, where \(n_{iter}\) gives the number of loops (forward and backward steps both included). The computational cost of the decomposition algorithm is thus proportional to

\[^{2}\text{If } S_2 \text{ is blocked by its two successors, the blocking will end when each of them will have ended its job (see the synchronization constraints, term (4.9) in Lemma 4.2). The blocking time to be added to } S_d(1) \text{ is thus given by the maximum of the blocking times computed on } S_u(2) \text{ and } S_u(3).\]
6.1. MODELLING METHOD

Table 6.1: Computational time (in seconds) of the decomposition, for tandem queues, using PMF without instant job, and with a total storage space equal to two. The number \( m \) of stations and the number \( a \) of discretization steps vary. The computational time of the state model is given under parenthesis.

<table>
<thead>
<tr>
<th>( a )</th>
<th>( m = 2 )</th>
<th>( m = 3 )</th>
<th>( m = 4 )</th>
<th>( m = 5 )</th>
<th>( m = 10 )</th>
<th>( m = 20 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.1 (0.1)</td>
<td>0.2 (0.1)</td>
<td>0.2 (0.3)</td>
<td>0.3 (4.8)</td>
<td>0.7</td>
<td>1.1</td>
</tr>
<tr>
<td>6</td>
<td>0.1 (0.1)</td>
<td>0.2 (0.2)</td>
<td>0.3 (1.2)</td>
<td>0.4 (41)</td>
<td>0.8</td>
<td>1.3</td>
</tr>
<tr>
<td>8</td>
<td>0.1 (0.1)</td>
<td>0.3 (0.3)</td>
<td>0.4 (4.7)</td>
<td>0.4 (240)</td>
<td>0.8</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 6.2: Computational time (in seconds) of the decomposition, for three station tandem queues, using PMF without instant job, and with a total storage space equal to two. The total storage space \( B_\Sigma = \sum b_i \) and the number \( a \) of discretization steps vary. The computational time of the state model is given under parenthesis.

<table>
<thead>
<tr>
<th>( a )</th>
<th>( B_\Sigma = 2 )</th>
<th>( B_\Sigma = 4 )</th>
<th>( B_\Sigma = 6 )</th>
<th>( B_\Sigma = 8 )</th>
<th>( B_\Sigma = 10 )</th>
<th>( B_\Sigma = 20 )</th>
<th>( B_\Sigma = 40 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.2 (0.1)</td>
<td>0.2 (0.1)</td>
<td>0.2 (0.3)</td>
<td>0.2 (0.4)</td>
<td>0.2</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.2 (0.2)</td>
<td>0.3 (0.3)</td>
<td>0.3 (0.5)</td>
<td>0.3 (1.4)</td>
<td>0.3 (2.3)</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>8</td>
<td>0.3 (0.3)</td>
<td>0.3 (0.6)</td>
<td>0.4 (2.9)</td>
<td>0.4 (8.1)</td>
<td>0.4 (44)</td>
<td>0.7</td>
<td>1.2</td>
</tr>
</tbody>
</table>

\[
2n_{iter} \sum_{i=1}^{m_b} (b_i + 1)(a_i + 2)(a_{i+1} + 2). \]

This can be compared to the complexity when a state model is used: \( \prod_{i=1}^{m_b}(a_i + n_{e;i}! + n_{f;i}) \cdot \prod_{i=2}^{m_b}(b_i + 1) \) (see Section 3.2). It can be seen that the decomposition does not suffer from an explosion of its computational cost when the system size increases, unlike the state model\(^3\).

In tables 6.1 and 6.2, we give the actual computational time\(^4\) needed by the decomposition method. Table 6.1 shows the impact of the number of stations while Table 6.2 shows the impact of the total storage space. On both tables, it can be seen that the computational time keeps low even for large system sizes. Compared to the state model (under parenthesis, also see tables 3.1 and 3.2), the computational cost is much smaller for large systems. The decomposition becomes better, in terms of computational time, from four stations and from a total storage space of size four. Also note that, in all our experiments, the number of iterations of the decomposition method is quite small: it is around five, independently of the parameters \( a, m \) or \( \sum b_i \).

The benefit brought by the decomposition in terms of computational cost is clear. However, this improvement is, of course, made at the expense of the accuracy of the performance evaluation (and the loss of the bounding property). In the following section, we evaluate the accuracy of the performance evaluation by computational experiments.

\(^3\)If we suppose, to be clearer, that \( a \) and \( b \) represent “average values” for the number of steps and the buffer sizes, and that \( m \approx m_b \), we get that the computational cost is proportional to \( 2n_{iter}m_ba^2 \) using decomposition, it is proportional to \( a^m b^m \) using a state model.

\(^4\)Using the Gaussian elimination implemented in MATLAB\textsuperscript{R\textregistered} on a 2.16 GHz usual PC, 2 GB RAM, Intel\textsuperscript{R\textregistered} Centrino Duo processor.
CHAPTER 6. DECOMPOSITION

Tandem Fork Join

<table>
<thead>
<tr>
<th>a</th>
<th>m = 3</th>
<th>m = 4</th>
<th>m = 3</th>
<th>m = 4</th>
<th>m = 3</th>
<th>m = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.69%</td>
<td>1.14%</td>
<td>0.96%</td>
<td>1.38%</td>
<td>0.93%</td>
<td>1.37%</td>
</tr>
<tr>
<td>6</td>
<td>0.53%</td>
<td>1.05%</td>
<td>0.78%</td>
<td>1.30%</td>
<td>0.76%</td>
<td>1.28%</td>
</tr>
<tr>
<td>8</td>
<td>0.51%</td>
<td>1.01%</td>
<td>0.74%</td>
<td>1.28%</td>
<td>0.71%</td>
<td>1.25%</td>
</tr>
<tr>
<td>20</td>
<td>0.52%</td>
<td>1.02%</td>
<td>0.74%</td>
<td>1.29%</td>
<td>0.71%</td>
<td>1.27%</td>
</tr>
</tbody>
</table>

Table 6.3: Average relative error reached by the decomposition, using PMF without instant job with $\alpha/\tau = 0.5$, the number of steps $a$ and the number of stations changing.

6.2 Computational Experiments

The modelling method, coupling probability mass fitting and the decomposition technique, has been implemented, for fork-join queueing networks of any topology (i.e. with queues in series, fork stations and join stations in the same network). Note that the tandem queue is a particular case of FJQN. In this section, we test the accuracy of the method by computational experiments.

6.2.1 Comparison on Small Systems

To begin, we study the set of experiments which has already been used in sections 4.2.2 and 5.1.1 in order to allow comparison with the state model. We now briefly recall the characteristics of this test set. The analysed networks have three or four stations (tandem, fork or join topologies). The global storage space of a network goes from zero to four and is balanced among the buffers. The service time distributions are chosen randomly from the ten distributions used throughout the text (see Figure 2.5). In total, 1500 networks (50 for each storage space) were analysed. The best PMF parameter choice, as found previously, i.e. PMF/nIJ with $\alpha/\tau = 0.5$, is used (to analyse the subsystems). The number $a$ of discretization steps equals 4, 6, 8 or 20 (20 is possible thanks to the low complexity of decomposition). The results were compared to simulation results. The obtained average relative errors are given in tables 6.3 and 6.4. Note that the errors presented here are the errors made by the global modelling method, on queueing networks with general service time distributions. The errors have thus two components: the error brought by the probability mass fitting, and the error brought by the decomposition. In the previous chapters, as a state model is exact, the errors we presented were just made of the error coming from the PMF discretization.

First of all, Table 6.3 shows that the decomposition method leads to good accuracy. The relative error is on the order of one percent. It can also be seen that the topology (tandem, fork or join) has little impact. On the opposite, the accuracy level tends to deteriorate when the number of stations increases. Table 6.4 shows the impact of the buffer sizes. The decomposition approximation improves when the storage space increases. This is not surprising as the blocking and starving time decrease when the storage space increases (the decoupling between the stations gets stronger), and as the main approximation concerns them when a system is
6.2. COMPUTATIONAL EXPERIMENTS

Table 6.4: Average relative error reached by the decomposition, using PMF without instant job with $\alpha/\tau = 0.5$, the number of steps $a$ and the storage space changing ($B_\Sigma$ stands for $\sum b(i,j)$).

<table>
<thead>
<tr>
<th>$a$</th>
<th>$B_\Sigma = 0$</th>
<th>$B_\Sigma = 1$</th>
<th>$B_\Sigma = 2$</th>
<th>$B_\Sigma = 3$</th>
<th>$B_\Sigma = 4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2.75%</td>
<td>0.97%</td>
<td>0.55%</td>
<td>0.57%</td>
<td>0.56%</td>
</tr>
<tr>
<td>6</td>
<td>2.93%</td>
<td>0.90%</td>
<td>0.32%</td>
<td>0.23%</td>
<td>0.29%</td>
</tr>
<tr>
<td>8</td>
<td>2.93%</td>
<td>0.90%</td>
<td>0.28%</td>
<td>0.22%</td>
<td>0.24%</td>
</tr>
<tr>
<td>20</td>
<td>2.99%</td>
<td>0.93%</td>
<td>0.25%</td>
<td>0.22%</td>
<td>0.23%</td>
</tr>
</tbody>
</table>

decomposed. Also note that the decomposition approximation is clearly less good with zero buffers. Moreover, there is a surprising, and unexplained, phenomenon in this case: the approximation is better with few discretization steps.

Otherwise, the relative error made by the global modelling method improves when the PMF discretization refines. However, the impact of the number of discretization steps $a$ is quite low. This can be understood from the fact that the global error is composed of the PMF error and of the decomposition error. The PMF error decreases with $a$ increasing but is significantly smaller than the decomposition error. The decreasing PMF error has thus little impact on the global error. This observation also tends to show that the decomposition error is independent of the number of discretization steps $a$. In other words, the decomposition approximation, and more precisely the approximation of the starving and blocking distributions, do not improve when the PMF discretization of the service time distributions refines. This is consistent with the observation made in Section 5.3.2: the approximation of the cycle time distribution is already good with few discretization steps (see Figure 5.3). From $a = 8$, and even $a = 6$ in most cases, the PMF error seems to have no impact on the global error anymore. In these cases, the relative errors shown on tables 6.3 and 6.4 give the decomposition error. In our experiments, it thus makes no sense, when the decomposition is used, to discretize the distributions with more than 6 or 8 steps (as the approximation error does not decrease further on).

Tables 6.3 and 6.4 can be compared to tables 5.8 and 5.10 where the accuracy of the modelling method coupling PMF with an exact state model is shown. This modelling method is of course more accurate, particularly for the higher number of discretization steps ($a = 8$), and with a small buffer storage space. We also recall that this methodology leads to bounds, as shown in Chapter 4. The modelling method implying decomposition is thus less accurate and loose the bounding property but is far less computationally expensive for large systems.

6.2.2 Larger Fork-Join QN

The set of experiments used in the previous section is the same than the one used in previous chapters, to allow comparison. However, decomposition allows to analyse larger systems. We now show how the decomposition method, using PMF, behaves on larger fork-join queueing networks. For this, we analyse fork-join QN with the topology shown on Figure 1.5, i.e. FJQN with a disassembly station as
well as an assembly station, with eight stations and seven buffers. We run five sets of hundred experiments. For each experiment, the buffer sizes are randomly chosen in the interval \([2(i-1), 2i]\), where \(i = 1, 2, \ldots, 5\) is the index of the set of experiments (i.e. the buffer sizes are chosen among 0, 1 or 2 in the first set, 2, 3 or 4 in the second set, etc.). The distributions are chosen among the set used in the previous experiments (see Figure 2.5). For these 500 networks, we apply the decomposition method, using PMF/nIJ with \(\alpha/\tau = 0.5\), and \(a = 4\) or 6, and compare the results to simulation results.

The accuracy level reached on the cycle time by the decomposition is given in Table 6.5. The table reveals that, even for large networks, the accuracy is (very) good. In fact, the accuracy does not deteriorate when the system size increases, as could be thought from Table 6.3. On the contrary, Table 6.5 confirms what was observed on Table 6.4: the accuracy is less good for small buffer sizes. On all experiments, with any storage space, we get an average relative error of 0.78% (0.52%) with 4 (6) discretization steps. For buffer sizes larger than two, the relative errors is around 0.3% with six discretization steps. These experiments confirm the interest of decomposition: it allows to accurately evaluate the performance of large networks in a short computational time (given in italic in Table 6.5).

Ideally, we would like to compare these results to the results of other methods, and particularly to other decomposition methods. Unfortunately, we did not find results in the literature that are readily comparable. As already said, most papers do not consider the distribution fitting, but focus on the analytical modelling. However, we argued that the decomposition error can be isolated in the presented results. They could thus be compared to the errors found in the literature, but, again, few comparable results are available. Most papers suppose exponential distributions. This leaves us with few papers analysing queueing networks with phase-type distributions and similar assumptions (finite buffers in particular).

Altiok [4] provides four examples of three station tandem queues under saturation, with buffers of size three. The decomposition he propose leads to average errors ranging from 2.31% to 6.87%. These errors can be compared to the average relative errors presented on Table 6.4 (in Altiok’s examples, \(B_1 = 6\)). Altiok [4] also provides three other examples of unsaturated tandem queues (Poisson arrivals and finite buffer in front of the first station). The relative errors are 1.3% (buffer sizes of 5, 8 and 3), 4.26% (buffer sizes of 5, 3 and 2), and 3.69% (5 stations, buffer

<table>
<thead>
<tr>
<th>(a)</th>
<th>(0 \leq b_i \leq 2)</th>
<th>(2 \leq b_i \leq 4)</th>
<th>(4 \leq b_i \leq 6)</th>
<th>(6 \leq b_i \leq 8)</th>
<th>(8 \leq b_i \leq 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.39%</td>
<td>0.48%</td>
<td>0.65%</td>
<td>0.70%</td>
<td>0.70%</td>
</tr>
<tr>
<td>6</td>
<td>1.40%</td>
<td>0.26%</td>
<td>0.30%</td>
<td>0.33%</td>
<td>0.29%</td>
</tr>
</tbody>
</table>

Table 6.5: Average relative error reached by the decomposition on large fork-join QN \((m = 8)\), using PMF without instant job with \(\alpha/\tau = 0.5\), the number of steps \(a\) and the storage space changing (\(b_i\) stands for the size of buffer \(i, i = 1, \ldots, m\)). The computational time is given in italic.
sizes larger than 2). In his book, Altiok [5] give two other examples of unsaturated tandem queues (Poisson arrivals and infinite buffer in front of the first station). On these examples, his decomposition method leads to relative errors of 0.35% (three stations, buffer sizes of three) and 1.02% (five stations, buffer sizes larger than four). Gun and Makowski [41] do not give any results on the cycle time approximation. van Vuuren and Adan [95] propose and test a decomposition method for assembly (join) systems (without queues in series, and fork stations). They apply the method on an experiments set of 768 assembly systems, with 3, 5 or 9 stations and buffers of sizes 0, 2, 4 or 8, and report an average relative error of 1.5% (distribution fitting error not included). Some other references can be cited as limiting cases, even if the queueing networks assumptions are pretty different. Brandwajn and Jow [21] were among the first authors to propose a decomposition method. They studied tandem queues with blocking and exponential service time distributions. On nine examples, their decomposition technique lead to a 0.89% average relative error, with buffer of sizes larger than 2 on all examples. The results brought by the generalized expansion method can also be compared. Andriansyah et al. [7] analyse bufferless multi-server queueing networks with exponential service time distributions. Their networks are not saturated, but are fed by a arrival process. They report a average relative error of 2.5% on about twenty examples.

Even if this comparison is quite limited, it tends to show that probability mass fitting brings an improvement in the application of decomposition. As shown in Section 5.3, probability mass fitting allows to accurately approximate the cycle time distribution. The approximation of the starving and blocking time (which are part of the cycle time) distributions is crucial in the decomposition technique. Consequently, PMF allows a fine approximation of the starving and blocking time and, then, a more accurate decomposition of the system.

6.2.3 Cycle time Distributions

One of the important advantages of the modelling method described in Chapter 3, implying a state model, is that it allows to compute good approximations of the cycle time distributions (see Section 5.3). In the present section, we check if this ability stays true when decomposition is used instead of a state model. The cycle time distributions are simply computed on the subsystems (with the modified service time distributions), which are analysed by a state model, so that the computation explained in Section 5.3.1 can be done. In the following, we compare this estimation to simulation results on some examples.

To begin, we go back to the first example given in Section 5.3.3. We analyse a three station tandem queue with uniform service time distributions and no storage space. Figure 6.4 shows the approximation of the cycle time distribution of the last station, using decomposition (left-hand side), which can be compared to the result of a simulation (right-hand side). It can be seen that the approximation is quite good. However, it is not as good as the result obtained with a state model, shown

\footnote{To allow better comparison with Table 6.5, the average relative errors are 2.35%, 1.68%, 1.05% and 0.82%, for buffers of sizes 0, 2, 4 or 8.}
Figure 6.4: Cycle time distribution of the third and last station of a tandem queue (uniform service time distributions, no storage space). The left-hand side gives the distribution computed by the decomposition using PMF/nIJ \((\alpha/\tau = 0.5)\) with 10 discretization steps. The right-hand side gives the distribution computed via simulation.

Figure 6.5: Cycle time distribution of the last station of a large fork-join QN (uniform service time distributions, buffers of size 2). The left-hand side gives the distribution computed by the decomposition using PMF/nIJ \((\alpha/\tau = 0)\) with 10 discretization steps. The right-hand side gives the distribution computed via simulation.
6.2. COMPUTATIONAL EXPERIMENTS

Figure 6.6: Cycle time distribution of the last station of a large fork-join QN (beta(2,2) service time distributions, buffers of size 4). The left-hand side gives the distribution computed by the decomposition using PMF/nIJ ($\alpha/\tau = 0$) with 10 discretization steps. The right-hand side gives the distribution computed via simulation.

on Figure 5.5 (see the values around one for example). This is not surprising, as we already observed that the decomposition is less accurate, and particularly with buffers of size zero.

We now test the accuracy of the estimation of the cycle time distribution on examples with larger storage space. We also choose to compute it on a larger fork-join queueing network, namely the FJQN used in the previous experiments set, and shown on Figure 1.5. On the example illustrated on Figure 6.5, the buffer sizes equal two and the service times are uniformly distributed. Figure 6.6 shows the cycle time distribution estimation for the case where the buffer sizes equal four and the service times distributions are beta(2,2). It can be seen, when compared to the simulation result, that the distribution estimation is good, in its shape and values. Unsurprisingly, the approximation is better in these examples, as the buffer sizes are a little larger than zero.

The ability to compute a good approximation of the cycle time distribution is an important advantage of probability mass fitting, compared to other concurrent methods (moments fitting in particular). It is true when a state model is coupled to PMF and also when the decomposition technique is applied (at least for non-zero buffer sizes). The distribution provides more information on the behavior of the system, compared to the isolated expectation. Percentiles, for example, help the manager to evaluate the risk and the service level of the system.
Chapter 7

Conclusions

We now review the main results presented in the thesis and discuss some future perspectives for future research.

7.1 Results Achieved

In this thesis, we are interested in the **modelling of queueing networks with finite buffers and general distributions**. Such models have proven to be very useful to model complex systems in various applications fields (production, computer and communication networks, healthcare management, etc.). They essentially aim to evaluate the performance measures, and get insights on the behaviour of the system from these measures. In this thesis, we focus on tandem queues, fork-join and split-and-merge queueing networks. The modelling process of queueing networks with general service time distributions can be seen as made of three steps: data collection, distribution fitting, and analytical modelling. From the collected data, an analyst has to build distributions which are tractable for the analytical method, most often phase-type distributions so that the Markov theory can then be applied.

Our main originality lies in the second step of the modelling process. In Chapter 2, we introduce a new distribution fitting method, called **probability mass fitting** (PMF). PMF builds a discrete phase-type distribution from the original distribution (which is possibly in the form of a histogram). The idea is the following: the probability of each discrete value equals the original probability mass around this discrete value. More formally, PMF transforms a given distribution into a discrete one by aggregating the probability mass distributed in the interval \(((j - 1)\tau + \alpha,j\tau + \alpha]\) on the point \(j\tau\) (where \(j = 0,1,\ldots,a\)), where \(\tau\) is the width of the intervals on which the probability masses are computed, \(\alpha\) is the shift parameter, and \(a\) is the number of non-zero discrete values in the resulting distribution. This PMF is called PMF with instant jobs (PMF/IJ) as service times of length zero can potentially occur. The latter sensibly complicate the subsequent modelling, leading us to propose an alternative PMF: PMF without instant job
(PMF/nIJ). Compared with PMF/IJ, PMF without instant jobs is just modified in order to remove the possibility of instant jobs: the original probability mass on the interval $[0, \alpha]$ is aggregated on $\tau$ instead of 0.

We showed some nice properties of probability mass fitting (with or without instant job). From its definition, it can be seen that, when discretized, a service time cannot be shifted outside a limited interval around it. PMF is thus bounding: an original service time can be bounded, from below and from above, using its discretized value. The lower bound is less tight in the PMF/nIJ case as the first integrating interval is larger. Moreover, the evolution of the discretized service times according to the shift parameter $\alpha$ is a monotonic, decreasing, function. This allows to show that a parameter $\alpha$ can be found so that the expectation of the distribution is conserved, if the original distribution function is continuous (plus another condition on the expectation in the PMF/nIJ case).

Probability mass fitting shows various other advantages. It is simple and intuitive. It preserves the shape of the distribution. This characteristic is even a motivation of PMF and makes an important difference with moments fitting for example. PMF suits well when data is collected in the form of a histogram, the most common form in practice. It makes full use of the information in a histogram. PMF is intelligible, i.e. the transformation of each original service time to the corresponding discretized time is exactly known. The PMF approximation is refined when the integrating interval size $\tau$ is decreased and, at the limit, it tends to the exact distribution. It means that the accuracy of the approximation can be chosen according to the affordable computational effort. Finally, as it builds discrete PH distributions, PMF is able to fit distributions with a low variance, a delay, abrupt changes, or deterministic values.

However, PMF does not, in general, conserve the moments, and it requires distributions with a finite support. The restrictive nature of these characteristics can be discussed. In particular, the support of a service time distribution is most often finite in practice and the ability to approximate such distributions can be seen as an advantage. The main weakness of PMF is thus in the fact that it is badly suited for service time distributions with rare events.

The probability mass fitting being well defined, the goal of the rest of the thesis is to study its effect on the global modelling process of queueing networks with blocking. In Chapter 3, we present a modelling method which applies a state model on the system modified using PMF. From the discrete service time distributions built by probability mass fitting, the evolution of the modified system is exactly described by a Markov chain, whose states are the possible combinations of the stages of the various stations and the contents of the various buffers. The performance measures of the system (cycle time, work-in-progress, flow time, etc.) can then be estimated from the chain. The method is implemented for tandem queues, fork-join QN and split-and-merge QN. The main limitation of this method lies in its complexity: the state space size increases very quickly when the system size increases. The Markov chain modelling the system is more complicated when PMF with instant jobs is used, as instant jobs multiply the number of possible transitions. The transition matrix is thus less sparse, and the computational time significantly increases.
7.1. RESULTS ACHIEVED

As a first result, this modelling method allows to compute bounds on the cycle time. For queueing networks with general service time distributions, bounds are the only exact information available. They give some certainty when evaluating the performance of queueing networks, while simulation as well as available analytical methods offer little certainty about the quality of their estimations. Very few bounding methodologies have been proposed for queueing networks with general service time distributions. In Chapter 4, we propose two methodologies.

The first bounding methodology essentially extends the bounding property of probability mass fitting. The bounds are proved using the concept of critical path, i.e., the sequence of jobs which covers a run, without gap and without overlap. The critical path allows to express the running time as a sum of service times and thus to extend the bounding property on service times to the throughput. We get upper and lower bounds on the throughput, in transient and in steady-state (cycle time). The interval between both bounds is directly proportional to the discretization step size $\tau$. The bounds are thus refinable: they become tighter when the PMF discretization is refined. Moreover, their accuracy can be a priori chosen. The lower bound is less good using PMF without instant jobs than using PMF with instant jobs, as the first integrating interval is larger using PMF/nIJ.

From computational experiments, we observe that the bounds are not really tight with a reasonable number of discrete values (accuracy in the order of 10%). We also see that the bounds improve when the shift parameter $\alpha$ increases, and that grouping at the beginning ($\alpha/\tau = 1$) thus leads to the better bounds.

The second bounding methodology is based on the critical path computation. We show that the critical path can be computed in the discretized time. The same sequence of jobs is non-overlapping in the original time. The sum of the lengths of these jobs is thus shorter than the running time and leads to a lower bound on the cycle time. Moreover, the tightness of the bound can be argued from the fact that the critical path in the discretized time is close to the one in the original time (which gives the exact running time). This thus leads to a computable, tight and refinable lower bound on the cycle time of queueing networks. The tightness of the bound is shown on computational experiments. On average, the level of accuracy reached is 0.4% with eight discretization steps, 0.7% with six and 1.4% with only four steps. Concerning the configuration of the network, while the topology does not matter, the tightness of the bound seems to deteriorate when the number of stations increases, and to improve when the storage space increases. Compared to the few bounding methodology available in the literature, our second methodology compares favorably in terms of accuracy and both bounding methodologies compare favorably in terms of the scope of application (more configurations and buffer sizes).

Both bounding methodologies are valid for fork-join queueing networks and for split systems. They are not valid for merge systems because, in the original time, an overlap can appear in the sequence of jobs defining the critical path in the discretized time. Furthermore, the bounds lead quite easily to bounds on other related, performance measures: the idle time, the utilization and the proportion of idle time in a given station. Finally, we also extend the monotonicity of probability mass fitting to the global modelling method. The evolution of the cycle
time computed in the discretized time is a monotonic, decreasing, function of the
shift parameter $\alpha$. Moreover, using the same integrating step size $\tau$, grouping at
the beginning (PMF/IJ with $\alpha = 1$) leads to a better lower bound, while grouping
at the end (PMF with $\alpha = 0$) leads to a better upper bound (applying the first
methodology).

In Chapter 5, we study the performance evaluation, i.e. the estimation of
the performance measures. We first propose two approximations of the cycle time,
reducing the error made on the service time, while the error on
the idle time remains. Extensive computational experiments are then conducted,
leading to various interesting observations. They reveal that the natural solution,
consisting in the direct computation with PMF/nIJ and $\alpha/\tau = 0.5$, is the best
choice. In other words, the best results are obtained with grouping at the middle,
and avoiding instant jobs (which significantly increase the computational cost,
without clear benefit). The experiments show that the method computes accurate
estimations and that the latter improve when PMF is refined. For fork-join QN,
average relative errors are in the order of a few tenths of a percent (0.2% with
$\alpha = 6$ discrete values). Moreover, the accuracy level shows to be remarkably
stable when the system’s configuration changes, i.e. in its topology, its number of
stations, its buffers sizes, and its service time distributions. For split-and-merge
QN, the method performs less nicely, notably because of the coincident events due
to discrete service time distributions. However, the approximation is still accurate,
particularly for split networks (0.5% for split, 1.25% for merge, with $\alpha = 6$). The
accuracy tends to deteriorate when the system becomes more complex (in terms of
the number of stations and the storage space), except for split networks for which
the number of stations has little impact. Note that the reported errors reveal the
errors caused by the PMF as the subsequent analysis is exact.

A third approximation of the cycle time takes advantage of the critical path
computation. The critical path allows to express the cycle time only in terms of
service times, without idle times on which we have little knowledge. However,
this approximation suffers from high complexity and does not make sense when
using grouping at the middle ($\alpha/\tau = 0.5$). It is only interesting in the particular
cases where one chooses $\alpha/\tau \neq 0.5$, and leads to a significant improvement of the
estimations in these cases. This approximation is illustrated on a small managerial
example, where it is used to assess the profit of a storage space enlargement.
Furthermore, the proposed method offers a quite complete modelling of the system.
Other performance measures can thus be computed. We show the accuracy of
the estimations of the work-in-progress and of the flow time on computational
experiments. The average relative error is in the order of some tenths of a percent,
except for merge networks, for which the method is again less accurate. The
accuracy improves when the PMF is refined. Moreover, it tends to improve when
the number of stations increases (the effect of the buffer sizes is mixed).

Beside the average performance evaluation, the modelling method is able to
estimate the cycle time distributions. The latter are computed in the discretized
time as discrete phase-type distributions. We argue that the shape conservation
of PMF extends to the cycle time distributions, as the shape of the discrete es-
timation is the same with 5, 10 or 20 discretization steps. The accuracy of the
distribution estimations is also illustrated on several examples, for tandem queues, fork-join QN and split-and-merge QN. The computation of a good cycle time distribution estimation is a significant advantage of our method, compared to other fitting methods which are unable to do so. The distribution offers more detailed information on the behavior of the system, compared to the isolated expectation. The percentiles, for example, are crucial information in order to help managers to evaluate the risk and the service level. We believe this advantage is source of interesting future prospects (see below).

We conclude this chapter with a comparison of PMF with other distributions fitting methods, particularly with the main ones: moments fitting and likelihood maximization. First, PMF leads to bounds and distribution estimations which are unreachable with other methods. Concerning the number of phases, i.e. the number of values in the discretized distributions, we show that it is reasonable using PMF, in comparison with other methods. This number even seems smaller for PMF when fitting non “continuous phase-type behaved” distributions (low variability, abrupt changes, etc.). Concerning the accuracy of the performance evaluations, some papers demonstrate that moments fitting can lead to (very) large errors on single queues performance estimations. We propose a simple example where two tandem queues with service time distributions sharing the same two first moments lead to cycle times with a 1.62% difference. We show that our method clearly differentiate both systems while a modelling based on moments fitting would not see any difference. Finally, we also applied a basic closed-form method for two station tandem queues, in order to compare its results to ours. Our method show to be more accurate (0.24% compared 5.7%) and more reliable.

The main weakness of the previous modelling method, implying an exact state model, lies in its complexity, which limits the applicability of the method. In Chapter 6, we study a potential solution in this matter: the application of decomposition after probability mass fitting, instead of a state model. We develop and implement the method for tandem queues and fork-join QN. A queueing network is decomposed into two station subsystems and the service time distributions of the virtual stations are iteratively modified to approximate the impact of the rest of the network, adding estimations of the blocking and starving distributions. The ability of PMF to reliably approximate the cycle time distributions (and thus the blocking and starving time distributions) is thus an important advantage of PMF for the application of the decomposition method. Applying decomposition, the computational time is drastically decreased, compared with a state model. It does not explode when the system size increases.

On computational experiments, the decomposition method, coupled to PMF, reveals to provide accurate estimations. The accuracy improves when the storage space increases (it is sensibly less good with zero buffers). The global error is made of the PMF error and the decomposition error. From 6 or 8 discretization steps, the PMF error has no impact on the global error, which is essentially composed of the decomposition error. It thus makes no sense to further refine the PMF. On a set of large fork-join queueing networks, the average relative error equals 0.3% with buffer sizes larger than two, and 1.4% with buffer sizes smaller than two. Compared with the few available results in the litterature, it seems that probabil-
ity mass fitting has a positive impact in the application of decomposition, due to its ability to accurately approximate cycle time distributions. Moreover, we show on some examples that, even using decomposition, the cycle time distributions can be fairly approximated. The estimations are less accurate than using a state model (particularly with small buffers) but are still good.

All in all, we believe that probability mass fitting can be considered as a valuable alternative in order to build tractable distributions for the analytical modelling of queueing networks. PMF allows to compute bounds and accurate estimations of the average of the performance measures as well as of their distributions. Together with the bounds, the approximations allow to get a good grasp on the exact measure with certainty. The estimations of the distributions offer more detailed information, which is useful in practice. This ability of PMF leads also to good results when decomposition is used, in order to increase the applicability of the analysis. Moreover, the required number of phases is reasonable. PMF does not need more computational effort than concurrent methods. Finally, PMF has the advantage to be refinable, so that the accuracy can be chosen according to the affordable computational cost, and intuitive.

7.2 Future Prospects

Many future research perspectives exist for probability mass fitting. PMF is a method to build tractable distributions. We analysed its effect on the modelling of queueing networks, and were bound to focus on some particular configurations. However, the application of the ideas proposed in the thesis are not limited to these particular configurations and assumptions.

Queueing networks with loops or closed queueing networks could be analysed. The PMF discretization can be applied on the service time distributions and the evolution of the modified system can be modelled using a Markov chain. The performance measures can then be evaluated (their average as well as their distributions). Concerning the bounds, their proofs make use of the notion of critical path. As its existence essentially relies on the fact that a job’s start is always caused by another job’s ending, the critical path can be extended to these networks. We believe that the bounds should also extend for closed systems with a fork-join type of organization. By fork-join type of organization, we mean with one buffer on each link between two stations, which synchronize the predecessors and the successors of a station. This characteristic prevents overlaps in the sequence of jobs making a critical path when changing from discretized to original time (see the discussion about the extension of the bounds to merge systems, Section 4.4). Consequently, the bounding methodologies should extend in this case. However, it needs more investigation in order to be proved.

The ideas presented in this thesis could also be extended to multi-server queueing networks. PMF could be used in the distributions fitting stage of their modelling. Unfortunately, the modelling of the modified system by a Markov chain would suffer from high complexity: the stage of service of each server, for each station, has to be included in the Markov chain state. However, this modelling can be done, and the performance measures can be evaluated. To analyse larger
7.2. FUTURE PROSPECTS

systems, approximate methods could also be developed. Concerning the bounds, again, the notion of critical path should extend as the beginning of a job on a server is always triggered by the end of another job’s end (on the same server or on a previous or subsequent station’s server). But, unfortunately, similarly to the case of merge systems, when a multi-server station precedes another station, the unstarving server is the first to finish its job. This server can be different in the discretized and in the original time, and an overlap may appear (see Section 4.4). This significantly lowers the chances to extend the bounds to multi-server queueing networks. A more extensive analysis should be performed in this matter.

Furthermore, it would be interesting to widen the scope of application of PMF by allowing infinite support distributions. One can think about adding a loop transition in the last state of the PH distribution for example, to model the tail of the distribution.

Besides the extension to other networks configurations, we could also be bring some technical improvements to the method and its implementation.

- We observed throughout the text that the performance evaluation is less accurate for merge networks. As explained in Section 3.3, in discrete time, two jobs in two predecessors of a merge station can end simultaneously, and a priority rule has thus to be implemented. We implemented a very simple rule: priority is given to the first stations. An option to improve the accuracy of the estimations for merge systems is to implement a more evolved priority rule, such as a random rule with equiprobable priority for each predecessor.

- In order to improve the accuracy of the decomposition method, we could still improve the estimation of the modified cycle time distribution. Indeed, in the current version of the algorithm, the starved time is added after the service time, as usually done in the literature. However, to get a better approximation of the cycle time, the starved time should be added before the service time.

- The resolution of the linear system of equation (to find the steady state probabilities) is made using the Gaussian elimination implemented in MATLAB\textsuperscript{\textregistered}. This could thus clearly be improved. The construction of the matrix via Kronecker products gives it a very particular structure, which can be exploited to speed up the resolution (see Stewart [78] and Dayar [29] for example). Moreover, iterative methods are usually applied to compute the steady-state probabilities. The initial solution, from which the method starts, is crucial for the speed of the resolution. In this context, we could first find a good initial solution with a coarse PMF discretization, and then use it to start the resolution of the system which has been discretized more accurately.

The ability to compute good estimations of the cycle time distributions is an important advantage of PMF, which could be further exploited. In practice, the distributions offer more detailed information on the behaviour of the system than average estimations, and are crucial to help managers to evaluate the risk and the service level. So, PMF is a tool which allows answering various managerial questions with a new point of view. With PMF, decisions can be evaluated from their effect on the distribution of the performance instead of only the average,
CHAPTER 7. CONCLUSIONS

giving rise to many attractive research problems. Decisions may concern the load of a production system, the inventory management in a supply chain, the routing in a computer network, or the dedication of surgical rooms, for example. Distributions of other performance measures could also be studied, such as the flow time or the throughput. An interesting and important open question in this matter also concerns the renewable property of the distributions.

Many more perspectives could be presented. We just cite some of them. The expansion method could be adapted and coupled to probability mass fitting (instead of a state model or the decomposition method). The evaluation of transient performance measures could be further studied. Our performance evaluation methods could be coupled with stochastic optimization algorithms, to solve problems like the buffer allocation or the network configuration.
Bibliography


