Performance evaluation of a solution for composite service selection problem with multiple consumers

Lynda Mokdad, Jean-Michel Fourneau, Abdelkrim Abdelli, Jalel Ben-Othman

Version of Record: https://www.sciencedirect.com/science/article/pii/S1569190X21000022
Manuscript_1b05f65a06e2a7d0023a56a38ba584c0

Performance evaluation of a solution for composite service selection problem with multiple consumers

Lynda Mokdad[a], Jean-Michel Fourneau[b], Abdelkrim Abdelli[c], Jalel Ben Othman[d]

[a] *Univ Paris Est Creteil, LACL, F-94010 Creteil, France*
[b] *Univ de Versailles, DAVID Lab*
[c] *Univ USTH, LSI Lab*
[d] *Univ Paris-Saclay, CNRS, CentraleSuplec*
Labo L2S, 91190, Gif-sur-Yvette, France
Univ Sorbonne Paris Nord

Abstract

In recent years, building business applications from independently developed services has become one of the current trends in service computing. To satisfy clients' requirements, service composition is performed to compose the various capabilities of available services. With the proliferation of services having similar functionalities, assessing the quality of a given composition is a paramount factor to decide which services must be selected, or to choose whether a given composition can provide the requested QoS (Quality of Service). However, due to the fluctuating conditions in the dynamic cloud computing environment, the QoS and the performances of the services become unreliable, and therefore call into the question the accuracy of the composition QoS. To tackle this issue, different methods that analyse the QoS have been developed, making it possible to help the designers to first, understand the system behaviour when providers and consumers, are interacting, thus allowing to optimize the system by identifying performance bottlenecks within a specified deployment environment.

In this study, the particular problem of many consumers that are competing to acquire services with same functionalities but with different QoS, is considered. For this purpose, assuming a dynamic environment, different

Email addresses: lynda.mokdad@u-pec.fr (Lynda Mokdad), jmf@uvsq.fr (Jean-Michel Fourneau), abdelli@lsi-usthb.dz (Abdelkrim Abdelli), jalel.benothman@l2s.centralesupelec.fr (Jalel Ben Othman)

Preprint submitted to Journal Simulation Modelling Practice and Theory December 8, 2020

models based on Discrete Time Markov Chain are developed to implement several policies of the system. A theoretical modelling of the problem is proposed including analytical results obtained using the Xborne tool. For each model, the average time to reach a given QoS for a community of consumers is reported. Such a performance metric allows the designer to predict the system behaviour in a dynamic environment.

Keywords: `Performance evaluation; Web services; Markov chains; QoS.`

---

1. Introduction

During recent years, the rapid development and the increasing popularity of the Web service technology have promoted the Cloud as an Internet-based service architecture to achieve various IT capabilities and flexible invocations. The traditional Web service architecture has evolved to be more flexible, dynamic, and scalable. Indeed, there are more and more popular cloud applications which are composed by software components and cloud services. Service composition is needed when a complex request cannot be satisfied by a single service. It could be obtained by combining various available services functionalities into a more powerful new service in the cloud [17]. As for web service architectures, the design of a cloud architecture is oriented on three major issues that are service discovery, service selection and service composition [13].

Service discovery is the process of exploring a population of services to locate alternatives that meet the functional requirements requested by the user. To distinguish services with the same functionalities, service selection uses QoS parameters (such as service reliability, security, trust and execution cost, etc.) to refine the discovery and select the best alternatives. Hence, it is clearly obvious that service discovery is a prerequisite phase for the selection process. Generally, selection is needed when it comes to manufacture new composite services with customized functionalities and according to user's QoS requirements [24]. Such a process is called QoS aware service composition [21].

Nowadays, it is often required that service architectures can be dynamic, in the way that they can dynamically assemble complex services to develop distributed, interoperable, and scalable systems. Moreover, in the open and variable cloud environment, the fluctuation and the instability of the envi-

ronment make that the measured QoS be changing. Therefore, managing the dynamic QoS of the cloud services, along with ways to build credible and accurate combination of services become an important research problem. In this context, the performance of a composition of services is a crucial factor for deciding which components must be selected, or to choose whether a given sequence of interactions can provide the requested QoS.

Although managing the QoS of a composite service is not new, it still remains that many additional issues need to be fixed. Indeed, as service based applications are manufactured by composing different distributed software components, the configuration of the latter as well as the QoS can change. Besides as the environment can fluctuate the original QoS of the considered services is thus affected. Therefore, the user requirements can no longer be met. Updating dynamically the composite services by replacing some of their components with new ones having better QoS is one of the promising solution, with however many challenges to overcome, as for instance:

(i) How to build the composite service with respect to user's requirements, and service availability. When no constraints are formulated on service usability and duplicity, the problem is modelled as a Multidimensional Multi-choice Knapsack Problem (MMKP) [36] to design the QoS aware service composition problem for one consumer. Otherwise, the problem becomes even more complex as it is dedicated for multiple consumers.

(ii) As the exact resolution of the QoS aware service composition problem is NP-hard, alternative approaches to reduce the cost in terms of time complexity are investigated to meet the delay constraints and without loosing in the quality of the computed solutions [22, 8, 6, 30].

Most of the literature approaches are dealing with the problem of QoS aware service composition problem for one consumer. For the case of a multiple consumers, the existing works studies different models for composite services for a group of clients [8].

One of these models is to consider different consumers for the same service such that the latter can be assigned only to one composite service (consumer), providing that the requested QoS is met. Each composite service is thus manufactured according to each client constraints. Rather than seeking for the optimality of the composite service, the problem consists then in finding all the similar composite services in terms of functionalities so that the satisfaction of the consumers group is maximised and each client is satisfied with the returned composite service [16].

For this purpose, different approaches to assess and analyse the perfor-

mances of the composed services have been proposed. These approaches consider in general different set of QoS metrics and different models. Besides, they can be applied at run-time or at the design time.

The goal of this study is to evaluate the systems performances in the context of a QoS aware service composition problem when assuming multiple consumers in a dynamic environment. In the proposed model, we consider an architecture where services with same functionalities are continuously provided to different consumers which are competing to acquire them for service composition. Each service is characterized by a grade that determines an aggregation of its QoS parameters. If a consumer acquires a new service with a higher grade, it should release the old one that will be offered to other consumers, and so on. Different policies are implemented and evaluated in the proposed model, by assuming: first, (i) no constraints on the process duration and the required level of the service grade; (ii) the consumer stops acquiring new services when it reaches a satisfactory grade; (iii) or after reaching a time-out, denoting that the duration of the composition process is overtaken.

In order to evaluate the performances of the system, different models as Discrete Time Markov Chains (DTMC) are elaborated. Then, the obtained models are implemented to derive some performance parameters using the Xborne tool [7]. Mainly, this evaluation makes it possible to assess the average time needed for each consumer to reach a given grade of QoS. Such a metric is important to understand how the system behaves when assuming different number of consumers, QoS levels to reach, and time-outs to achieve the composition process.

The remainder of the paper is organized as follows. Section II presents the related works. Section III discusses the architecture functioning to consider and lays down the assumptions for our performance evaluation model. In section IV, we discuss the general DTMC model and proves its lumpability. In section V, we present the different derived models implementing the considered policies. Section VI presents the numerical analysis of the structure of the DTMC. Section VII reports the numerical results of the performance evaluation of the DTMC models. Section VIII concludes this paper and gives the perspectives.

2. Related works

The majority of approaches dedicated to composite service selection can be classified into two major classes: approaches based on negotiation and those based on optimisation.

2.1. Negotiation

These approaches propose to design a negotiation framework to respond to user's requirements by making the QoS flexible and negotiable between the user and the provider through a common agreement, called SLA (Service Level Agreement) [9]. The SLA contains the negotiation terms that specify, for instance, the QoS parameters, the rewards and the penalties [17]. Such an approach offers more flexibility to achieve a compromise between the user and the provider. Patankar *et al* [29] propose an automated approach based on compromise achievement to buy services by using a bilateral protocol to manage the interactions between users and providers. They consider a mechanism of iterative compromise to evaluate the offers of the different protagonists and thus generate counter-offers of common awards based on the considered QoS parameters. Sathya *et al* [20] proposed a negotiation model with equal reward to select the best service that responds to client requirements. In [23], the authors deal with the selection of composite services on the base of Multi-Agents negotiation. The goal of these agents is to determine the best Composite QoS. They proposed an architecture that achieves to reduce the selection time.

The major drawback of negotiation base approaches is the limited number of QoS criteria to be considered in the SLA (no more than two), because of the complexity of the solution [17].

2.2. Optimisation

In the context of service selection optimization, the problem is described generally as a Multi-dimensional Multichoice Knapsack Problem (MMKP) to determine the service composition with the best QoS that satisfies user requirements [22, 35, 10].

MMKP is well known to be NP-Hard, and many approaches have been proposed in the literature. In [25], the authors survey the existing approaches and classified them considering five criteria: the class of methods using linear or non linear objective functions; with local or global optimisation; with or without QoS requirements; single or multi-objective optimisation; exact

solutions or heuristics. The taxonomy has been extended in [24] to consider "dynamic or static composition" as an additional criterion:

- A linear resp non linear program seeks to maximise or minimize the linear resp. non linear objective function with respect to user requirements.
- Local optimisation tends to select the best service for each individual task by considering the QoS requirements of each task. The global optimisation aims at selecting the best composition taking into account the global QoS requirement.
- If the selection process can deal with user QoS requirements, it considers the best service composition among those available. Otherwise, it considers the optimal composition according to user requirements.
- A single objective optimisation aggregates the QoS parameters values by using an utility function, and compares the results to determine the best composition. On the other hand, a multi-objective optimisation associates to each QoS parameter an utility function and then compares the services for each parameter to select the optimal compositions [11].
- An exact solution selects the optimal composition without taking into account the cost needed to process it. An heuristic solution selects a composition by reducing the research space, while maintaining the processing time under a given threshold [2, 35].
- In a dynamic environment, the availability and the accessibility to the providers are not guaranteed. The latter can also improve the QoS of the services they offer, to keep competitive. In such environment, the composition process is updated when changes are noticed in the QoS. In a static environment, the QoS information is assumed fix, and the composition is performed according to that, without any update [24].

In QoS aware service composition, a set of QoS attributes is considered to manufacture the best composite service. However, few works address the problem of QoS aware service composition with multiple consumers.

In [3], Bentallah *et al* grouped services with similar functionalities and different non-functional properties to ease the composition process. In [5], the authors proposed to use communities formed by similar service operations

to promote the service composition and also to substitute operations in the composite service. Wang *et al* proposed in [32] to create a community formed by agents (service providers) having the same area of interest, and each agent gives his opinion and judgement criteria regarding a service. A super agent plays an important role in managing the community by choosing its members and maintaining its reputation. The goal is to promote the trustworthiness of services as a key factor to select services and to join the communities.

In [18], the authors consider the Social Spider Algorithm (SSA) to solve the problem. The experiments evaluate the efficiency and the feasibility of the proposed algorithm against Particle Swarm Optimization (PSO). SSA is found to outperform PSO in terms of both execution time and fitness. In [14], the authors propose an approach for QoS aware service selection for multiple choreographies considering the sharing of services among them. They consider the aggregation service load that results from the sharing to compute the QoS. In [33], the authors propose a service recommendation approach that improves QoS aware efficiency service selection for multi-tenant SaaS (Software as a Service). They consider representative candidate for communities of services based on the diversity and the similarity in tenants' QoS requirements. In [34], the authors propose an on-demand strategy for QoS aware service composition by introducing a service broker whose role is to purchase a number of service instances for each component from providers and then to provision consumers with composite services with different QoS classes. The efficiency of the solution is evaluated by heuristic approaches. In [1], the authors introduce two types of services, called: leaders and followers. Leaders are those services that enjoy high reputation, market share, and capacity of handling requests; whereas followers are those services that cannot compete against the leaders. The problem is modelled as a virtual trading market and propose a distributed Stackelberg game for this purpose to achieve higher performance, efficient services compositions, and better resources utilization. They show that their model is capable to improve the user' satisfaction regarding the QoS requirements. The authors in [31] propose a QoS-aware service selection model based on fuzzy linear programming. The proposed approach provides the optimal solution of consensual weight of QoS attribute and fuzzy positive ideal solution (FPIS) by extending LINMAP method. They report experimental results that advocate the usability of the approach.

In [16], we proposed to evaluate the performances of a service composition architecture in a dynamic environment. For this purpose, a model based on

a DTMC (Discret Time Markov Chain) is proposed to specify the basic functioning of the architecture in order to evaluate the performances of the system. For each scenario, the mean processing time to compute a solution with a given QoS is calculated. To the best of our knowledge, there exist no further papers in the literature that have addressed the use of the DTMC to evaluate the performances of QoS aware Web service composition with multiple consumers. The subject of our study in the present paper, is to extend the work in [16] by considering different policies and scenarios. This requires to introduce new theories in order to specify the DTMC models and to evaluate their performances.

3. Problem description

In this section, the concept of community of composed services is introduced and a discussion is given on the problem of the optimal solutions in a dynamic environment selection. The proposed architecture is described and discussed according to different policies.

3.1. Community of services

Complex tasks need the execution of different simple tasks, each one could refer to a specific service offered by one or different providers. Services offering the same functionalities can be grouped within the same class and distinguished by their QoS performances. For the sake of simplicity, we consider in this study only one QoS parameter associated with each service that denotes an aggregation of all its performances. Although services are offering a wide flexibility, several classes of services are combined to offer composite services. It is assumed for this study that service composition requires exactly M classes Sc_i.

A *community* is called as a composition of M services as a response to a user request, such that each service s_i is taken from the class Sc_i. The concept of a *community* is introduced to build a dynamic composite service, quality of which can be assessed by the aggregation of the QoS of the services that compose it. As services composing a community vary over time, the community is thus dynamic and each affectation is thus called *configuration*.

A service is given to a community only if it maximises the global award. However, each community aims at maximising its own global QoS. To introduce more rationality it is assumed that the communities avoid, if possible, unfavourable configurations. That is, communities have the right to exchange

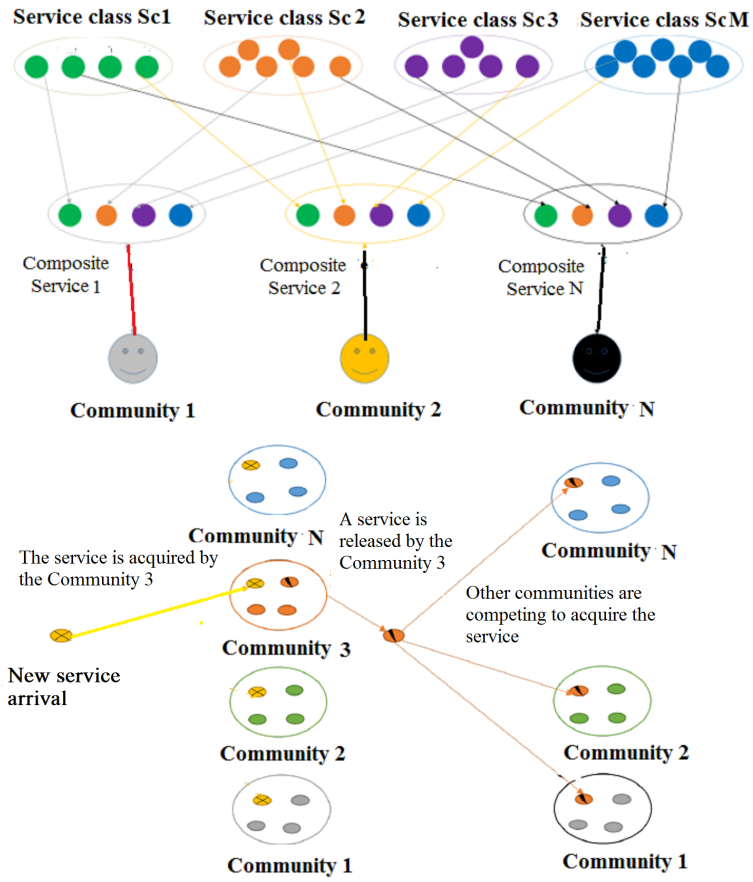

Figure 1: **Composite service selection with multiple consumers**

their services to improve their QoS over time. From a starting configuration a community which wants to update some of its services cannot take those already assigned to other communities. Therefore, only new services that are not used or those released can be acquired.

Determining the optimal configuration for a community amounts to solve a variant of the knapsack problem, well known to be NP-Hard [22]. Although approximate solutions can be obtained by using heuristics to solve such a problem, they may take a very long time to be computed. Moreover, services are unstable as they can be removed by their providers without notice, thus making the computed solutions obsolete. More generally, adding or deleting a service can affect the existing communities. A basic and naive solution is to recompute the optimal configurations but this seems not practical, because of its high cost in terms of computation time [15].

Building dynamic communities makes it possible to provide a more flexible and reliable solution by updating the configurations on the fly using the global award as a key parameter to guide the process. In case a community wants to swap one of its services with a new one having better QoS performances, then the released service may be acquired by other communities providing that its QoS meets their requirements.

In order to limit the number of swaps and to maximise the global award, it is considered the following assumptions:

- For each community, a service can hold only if it increases the QoS of the configuration.
- If different communities are in competition to acquire a new service, the latter should be given to the community such that the global award is maximized. This is the case when the communities are created for the first time. There are many ways to proceed, one of them is to consider a priority according to the QoS requirements. Two strategies can be enforced:
    1. The community requesting the highest QoS requirement for the arriving service has priority: Such a strategy is more realistic as it seeks to guarantee the optimal solution for each community. However, it takes much longer time to reach the optimal solution.
    2. The community requesting the lower requirements has priority: Such a strategy allows most of the communities to satisfy their

requirements faster than the previous strategy, but does not guarantee the optimality of the composition.

To sum up, the problem can be summarized by the following points:

- A group of N clients, each one looks for a composite service, denoting the community.
- Each composite service is composed of M services issued respectively from M classes of services.
- Each class of services provides one service for each composite service .
- Each client expresses requirements for the composite service to look for.
- One service can be consumed by maximum one community.

Solving such a problem amounts to maximize the distribution of services on the N composite services (communities), so that the QoS requested by each clients be met. Figure 1 describes the problem.

In this study, it is considered the following policies to decide when to stop the composition process:

1. The first policy considers that a given number of consumers are seeking for similar services with the same functionalities, but with the highest grade. At this stage, no constraint on the time is enforced to complete the task, nor on the satisfactory grade for each community.
2. In the second policy, we assume in our model further that a consumer has to stop acquiring new service once he achieves a satisfactory predefined grade.
3. In the third policy, we assume that the the consumer must stop acquiring services if the processing time overtakes a given threshold.
4. By combining the policies 2 and 3.

4. Lumpability of the general DTMC model

In this section, a description of the model is given based on discrete-time Markov chain, and we propose a partition of the state space which provides an exact aggregation of the Markov chain. This proof is based on the strong

lumpability property (see [12] for the initial definition of aggregation for finite chains, and [4] for more detailed results). Let X be a Markov chain on set of states $\mathbb{X}$. Let $(B_1, .., B_k)$ be a partition of $\mathbb{X}$. We define a new process Y as follows:

$$Y_n = m \iff W_n \in B_m.$$

The question is to fo find conditions such that Y is also a Markov chain. Y will be denoted as an exact aggregation of W for partition $(B_1, .., B_k)$. The ordinary lumpability condition (defined in the following) implies such a result.

Definition 1 (Ordinary Lumpability, [4]). *W is ordinary lumpable for partition $(B_1, .., B_k)$ of its state space if for all subset index i and j and for all state $m1$ and $m2$ in B_i, we have*

$$Pr(W_{n+1} \in B_j | W_n = m1) = Pr(W_{n+1} \in B_j | W_n = m2).$$

B_i is denoted as macro state i.

Two other properties will be used here (see [4] for more details).

Definition 2 (exact Lumpability). *W is exactly lumpable for partition $(B_1, .., B_k)$ of its state space if for all subset index i and j and for all state $m1$ and $m2$ in B_i, we have*

$$\sum_{i \in B_j} Pr(W_{n+1} = m1 | W_n = i) = \sum_{i \in B_j} Pr(W_{n+1} = m2 | W_n = i).$$

B_i is denoted as macro state i.

and finally:

Definition 3 (strict Lumpability). *W is strictly lumpable for partition $(B_1, .., B_k)$ if it is both ordinary lumpable and exactly lumpable for this partition.*

This is very important because our software has, like all numerical tools, a limit on the size it can handle. Exact aggregation and ordinary lumpability allows to build a smaller model despite larger parameters and this model is still Markovian. Exact lumpability implies that all the states which are aggregated by the partition all have the same steady-state definition. This is

also an help to we deal with models having large state space. For instance, with parameters $N = 12$ and $G = 2$, Model1 without lumping has 531522 states while the lumped version of this model only has 91 states. Clearly, we have a substantial reduction of the states space. We will give more details in the following when we compute the size of the models.

4.1. Model description

In the proposed model, N communities of services are considered. Each service has a grade g which represents an aggregate value of its QoS. The higher the value the grade has, the more the communities want to acquire it. G is denoted as the highest value of the service grade, such that $0 \leq g \leq G$.

An event e_g consists in the arrival of a service with grade g. It has a probability $p(e_g)$ which does not depend on the state X. The considered model can be be represented by a discrete-time Markov chain (DTMC). A given state of the Markov chain can be described by N components $(x_1, x_2, x_3, \ldots, x_N)$, where x_i represents the current grade of the service s in the community i with $(1 \leq i \leq N)$ and $(0 \leq x_i \leq G)$. Thus, the number of states is $(G+1)^N$.

The application of the event on state X produces a subset $e_g(X)$ of states. This subset may be a singleton (for instance when the service is rejected because its grade is too low). For each state in $e_g(X)$, $Pr(X, Z, e_g)$ will denote the transition probability from X and Z due to the occurrence of the event e_g. By construction we have:

$$Pr(X, Z) = \sum_g Pr(X, Z, e_g).$$

Assume that a service with grade g arrives. Let X be the state of the chain just before the arrival. The rules to affect the service are the following:

- Find the subset (say Sb) of the components of X which have the highest grade smaller than g.
- If this subset is empty, the service is rejected.
- Otherwise, select at random with an uniform distribution a component in Sb and replace the service of this component by the incoming service. Affect again the service which has been released by the component to X with the same set of rules.

Example 1. *For instance, consider state* $(1, 3, 0, 1)$ *and assume an arrival of a service with grade* 2*. First, components* 1 *and* 4 *are selected. This is due to their grade (i.e.* 1*) which is the highest one smaller than* 2*. As two components are selected, one of them is chosen at random with equal probability. Then, a service with grade* 1 *is released by component* 1 *or* 4*. And it is given to component* 3 *because its grade is smaller (as it is the only component with grade* 0*, we do not need a random choice here). Finally, we get:*

$$e_2(1, 3, 0, 1) = \{(2, 3, 1, 1), (1, 3, 1, 2)\}$$

Property 1. *As, at each step, the choice is made at random with equal probability between all the components which have the highest grade smaller than* g*, all the states in subset* $e_g(X)$ *have the same probability to be reached from* X*. Therefore for all the states* Z *in* $e_g(X)$ *we have:*

$$Pr(X, Z, e_g) = \frac{p(e_g)}{|e_g(X)|},$$

where $|e_g(X)|$ *is the cardinality of* $e_g(X)$*.*

Property 2. *Due to the assignment rules, the result of an arrival of a service with grade* g *is defined as:*

- *Service assignment: If a community has previously a service with a grade strictly smaller than* g*. The service with the weakest grade is released. Therefore* $X \notin e_g(X)$*.*
- *service rejection: If all the communities had previously acquired a service with a higher grade. Thus,* $e_g(X) = \{X\}$*.*

Proof: At each step we exchange one of the components which have the highest grade smaller than g (say h) with g. And at next step we make the assignment of a service with grade h. Thus, the results only differ by the components receiving the service, not by the grade of the service.

4.2. Lumpability

We define the mapping s on state space $\mathcal{S}$ as follows: $s(X)$ is a sorted word (in decreasing order) with the same letters as in X. Clearly, s defines an equivalence relation between the states.

Assumption 1. *Let's now consider the partition based on s: X_1 and X_2 are in the same subset (or macro-state) iff $s(X1) = s(X2)$.*

We also apply s to the subset of states. By definition, in the image of a subset, the duplicated states are removed. Let us begin with an example.

Example 2. *We consider a simple example with $N = 3$ and $G = 1$. The Markov chain has 8 states: $(0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,0), (1,0,1), (0,1,1), (1,1,1)$. The transition matrix is described in Eq. 1 and is given in the following:*

$$\left(\begin{array}{c|ccc|cccc} p_0 & p_1/3 & p_1/3 & p_1/3 & 0 & 0 & 0 & 0 \\ \hline 0 & p_0 & 0 & 0 & p_1/2 & p1/2 & 0 & 0 \\ 0 & 0 & p_0 & 0 & p_1/2 & 0 & p_1/2 & 0 \\ 0 & 0 & 0 & p_0 & 0 & p_1/2 & p_1/2 & 0 \\ \hline 0 & 0 & 0 & 0 & p_0 & 0 & 0 & p_1 \\ 0 & 0 & 0 & 0 & 0 & p_0 & 0 & p_1 \\ 0 & 0 & 0 & 0 & 0 & 0 & p_0 & p_1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & p_0 + p_1 \end{array}\right) \tag{1}$$

Macro-state	States
(0,0,0)	(0,0,0)
(1,0,0)	(1,0,0), (0,1,0), (0,0,1)
(1,1,0)	(1,1,0), (1,0,1), (0,1,1)
(1,1,1)	(1,1,1)

Table 1: Table of Macro-states.

The block decomposition, given in the previous transition matrix by Eq. 1, is based on the partition required for the lumpability. Clearly the model is ordinary lumpable as the row sum is a constant per block. Similarly, the model is exactly lumpable as the column sum is a constant per block. The lumped chain has 4 states which are described in Table 1, and the lumped matrix is:

$$\begin{pmatrix} p_0 & p_1 & 0 & 0 \\ 0 & p_0 & p_1 & 0 \\ 0 & 0 & p_0 & p_1 \\ 0 & 0 & 0 & p_0 + p_1 \end{pmatrix}$$

Note that even for this small example, we have a state space reduction.

Let us start by laying down some technical lemmas.

Property 3. *For all the events e_g, and for all the states X, $s(e_g(X))$ is a singleton.*

Proof: If $e_g(X)$ is already a singleton, the conclusion clearly holds. If the cardinal of $e_g(X)$ is larger than 2, we consider two distinct states Y_1 and Y_2 in $e_g(X)$. Due to Prop. 2, Y_1 and Y_2 only differ by the components that update their service grade. Therefore, we obtain $s(X_1) = s(X_2)$ and the proof is complete.

Let mY be a macro-state and X a state. We denote by $Pr(X, mY)$ the summation $\sum_{Y_1 \in mY} Pr(X, Y_1)$. Let R_{X_1} be the set of events e_h which are rejected at state X_1. Similarly, let A_{X_1} be the set of events e_h which are accepted at state X_1.

Property 4. *For all the events e_g, states X_1 and X_2, such that $s(X_1) = s(X_2)$, we have:*

$$s(e_g(X_1)) = s(e_g(X_2)).$$

Furthermore, we also have: $R_{X_1} = R_{X_2}$. The rejected arrivals are the same for states X_1 and X_2. $A_{X_1} = A_{X_2}$ also holds as A_{X_1} is the complementary subset of R_{X_1}. Finally, we have:

$$\sum_{Y_1 \in s(e_g(X_1))} Pr(X_1, Y_1) = \sum_{Y_2 \in s(e_g(X_2))} Pr(X_2, Y_2)$$

Proof: The first property is a trivial consequence of the definition of the events and Prop. 3. Now assume that an event e_g is in R_{X_1}, then $e_g(X_1) = X_1$. Thus, $s(e_g(X_1)) = s(X_1)$. By construction $s(X_1) = s(X_2)$ and due to the first relation $s(e_g(X_1)) = s(e_g(X_2))$. Therefore by transitivity, $s(e_g(X_2)) = s(X_2)$ and e_g is also in R_{X_2}.

Let's turn to the final relation. We have two cases:

- The event e_g is rejected
- The arrival of a service with grade g triggers some changes following the selection rules.

In the first case, $e_g(X_1) = X_1$. For all h in R_{X_1}, we have $e_h(X_1) = X_1 = e_g(X_1)$. Therefore $s(e_h(X_1)) = s(X_1) = s(e_g(X_1))$. Thus,

$$\sum_{Y_1 \in s(e_g(X_1))} Pr(X_1, Y_1) = \sum_{h \in R_{X_1}} p(e_h)$$

Similarly, we get:

$$\sum_{Y_2 \in s(e_g(X_2))} Pr(X_2, Y_2) = \sum_{h \in R_{X_2}} p(e_h)$$

As $R_{X_1} = R_{X_2}$ we finally get:

$$\sum_{Y_1 \in s(e_g(X_1))} Pr(X_1, Y_1) = \sum_{Y_2 \in s(e_g(X_2))} Pr(X_2, Y_2)$$

In the second case (i.e. g is accepted), we first remark that there are no grade h (distinct from g) such that $s(e_g(X_1)) = (s(e_h(X_1))$. Indeed:

- If h is rejected ($s(e_h(X_1)) = X_1$ and $s(e_g(X_1)) \neq X_1$. Thus, $e_h(X_1) \neq e_g(X_1)$
- If h is accepted, assume without loss of generality that $h \neq g$, thus $s(e_h(X_1)) > s(e_g(X_1)$.

Now, by construction, $\sum_{Y_1 \in s(e_g(X_1))} Pr(X_1, Y_1) = p(e_g)$. The summation does not depend on X. Therefore we have:

$$\sum_{Y_1 \in s(e_g(X_1))} Pr(X_1, Y_1) = \sum_{Y_2 \in s(e_g(X_2))} Pr(X_2, Y_2) = p(e_g),$$

or:

$$Pr(X_1, s(e_g(X_1))) = Pr(X_2, s(e_g(X_2))) = p(e_g).$$

Lemma 1. *The model is ordinary lumpable.*

Proof: With the notation of our problem, we have to prove that for two arbitrary states X_1 and X_2 such that $s(X_1) = s(X_2)$ (i.e. there are in the same macro-state), then for all the macro-states mY, we have:

$$Pr(X_1, mY) = Pr(X_2, mY)$$

We have two cases:

- Assume that $mY = s(e_g(X_1))$ for some event e_g. Then the proposition 4 is applied.
- Otherwise, there is no transition from state X_1 to macro-state mY and $Pr(X_1, mY) = 0$. Similarly, $Pr(X_2, mY) = 0$ and the equality holds.

Lemma 2. *The model is exactly lumpable.*

Proof: from the previous properties, we have three cases for the number of events associated with the transitions between two macro-states (say mX and mY):

- 0. No event may transform a state of mX in a state of mY. For instance, if macro-states mX and mY differ in more than 3 components.
- exactly 1 for a service with grade g which is accepted.
- 1 or more for rejected events. In that case, there is no transition. These events are associated to the diagonal elements of the transition matrix.

In the first and the third case, the criteria for exact lumpablity is clearly satisfied (the block matrices are respectively, null matrices and identity matrices). Let us turn to the second case. It is considered two macro-states mX and mY such that there exists only one event g such that $e_g(mX) = mY$. From Prop. 1 we know that each state of mX has de same output degree, and that all the transitions have the same probability. Clearly all the states of mY have the same number of predecessors in mX. Therefore the row column in a block is constant and the chain is exactly lumpable.

Theorem 1. *The model is strictly lumpable.*

Proof: because it is both ordinary lumpable and exactly lumpable.

Property 5. *After lumping, the number of states is* $\binom{N+G}{G}$.

Proof: We want to count the number of non decreasing sequence of length N based on a set of $G+1$ symbols. Consider an arbitrary sequence of length N. For $i = 0$ to G, let x_i be the number of symbol i in that sequence. Clearly, we have for all sequence by construction:

$$x_0 + x_1 + ... + x_G = N \tag{2}$$

There is one to one mapping between any vector $(x_0, x_1, ..., x_G)$ solution of Eq. 2 and a non decreasing sequence of length N as illustrated in Fig. 3

$$x_0 + x_1 + ... + x_G \iff (\underbrace{0..0}_{x_0}, \underbrace{1..1}_{x_1}, ..., \underbrace{G...G}_{x_G}) \tag{3}$$

Indeed, the non decreasing sequence must begin with x_0 letter 0, followed by x_1 letter 1 and so on, until x_G letter G. Thus, the number of increasing sequences is equal to the number of ways to distribute N balls into $G+1$ urns, a classical problem in combinatorics. It is already known that the number of solutions of Eq. 2 is $\binom{N+G}{G}$ (see for instance [28], p 571, it is also the number of states in a Markov chain modeling a closed network with N customers and $G+1$ stations).

Thus, we can design a lumped model with a much smaller number of states. Remember that the size of the initial Markov chain is $(G+1)^N$. We also develop new techniques to deal with larger values of parameters N and G in the next sections.

In the sequel, based on the model description and given proofs, the developed DTMC are assumed lumped models.

5. Models based on different policies

In this section, we redesign the DTMC model according to different policies. The first one refers to the policy wherein communities are always seeking to acquire services with the highest grade. In this case, the model contains only one absorbing state. The second one refers to the strategy where communities stop seeking for a new service once they acquire a service satisfying a given grade. The Markov chain contains several absorbing states. The last one refers to the policy using a time-out. In this case, the communities have the possibility of setting a threshold of grade values as a stopping criterion.

5.1. Model based on the highest grade

In this first model, it is considered that all the communities will end up having the service with the highest grade regardless of the time. The Markov chain size depends on the number of communities and the value of G. Thus, more communities and grades.

Property 6. *The number of states is* $\binom{N+G}{G}$.

The proof is given in Eq. 2 in section 4.

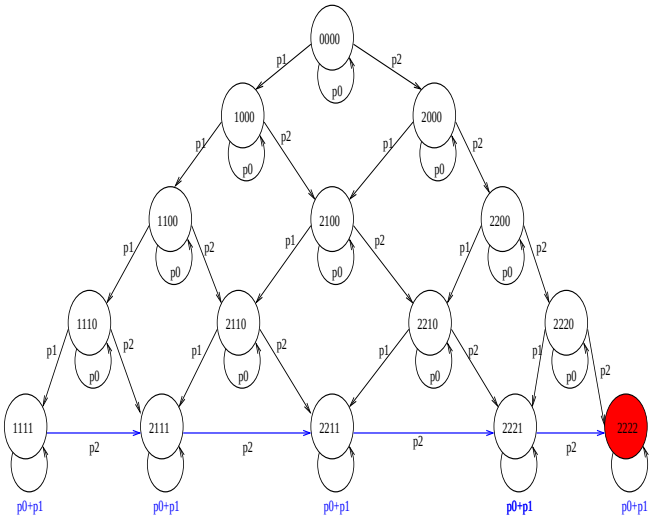

Figure 2: **The obtained Markov chain with** $N = 4$ **and** $G = 2$

Example. To explain clearly the proposed model, we consider an example with $N = 4$ and the score G is between 0 and 2. In this case, Figure 2 depicts the transitions of the Markov chain. The Markov chain size is $\binom{4+2}{2} = 15$ states.

(s_g) is the arrival of a service s with its grade g. Let p_g be the probability that the incoming service (s_g) arrives with a grade g. The states X of the DTMC is a state with four components (x_1, x_2, x_3, x_4) with $(0 \leq x_i \leq G)$, where:

- State $(0, 0, 0, 0)$ means that the service has a score 0 for all the communities.
- State (x_1, x_2, x_3, x_4) means that the service has a grade x_1 for the community 1, grade x_2 for the community 2, grade x_3 for the community 3 and grade x_4 for the community 4. According to our rules $(x_1 \geq x_2 \geq x_3 \geq x_4)$

Thus, due to the linearity of the expectation and the memoryless of the arrival process, the average time to reach this state is N times the average time to receive a service with grade G. The arrivals are geometric with rate p_G. Therefore the expected time to reach $(G, G, ..., G)$ is $\frac{N}{p_G}$.

More details are given in [16].

5.2. Model based on satisfactory grade

This case implements the second policy: communities stop seeking for a new service once they acquire a service satisfying a given grade. The Markov chain contains several absorbing states. The whole Markov chain is first built. Then the steady state distribution is computed in order to derive the rewards. For instance, it could be interesting to compute the average time to reach the states with a given grade value equal to v. v is considered as a satisfactory value for the service grade.

Based on example of Fig 2, we can determine when all the communities are able to acquire the services with grade $g \geq v$ with $v = 1$, and consequently when all the new services are being rejected automatically. Thus, the states $(1,1,1,1), (2,1,1,1)$ $(2,2,1,1), (2,2,2,1), (2,2,2,2)$ are the absorbing states when considering the second policy. In our study, we are interested in computing the average time to reach these states starting from the initial state $(0,0,0,0)$ or any other state. More details are given in [16].

5.3. Model based on time-out

In section 5.1, we have considered as a stopping criterion the case wherein all the communities obtain the service with the highest grade G. After, we relaxed this constraint and gave the possibility of setting a threshold of grade values as a stopping criterion in section 5.2. In this section, we introduce the time-out as another stopping criterion. Thus, the model associated with the previous policies is extended to design the clock parameter. To clearly explain the model, three communities are considered and a new parameter h is introduced, as well to design the clock.
A state X of the DTMC is characterized with four components (x_1, x_2, x_3, h) with $(0 \leq x_i \leq G)$ and $(0 \leq h \leq H)$ where:

- H is the value of the time-out.
- State $(0,0,0,0)$ means that the service has a score 0 for all communities at time 0.
- State (x_1, x_2, x_3, h) means that the service has a grade x_1 for community 1, grade x_2 for community 2 and grade x_3 for community 3 at time h. According to our rules $(x_1 \geq x_2 \geq x_3, h)$.

The different transition probabilities between states are given, next:

1. if the incoming service s_g arrives with a grade g, such that $(x_1 \geq x_2 \geq x_3, h)$:

$$(x_1, x_2, x_3, h) \rightarrow (1) \ (x_1, x_2, x_3, h+1)$$

   (1) means that the grade g of the new service s_g is less or equal than the current score for all the communities. Thus, there is no loop for the states in the Markov chain as the time h is incremented by 1. The transition probability is p_g.
2. if with probability p_g, the incoming service s_g arrives with a grade g, such that at least one $x_i < g$:

$$\begin{aligned}(x_1, x_2, x_3, h) &\rightarrow (2) \ (g, x_2 = x_1, x_3 = x_2, h+1) \\ &\rightarrow (3) \ (x_1, x_2 = g, x_3 = x_2, h+1) \\ &\rightarrow (4) \ (x_1, x_2, x_3 = g, h+1)\end{aligned}$$

   - (2) means that the new service has a score g which is greater than the current notation x_1. Thus, the community 1 takes the new service $(x_1 = g)$ and releases its current one which is proposed to the community 2. The latter is going to apply the same rules as the previous one and so on. The clock is incremented. The transition probability is equal to p_g.
   - (3) means that the new service has a grade g which is upper than the current notation x_2 and is less or equal to x_1. The community 1 is not interested by the new service. Thus, the community 2 takes the new service $(x_2 = g)$ and releases its current one which is proposed to the community 3. This community is going to apply the same rules as previously and so on. The clock is obviously incremented. The transition probability is equal to p_g for $0 \leq g \leq G$.
   - (4) means that the new service arrives with a grade g which is upper than the current notation x_3 and which is less or equal to x_1 and x_2. Then same rules as in (2) and (3) are applied. The clock is obviously incremented.
3. The final states are depending on the stopping criterion that has been chosen.

- If it is assumed that the stopping criterion is when all the communities obtain the service with the highest grade G, the absorbing states are all the states x of the form (G, G, G, h) with $G \leq h \leq H$. For $N = 3$, $G = 2$ and $H = 5$, the absorbing states are: $(2, 2, 2, 3), (2, 2, 2, 4), (2, 2, 2, 5)$
- If the stopping criterion is when the time-out is reached, the absorbing states are all the states $(x_1, x_2, x_3, h = H)$, such that $0 \leq x_1 \leq 2$, $0 \leq x_2 \leq 2$ and $0 \leq x_3 \leq 2$. For $N = 3$, $G = 2$ and $H = 3$, the absorbing states, as shown in Figure 3, are:

$$(0, 0, 0, 3), (1, 0, 0, 3), (2, 0, 0, 3), (1, 1, 0, 3), (2, 1, 0, 3),$$

$$(2, 2, 0, 3), (1, 1, 1, 3), (2, 1, 1, 3), (2, 2, 1, 3), (2, 2, 2, 3)$$

The graphic representation of the Markov chain is depicted in Figure 3.

6. Efficient numerical analysis due to the structure of the DTMC

In this section, we will exploit the interesting structure of the Markov chain. We will show that in this case, the computation of the stationary distribution will be done with less complexity. All the proofs are given in this section.

We consider the following partition of the states space: $\{\mathcal{I}, \mathcal{M}, \mathcal{F}\}$. $\mathcal{I}$ is the singleton containing the initial state. $\mathcal{F}$ contains the final states while all the other states are in $\mathcal{M}$. Taking into account the definition of the states and the transition, the chain has the following structure:

$$M = \left[\begin{array}{c|c|c} 0 & R1 & R2 \\ \hline 0 & U & C \\ \hline 0 & 0 & Id \end{array}\right].$$

This is the canonical structure of a transient DTMC in which it has been taken into account that the initial state (i.e. state $(0, 0, ...0)$) has the input

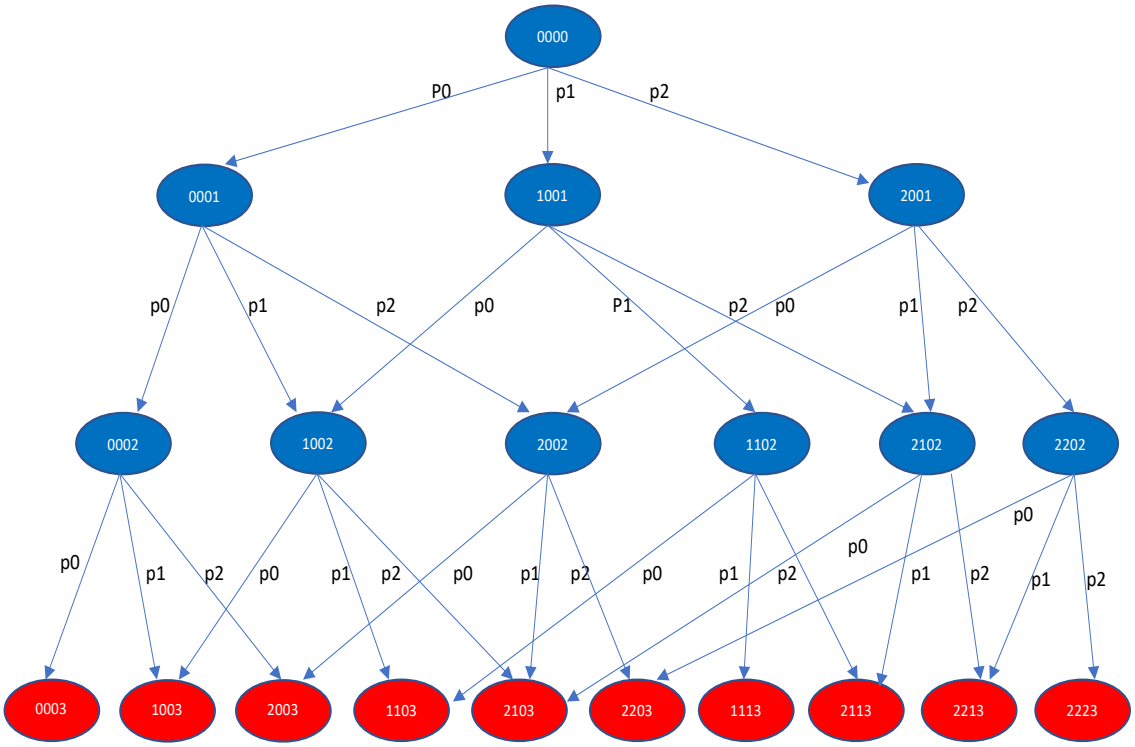

Figure 3: **Markov chain** $N = 3$**,** $G = 2$ **and** $H = 3$

degree 0. The method consists in computing the fundamental matrix F which is then used to obtain the average transient time for all the initial states and the absorption probabilities knowing the initial state:

$$F = (Id - \left[\begin{array}{c|c} 0 & R1 \\ \hline 0 & U \end{array}\right]_{\cdot})^{-1}$$

Note that the procedure to compute F already exists for an arbitrary matrix in XBorne. Remember that for the classical approach, we first have to compute F. The following property is well known:

Property 7. *$F[i,j]$ is the average time spent in j before absorption when the initial state is i. The average time before absorption when the first state is i is equal to $\sum_j F[i,j]$. Furthermore $(F * \left[\frac{R2}{C}\right])[i,j]$ contains the probability of being absorbed in state j when the initial state is i.*

Computing F for an arbitrary transient matrix Q has a cubic complexity. Unfortunately the complexity is still cubic when Q is upper diagonal [26]. It is also known to be numerically unstable when the some *transition probabilities* are very small. Therefore it makes sense to find a faster or more accurate algorithm.

Property 8. *The DTMC graph is such that all the directed cycles of length larger than* 1 *go through state* $(0,...0)$.

Proof: Indeed, the DTMC graph consists in a tree rooted in state $(0,...0)$ with some directed edges from the states labelled "*final states*" back to state $(0,...0)$. Note that all the states wherein all the communities have received a service enjoy a self loop. Let us consider an arbitrary event. It consists in the arrival of a service with a grade g. If g is smaller than all the grades already provided to the communities, then it is rejected and the transition is a self loop. When a new service with a grade g is accepted, the global mark of the state increases. As it is not possible to make that mark decreases, the graph does not contain directed cycles of length larger than 1. Finally when the state reaches one of the "final states", a transition takes place to return to the initial state.

Property 9. *For such a graph, Robertazzi proposed a very efficient algorithm to compute steady state distribution π of the DTMC [19]. We assume that state 1 is the root of the tree and that the nodes are ordered in the topological ordering. Thus, if $i < j$ there is no transition from j to i.*

1. *Initialize $\pi[1] = 1$*
2. *Use the topological ordering of the state in the tree to compute the probability. Due to the previous remark, the balance equation at node i is*
$$\pi[i] = \sum_{j<i} M_[j, i]\pi_j$$
And at step i, $\pi[1]$ to $\pi[i-1]$ have been solved. Therefore $\pi[i]$ can be obtained easily and it requires a number of operations equal to the number of non zero entries in column i.
3. *Once all node values have been computed, re-normalize the probability. Compute $S = \sum_{i=1}^{N} \pi[i]$ and perform $\pi = \pi/S$ for normalisation.*

Such an algorithm only requires $O(m)$ operations where m is the number of non zero entries in the matrix of the DTMC. In the following, we show how this structure can be used to solve the two questions related to the expectation of the first return time and probability to reach the "final states".

6.1. Expectation of the return time

Let $\Delta(i)$ be the average time needed to reach an absorbing state of M when starting at state i. It is obviously that $\Delta(i) = 0$ if i is an absorbing state. Furthermore, due to the ordering of the states in the DMTC, we have:

$$\Delta(i) = 1 + \sum_{j \geq i} M[i, j]\Delta(j)$$

Or

$$\Delta(i) = \frac{1 + \sum_{j>i} M[i, j]\Delta(j)}{1 - M[i, i]}$$

Thus all expectations times can be computed from the DTMC when it leaves back to the root (i.e. state $(0, 0, ..., 0)$). And the needed result is $\Delta(1)$. Such an algorithm only needs $O(m)$.

6.2. Probability of the "final states" (i.e. states in $\mathcal{F}$)

We slightly modify the DTMC to use a numerical algorithm dedicated to recurrent Markov chains. We consider the matrix given by the following block decomposition.

$$P = \left[\begin{array}{c|cc} 0 & R1 & R2 \\ \hline 0 & U & C \\ \hline 1 & & \\ 1 & 0 & 0 \\ 1 & & \end{array}\right].$$

Property 10. *Let π_P be the steady state distribution of P (we assume it exists). And let $\pi_M[i]$ be the probability to be absorbed by the state i beginning in the first state in the DTMC associated with M. We have for all the states i in $\mathcal{F}$:*

$$\pi_M[i] = \frac{\pi_P[i]}{\sum_{j\in\mathcal{F}} \pi_P[j]}.$$

Thus we can use the Robertazzi algorithm to compute π_P and obtain π_M, as a conditional probability.

Proof: First we write the balance equation in state 1 for the chain with the matrix P taking into account its structure.

$$\pi_P[1] = \sum_{j\in\mathcal{F}} \pi_P[j] \tag{4}$$

Let us denote by $\pi_P[\mathcal{M}]$ (resp. $\pi_P[\mathcal{F}]$) the subvector of π_P for state in $\mathcal{M}$ (resp. in $\mathcal{F}$). We also write the balance equation in the block form for $\mathcal{M}$ and $\mathcal{F}$.

$$\pi_P[\mathcal{M}] = \pi_P[\mathcal{M}]U + \pi_P[1]R1,$$

and

$$\pi_P[\mathcal{F}] = \pi_P[\mathcal{M}]C + \pi_P[1]R2,$$

As the matrix U is transient, $(Id - U)$ is not singular. Therefore we get:

$$\pi_P[\mathcal{M}] = \pi_P[1]R1(Id - U)^{-1} \tag{5}$$

$$\pi_P[\mathcal{F}] = \pi_P[1]R1(Id - U)^{-1}C + \pi_P[1]R2 \tag{6}$$

Let us now consider matrix M. It is well known that the reverse (if it exists), of the block matrix $\left[\begin{array}{c|c} X & Y \\ \hline Z & W \end{array}\right]$ is:

$$\left[\begin{array}{c|c} (X - YW^{-1}Z)^{-1} & -X^{-1}Y(W - ZX^{-1}Y)^{-1} \\ \hline -W^{-1}Z(X - YW^{-1}Z)^{-1} & (W - ZX^{-1}Y)^{-1} \end{array}\right]$$

Thus, the block decomposition of F becomes: $\left[\begin{array}{c|c} 1 & -R1(Id - U)^{-1} \\ \hline 0 & (Id - U)^{-1} \end{array}\right]$.

After multiplication by $\left[\begin{array}{c} R2 \\ \hline C \end{array}\right]$, we get that the probability to be absorbed in j when the initial state is 1 is:

$$\pi_M[j] = (R2 + R1(Id - U)^{-1}C)[j]. \tag{7}$$

By combining the Equations 4, 5, 6 and 7, we finally get to conclude the proof:

$$\pi_P[\mathcal{F}] = \pi_M[\mathcal{F}] \sum_{j \in \mathcal{F}} \pi_P[j].$$

7. Numerical results

In this section, some numerical results based on section 6 are given which take advantage of the chain structure to ease the resolution.

The proposed model is built and solved using XBorne [7]. This tool is written in C language.

Several programs are available in this tool. We have used the following ones.

1. generMarkov.c: Builds the Markov chain and gives as result the transition matrix.
2. absorbing.c: Determines the absorbing states and creates a partition wherein absorbing states are at the top.
3. reorder.c: Creates a transition file so that the states of the same block are consecutive. Thus, the blocks are ranked in the order of their number.

	H=3	H=4	H=5	H=6	H=7	H=8	H=9
G=2	20	30	40	50	60	70	80
G=3	35	55	75	95	115	135	155
G=4	56	91	126	161	196	231	266
G=5	84	140	196	252	308	364	420
G=6	120	204	288	372	456	540	624
G=7	165	285	405	525	643	765	885
G=8	220	385	550	715	880	1045	1210
G=9	286	506	726	946	1166	1386	1606
G=10	364	650	936	1222	1508	1794	2080

Table 2: Markov chain size for N=3

4. split.c: Decomposes a matrix into 4 blocks.
5. fundamental.c: Calculates the fundamental matrix of an absorbent chain without a recursive class. This program implements the Sheskin algorithm [27].
6. rowSum.c: Calculates the average time before absorption.

In the following we give the results of the average times for the strategy using time-out obtained with Xborne tool. For the strategies with one optimal solution and the other with a satisfactory solution, primary results are reported in [16].

Two cases of the community numbers have been considered: $N = 3$, $N = 4$. The values of G and the values of the clock H have been varied as well. In tables 2 and 3, the corresponding sizes of the Markov chain are given according to the values of N, to the different values of g, and different values of H. In table 4, the number of absorbing states is given according to the values of N and G.

Hence from the previous results, it seems clearly that the choice of the appropriate stopping criterion and the considered values of the thresholds N, G and H impact greatly the performances of the composition process and the final QoS of each community.

Figures 4 and 5 report the average times to reach the absorbing states starting from any state of the chain by considering $N = 3$ and two cases of grade: $G = 2$ and $G = 3$. The value of the clock H is varied from 3 to 9. As expected, we notice that the time elapsed to reach the absorbing states is greater when the time-out threshold is increased. Indeed, this is quite logical

	H=4	H=5	H=6	H=7	H=8	H=9	H=10
G=2	35	50	65	80	95	110	125
G=3	70	105	140	175	210	245	280
G=4	126	196	266	336	406	476	546
G=5	210	336	462	588	714	840	966
G=6	330	540	750	960	1170	1380	1590
G=7	495	825	1155	1485	1815	2145	2475
G=8	715	1210	1705	2200	2695	3190	3685
G=9	1001	1716	2431	3146	3861	4576	5291
G=10	1365	2366	3367	4368	5369	6370	7371

Table 3: Markov chain size for N=4

	G=2	G=3	G=4	G=5	G=6	G=7	G=8	G=9	G=10
N=3	10	20	35	56	84	120	165	220	286
N=4	15	35	70	126	210	330	495	715	1001

Table 4: Number of absorbing states

because the Markov chain is larger when the threshold is higher; this makes the depth of the chain more important.

Figures 6 and 7 show the average times needed to reach the absorbing states starting from any state of the chain by considering $N = 3$ and two cases of grade: $G = 6$ and $G = 9$.

Numerical results show that more the number of grade increases, more the delay is higher. This was expected because the number of chain states increases with the maximum value of the grade G.

The case where $N = 4$ is considered as well by varying different values of G and H. On figures 8 and 9, we plot the average times needed to reach the absorbing states starting from any state of the chain. The same remarks as in the case $N = 3$ can be stated for $N = 4$

Also figures 10 and 11 depict the average times needed to reach one of the absorbing states starting from any state of the chain.

According to the numerical results detailed above, it can be concluded that, for a given community size, the delays to reach an absorbing state increase as a function of the variation in the value of G and the time-out threshold H. Depending on the type of service, it is therefore necessary to choose the values G and H carefully. A compromise must therefore be sought

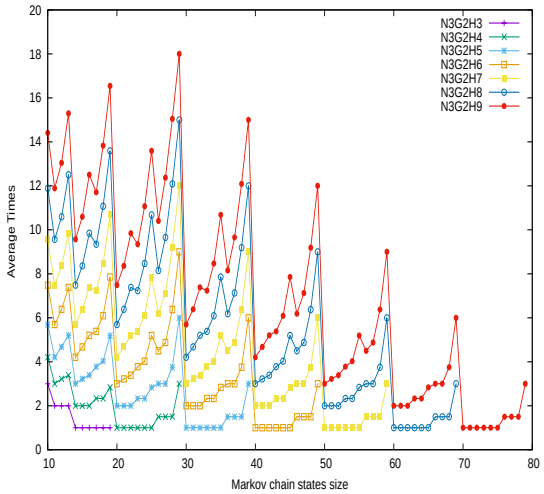

Figure 4: **Average Time** $N = 3$ **et** $G = 2$

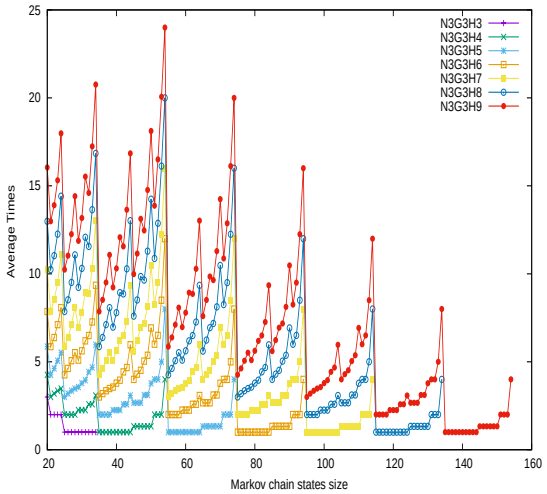

Figure 5: **Average Time** $N = 3$ **et** $G = 3$

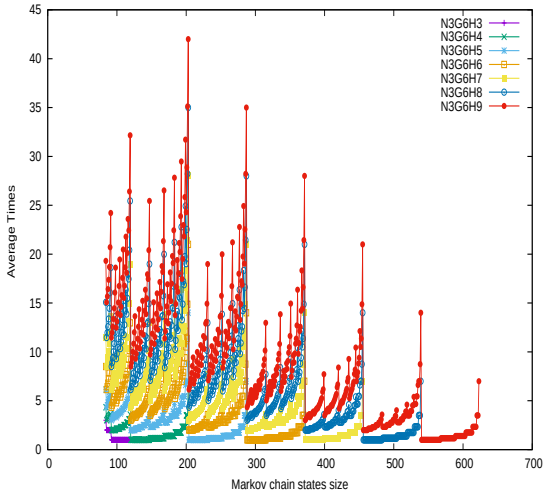

Figure 6: **Average Time** $N = 3$ **et** $G = 6$

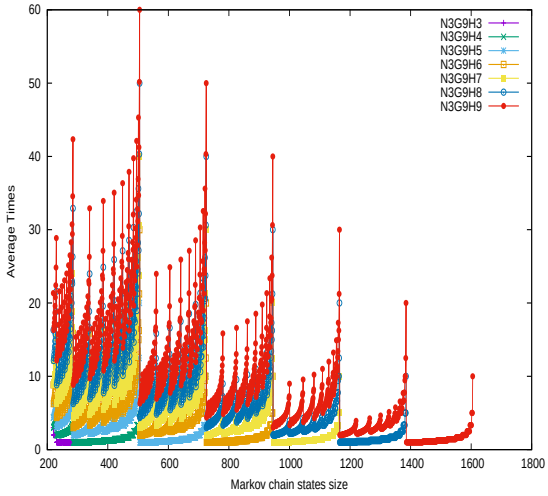

Figure 7: **Average Time** $N = 3$ **et** $G = 9$

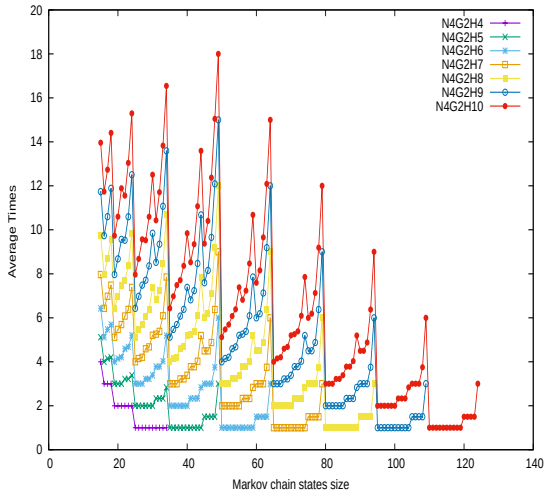

Figure 8: **Average Time** $N = 4$ **et** $G = 2$

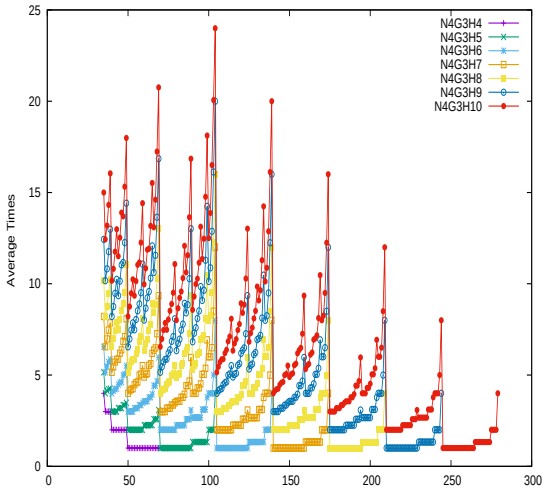

Figure 9: **Average Time** $N = 4$ **et** $G = 3$

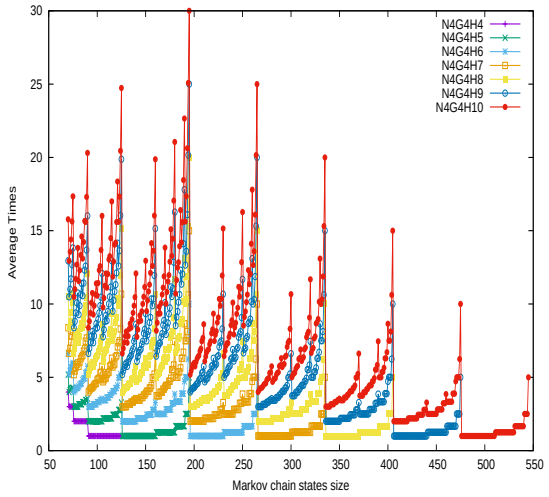

Figure 10: **Average Time** $N = 4$ **et** $G = 4$

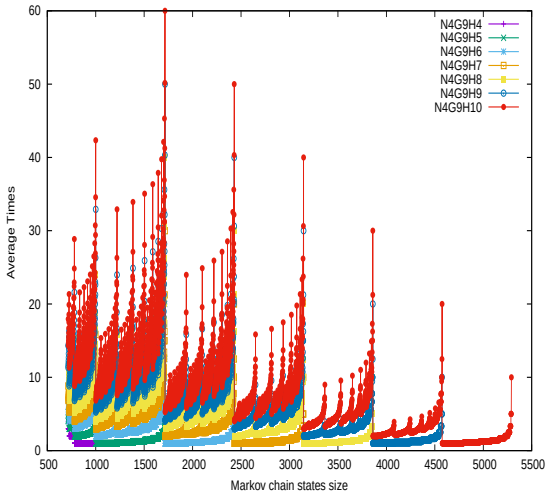

Figure 11: **Average Time** $N = 4$ **et** $G = 9$

between:

- promoting a service with the highest grade, that however needs a larger time-out threshold to potentially acquire it and;
- promoting a service with a satisfactory grade that can be acquired within a reasonable time-out threshold.

8. Conclusion

In this paper, we addressed the particular problem of the QoS aware service composition with multiple consumers in a dynamic environment. For this purpose, we developed various models based on Discrete Time Markov Chain to design several strategies to follow during the composition selection process, which are: selection of the service with the highest grade; selection of the service with the satisfactory grade; and the selection of the service within a time-out. The theoretical modelling of the problem together with the consistency proof of the developed DTMC models have been presented and discussed. We reported analytical results obtained using the Xborne tool, where we provided for each case, the average time to reach a given QoS for a community of consumers. The obtained results allow to understand the system behaviour when varying the different parameters, in order to adapt them according to user requirements and the system load. Future work will lead us to investigate more constraints and additional policies in the performance evaluation of such systems.

References

[1] O. AbdelWahab, J. Bentahar, H. Otrok, A. Mourad. A Stackelberg game for distributed formation of business-driven services communities. Expert Syst. Appl. 45: 359-372, 2016.

[2] I. Boussaid. Improvement of metaheuristics for continuous optimization. These. Universit Paris-Est, juin 2013. url: https://tel.archivesouvertes.fr/tel-00952774

[3] B. Benatallah, Q. Sheng, and M. Dumas. The Self-Serv environment for Web services composition. IEEE Internet Computing, vol. 7, pp. 4048, Jan. 2003.

[4] P. Buchholz. Exact and Ordinary Lumpability in Finite Markov Chains. Journal of Applied Probability, V31, N1, pp 59-75, 1994.

[5] C. Cherifi, Y. Rivierre, and J.-F. Santucci. A Community Based Algorithm for Large Scale Web Service Composition. arXiv:1305.0187[cs], arXiv: 1305.0187. May 2013.

[6] H. Dong, X. Yang, X. Teng, Y. Sha. A diversity reserved quantum particle swarm optimization algorithm for mmkp, in: 2016 IEEE/ACIS, 2016, pp. 1-7. doi:10.1109/ICIS.2016.7550941.

[7] J.M Fourneau, Y. Ait El Mahjoub, F. Quessette, D. Vekris. XBorne 2016: A Brief Introduction. ISCIS, pages 134-141, Springer, 2016.

[8] V. Gabrel, M. Manouvrier, K. Moreau, C. Murat. QoS aware automatic syntactic service composition problem: Complexity and resolution. Future Generation Comp. Syst. 80: 311-321, 2018.

[9] K. Hashmi, A.Alhosban, Z. Malik, B. Medjahed et S. Benbernou. Automated negotiation among web services. In Web Services Foundations. Springer, 2014, p. 451482.

[10] C. Jatoth, G. R. Gangadharan, and R. Buyya. Computational intelligence based QoS aware Web service composition: A systematic literature review. IEEE Transactions on Services Computing, vol. 10, pp. 475492, May 2017.

[11] N. Jozefowiez. Optimisation combinatoire multi-objectif : des mthodes aux problmes, de la Terre (presque) la Lune. Habilitation diriger des recherches. Institut National Polytechnique de Toulouse (INP Toulouse), dc. 2013.

[12] J.G . Kemeny, J.L. Snell. Finite Markov Chains. Van Nostrand, New York, 1960.

[13] A. L. Lemos, F. Daniel, B. Benatallah. Web service composition: A survey of techniques and tools. ACM Comput. Surv. 48 (3), 33:1-33:41, 2015.

[14] J.C. Lima, R.C.A. da Rocha, F.M. Costa. An Approach for QoS aware Selection of Shared Services for Multiple Service Choreographies. SOSE, pages 221-230. 2016.

[15] B. Medjahed, Z.Malik, S.Benbernou. On the Composability of Semantic Web Services. Web Services Foundations 2014: 137-160

[16] L. Mokdad, J.M Fourneau and A. Abdelli. Performance evaluation of the QoS-aware Web service composition with communities of consumers, IEEE Globecom 2019.

[17] M. Moghaddam and J. G. Davis. Service selection in web service composition: A comparative review of existing approaches. in Web Services Foundations, pp. 321346, Springer, 2014.

[18] A. Mousa, J. Bentahar. An Efficient QoS aware Web Services Selection Using Social Spider Algorithm. FNC/MobiSPC 2016: 176-182

[19] T.G. Robertazzi. Recursive solution of a class of non-product form protocol models. In INFOCOM, 1989.

[20] M. Sathya, P. Dhavachelvan and K. Vivekanandan. Egalitarian based Negotiation model for QoS based Web Service Selection. In International Journal of Soft Computing 8.2, p. 134142, 2013.

[21] W. Serrai, A. Abdelli, L. Mokdad, and Y. Hammal. Towards an efficient and a more accurate web service selection using MCDM methods. Journal of Computational Science, vol. 22, pp. 253 267, 2017.

[22] Y. Shi and X. Chen. A survey on QoS aware web service composition. in 2011 Third International Conference on Multimedia Information Networking and Security, pp. 283287, Nov 2011.

[23] F. Siala, K. Ghdira: A Multi-Agent selection of Web Service providers driven by composite QoS. ISCC 2011: 55-60, 2011.

[24] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu. Web services composition: A decades overview. Information Sciences, vol. 280, pp. 218238, 2014.

[25] A. Strunk. QoS-Aware Service Composition: A Survey. Eighth IEEE European Conference on Web Services Cyprus 2010.

[26] W.J. Stewart. Introduction to the Numerical Solution of Markov Chains, published by Princeton university press, 1995.

[27] T. J. Sheskin. A Markov partitioning algorithm for computing steady-state probabilities, Operat. Res. 33, pp. 228-235, 1985.

[28] K.S. Trivedi. Probability and Statistics with Reliability, Queueing and Computer Science Applications, Second Edition, Wiley, 2002.

[29] V. Patankar and R. Hewett. Automated negotiations in web service procurement. In proc of IEEE ICIW, p. 620625, 2008.

[30] P. Wang, J. Lan, X. Zhang, Y. Hu, S. Chen. Dynamic function composition for network service chain: Model and optimization, Computer Networks 92 (2015).

[31] P. Wang, K.M. Chao, C.C. Lo. On optimal decision for QoS-aware composite service selection. Expert Syst. Appl. 37(1): 440-449, 2010.

[32] Y. Wang, J. Zhang, and J. Vassileva. Effective Web Service Selection via Communities Formed by Super-Agents. pp. 549556, IEEE, Aug. 2010.

[33] Y. Wang, Q. He, Y. Yang. QoS aware Service Recommendation for Multi-tenant SaaS on the Cloud. SCC 2015: 178-185, 2015.

[34] Q. Wu, Q. Zhu, X Jian, F. Ishikawa. Broker-based SLA-aware composite service provisioning. Journal of Systems and Software 96: 194-201, 2014.

[35] Y. Xia, C. Gao, and J. Li. A stochastic local search heuristic for the multidimensional multiple choice knapsack problem. in Bio-Inspired Computing Theories and Applications, pp. 513522, Springer, 2015.

[36] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end QoS constraints. ACM Trans. Web, vol. 1, 2007.