Performance Evaluation of Stochastic Bipartite Matching Models

Céline Comte, Jan-Pieter Dorsman

Performance Evaluation of Stochastic Bipartite Matching Models*

Céline Comte†[1] and Jan-Pieter Dorsman[2]

[1]Eindhoven University of Technology, The Netherlands, c.m.comte@tue.nl
[2]University of Amsterdam, The Netherlands, j.l.dorsman@uva.nl

December 6, 2021

Abstract

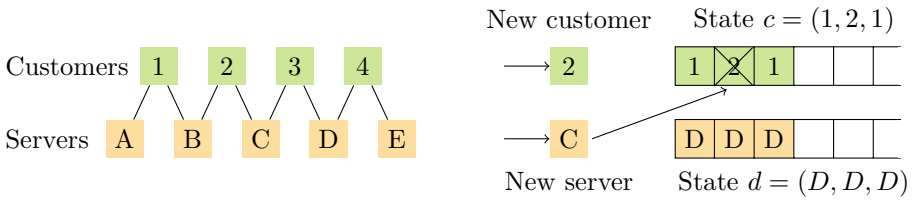
We consider a stochastic bipartite matching model consisting of multi-class customers and multi-class servers. Compatibility constraints between the customer and server classes are described by a bipartite graph. Each time slot, exactly one customer and one server arrive. The incoming customer (resp. server) is matched with the earliest arrived server (resp. customer) with a class that is compatible with its own class, if there is any, in which case the matched customer-server couple immediately leaves the system; otherwise, the incoming customer (resp. server) waits in the system until it is matched. Contrary to classical queueing models, both customers and servers may have to wait, so that their roles are interchangeable. While (the process underlying) this model was already known to have a product-form stationary distribution, this paper derives a new compact and manageable expression for the normalization constant of this distribution, as well as for the waiting probability and mean waiting time of customers and servers. We also provide a numerical example and make some important observations.
Keywords— bipartite matching models, order-independent queues, performance analysis, product-form stationary distribution

1 Introduction

Stochastic matching models typically consist of items of multiple classes that arrive at random instants to be matched with items of other classes. In the same spirit as classical (static) matching models, stochastic models encode compatibility constraints between items using a graph on the classes. This allows for the modeling of many matching applications that are stochastic in nature, such as organ transplants where not every patient is compatible with every donor organ.

In the literature on stochastic matching, a rough distinction is made between *bipartite* and *non-bipartite* models. In a bipartite matching model, the graph that describes compatibility relations between item classes is bipartite. In this way, item classes can be divided into two groups called *customers* and *servers*, so that customers (resp. servers) cannot be matched with one another. This is the variant that we consider in this paper. It is discrete-time in nature and assumes that, every time unit, exactly one customer and one server arrive. The classes of incoming customers and servers are drawn independently from each other, and they are also independent and identically distributed across time units. Following [7, 8], we adopt the common first-come-first-matched policy, whereby an arriving customer (resp. server) is matched with the earliest arriving compatible server (resp. customer). A toy example is shown in Figure 1. This model is equivalent to the *first-come-first-served infinite bipartite matching model*, studied in [1, 2, 8], which can be used to describe the evolution of waiting lists in public-housing programs and adoption agencies for instance [8].

*The final authenticated version is available online at https://doi.org/10.1007/978-3-030-91825-5_26.
†Corresponding author.



(a) Compatibility graph

(b) First-come-first-matched policy

Figure 1: A stochastic bipartite matching model with a set $\mathcal{I} = \{1, 2, 3, 4\}$ of customer classes and a set $\mathcal{K} = \{A, B, C, D, E\}$ of server classes.

In contrast, in a stochastic *non*-bipartite matching model, item classes cannot be divided into two groups because the compatibility graph is non-bipartite. Another notable difference is that only one item arrives at each time slot, the classes of successive items being drawn independently from the same distribution. While [15] derived stability conditions for this non-bipartite model, [16] showed that the stationary distribution of the process underlying this model has a product form. The recent work [10] built on the latter result to derive closed-form expressions for several performance metrics, by also exploiting a connection with order-independent (loss) queues [5, 14]. The present work seeks to provide a similar analysis for the above-mentioned bipartite model.

More specifically, we derive closed-form expressions for several performance metrics in the stochastic bipartite model studied in [1]. While earlier studies [3, 8] were skeptical about the tractability of the stationary distribution corresponding to this variant, [1] showed that this stationary distribution in fact possesses the product-form property, thus paving the way for an analysis similar to that of [10]. That is, we use techniques from order-independent (loss) queues [5, 14] and other product-form models with compatibility constraints (cf. [12] for a recent overview) to analyze the stochastic bipartite matching model.

The rest of this paper is structured as follows. In Section 2, we introduce the model and cast it in terms of a framework commonly used for analysis of product-form models. We then provide a performance evaluation of this model. More specifically, we first derive an alternative closed-form expression for the normalization constant of the stationary distribution. While the computational complexity of this expression is prohibitive for instances with many classes (as was the case for the expression derived in [1]), it draws the relation with product-form queues and paves the way for heavy-traffic analysis. Furthermore, it allows us to directly derive recursive expressions for several other performance metrics, such as the probability that incoming customers or servers have to wait and the mean number of customers and servers that are waiting. To the best of the authors' knowledge, this paper is the first to provide expressions for these performance metrics. This analysis is presented in Section 3. Finally, Section 4 numerically studies a model instance and makes some important observations.

2 Model and preliminary results

In Section 2.1, we describe a stochastic bipartite matching model in which items of two groups, called customers and servers, arrive randomly and are matched with one another. As mentioned earlier, this model is analogous to that introduced in [8] and further studied in [1, 2, 3, 7]. Section 2.2 focuses on a discrete-time Markov chain that describes the evolution of this model. Finally, Section 2.3 recalls several results that are useful for the analysis of Section 3.

2.1 Model and notation

Bipartite compatibility graph Consider a finite set $\mathcal{I}$ of I customer classes and a finite set $\mathcal{K}$ of K server classes. Also consider a connected bipartite graph on the sets $\mathcal{I}$ and $\mathcal{K}$. For each $i \in \mathcal{I}$ and $k \in \mathcal{K}$, we write $i \sim k$ if there is an edge between nodes i and k in this graph, and $i \nsim k$ otherwise. This bipartite graph is

called the *compatibility* graph of the model. It describes the compatibility relations between customers and servers in the sense that, for each $i \in \mathcal{I}$ and $k \in \mathcal{K}$, a class-i customer and a class-k server can be matched with one another if and only if there is an edge between the corresponding nodes in the compatibility graph. An example is shown in Figure 1a. To simplify reading, we consistently use the letters $i, j \in \mathcal{I}$ for customer classes and $k, \ell \in \mathcal{K}$ for server classes.

Discrete-time stochastic matching Unmatched customers and servers are stored in two separate queues in their arrival order. In Figure 1b, items are ordered from the oldest on the left to the newest on the right, and each item is labeled by its class. The two queues are initially empty. Time is slotted and, during each time slot, exactly one customer *and* exactly one server arrive. The incoming customer belongs to class i with probability $\lambda_i > 0$, for each $i \in \mathcal{I}$, and the incoming server belongs to class k with probability $\mu_k > 0$ for each $k \in \mathcal{K}$, with $\sum_{i\in\mathcal{I}} \lambda_i = \sum_{k\in\mathcal{K}} \mu_k = 1$. The classes of incoming customers and servers are independent within and across time slots. The matching policy, called *first-come-first-matched*, consists of applying the following four steps upon each arrival:

1. Match the incoming customer with the compatible unmatched server that has been in the queue the longest, if any.
2. Match the incoming server with the compatible unmatched customer that has been in the queue the longest, if any.
3. If neither the incoming customer nor the incoming server can be matched with unmatched items, match them together if they are compatible.
4. If an incoming customer and/or incoming server remains unmatched after the previous steps, it is appended to the back of its respective queue.

When two items are matched with one another, they immediately disappear. In the example of Figure 1b, the couple (2, C) arrives while the sequence of unmatched customer and server classes are $(1, 2, 1)$ and (D, D, D). According to the compatibility graph of Figure 1a, class C is compatible with class 2 but not with class 1. Therefore, the incoming class-C server is matched with the second oldest unmatched customer, of class 2. The incoming class-2 customer is not matched with any present item (even if it is compatible with the incoming class-C server), therefore it is appended to the queue of unmatched customers. After this transition, the sequence of unmatched customer classes becomes $(1, 1, 2)$, while the sequence of unmatched server classes is unchanged.

Remark 1. If we would consider the random sequences of classes of incoming customers and servers, we would retrieve the state descriptor of the infinite bipartite matching model introduced in [8] and studied in [1, 2, 3, 7]. For analysis purposes, we however adopted the above-introduced state descriptor consisting of the sequences of (waiting) unmatched customers and servers, corresponding to the *natural pair-by-pair FCFS Markov chain* introduced in [1, Section 2].

Set notation The following notation will be useful. Given two sets $\mathcal{A}$ and $\mathcal{B}$, we write $\mathcal{A} \subseteq \mathcal{B}$ if $\mathcal{A}$ is a subset of $\mathcal{B}$ and $\mathcal{A} \subsetneq \mathcal{B}$ if $\mathcal{A}$ is a proper subset of $\mathcal{B}$. For each $i \in \mathcal{I}$, we let $\mathcal{K}_i \subseteq \mathcal{K}$ denote the set of server classes that can be matched with class-i customers. Similarly, for each $k \in \mathcal{K}$, we let $\mathcal{I}_k \subseteq \mathcal{I}$ denote the set of customer classes that can be matched with class-k servers. For each $i \in \mathcal{I}$ and $k \in \mathcal{K}$, the statements $i \sim k$, $i \in \mathcal{I}_k$, and $k \in \mathcal{K}_i$ are equivalent. In Figure 1a for instance, we have $\mathcal{K}_1 = \{A, B\}$, $\mathcal{K}_2 = \{B, C\}$, $\mathcal{K}_3 = \{C, D\}$, $\mathcal{K}_4 = \{D, E\}$ $\mathcal{I}_A = \{1\}$, $\mathcal{I}_B = \{1, 2\}$, $\mathcal{I}_C = \{2, 3\}$, $\mathcal{I}_D = \{3, 4\}$, and $\mathcal{I}_E = \{4\}$. With a slight abuse of notation, for each $\mathcal{A} \subseteq \mathcal{I}$, we let $\lambda(\mathcal{A}) = \sum_{i\in\mathcal{A}} \lambda_i$ denote the probability that the class of an incoming customer belongs to $\mathcal{A}$ and $\mathcal{K}(\mathcal{A}) = \bigcup_{i\in\mathcal{A}} \mathcal{K}_i$ the set of server classes that are compatible with customer classes in $\mathcal{A}$. Similarly, for each $\mathcal{A} \subseteq \mathcal{K}$, we write $\mu(\mathcal{A}) = \sum_{k\in\mathcal{A}} \mu_k$ and $\mathcal{I}(\mathcal{A}) = \bigcup_{k\in\mathcal{A}} \mathcal{I}_k$. In particular, we have $\lambda(\mathcal{I}) = \mu(\mathcal{K}) = 1$, $\mathcal{K}(\mathcal{I}) = \mathcal{K}$, and $\mathcal{I}(\mathcal{K}) = \mathcal{I}$.

2.2 Discrete-time Markov chain

We now consider a Markov chain that describes the evolution of the system.

System state We consider the couple (c, d), where $c = (c_1, \dots, c_n) \in \mathcal{I}^*$ is the sequence of unmatched customer classes, ordered by arrival, and $d = (d_1, \dots, d_n) \in \mathcal{K}^*$ is the sequence of unmatched server classes, ordered by arrival. In particular, c_1 is the class of the oldest unmatched customer, if any, and d_1 is the class of the oldest unmatched server, if any. The notation $\mathcal{I}^*$ (resp. $\mathcal{K}^*$) refers to the Kleene star on $\mathcal{I}$ (resp. $\mathcal{K}$), that is, the set of sequences of elements in $\mathcal{I}$ (resp. $\mathcal{K}$) with a length that is finite but arbitrarily large [11, Chapter 1, Section 2]. As we will see later, the matching policy guarantees that the numbers of unmatched customers and servers are always equal to each other, and consequently the integer n will be called the *length* of the state. The empty state, with $n = 0$, is denoted by $\varnothing$.

The evolution of this state over time defines a (discrete-time) Markov chain that is further detailed below. For each sequence $c = (c_1, \dots, c_n) \in \mathcal{I}^*$, we let $|c| = n$ denote the length of sequence c, $|c|_i$ the number of occurrences of class i in sequence c, for each $i \in \mathcal{I}$, and, with a slight abuse of notation, $\{c_1, \dots, c_n\}$ the set of classes that appear in sequence c (irrespective of their multiplicity). Analogous notation is introduced for each sequence $d = (d_1, \dots, d_n) \in \mathcal{K}^*$.

Transitions Each transition of the Markov chain is triggered by the arrival of a customer-server couple. We distinguish five types of transitions depending on their impact on the queues of unmatched customers and servers:

$-/-$ The incoming customer is matched with an unmatched server and the incoming server is matched with an unmatched customer.

$\pm/=$ The incoming customer cannot be matched with any present server but the incoming server is matched with an unmatched customer.

$=/\pm$ The incoming customer is matched with a present server but the incoming server cannot be matched with any present customer.

$=/=$ Neither the incoming customer nor the incoming server can be matched with an unmatched item, but they are matched with one another.

$+/+$ Neither the incoming customer nor the incoming server can be matched with an unmatched item, and they cannot be matched with one another.

Labels indicate the impact of the corresponding transition. For instance, a transition $-/-$ leads to a deletion $(-)$ in the customer queue and a deletion $(-)$ in the server queue, while a transition $\pm/=$ leads to a replacement $(\pm)$ in the customer queue and no modification in the server queue $(=)$. Transitions $-/-$ reduce the lengths of both queues by one, transitions $\pm/=$, $=/\pm$, and $=/=$ leave the queue lengths unchanged, and transitions $+/+$ increase the lengths of both queues by one. Note that the numbers of unmatched customers and servers are always equal to each other. We omit the transition probabilities, as we will rely on an existing result giving the stationary distribution of the Markov chain.

State space The greediness of the matching policy prevents the queues from containing an unmatched customer and an unmatched server that are compatible. Therefore, the state space of the Markov chain is the subset of $\mathcal{I}^* \times \mathcal{K}^*$ given by

$$\Pi = \bigcup_{n=0}^{\infty} \{(c, d) \in \mathcal{I}^n \times \mathcal{K}^n : c_p \nsim d_q \text{ for each } p, q \in \{1, \dots, n\}\}.$$

The Markov chain is irreducible. Indeed, using the facts that the compatibility graph is connected, that $\lambda_i > 0$ for each $i \in \mathcal{I}$, and that $\mu_k > 0$ for each $k \in \mathcal{K}$, we can show that the Markov chain can go from any state $(c, d) \in \Pi$ to any state $(c', d') \in \Pi$ via state $\varnothing$ in $|c| + |c'| = |d| + |d'|$ jumps.

Remark 2. We can also consider the following continuous-time variant of the model introduced in Section 2.1. Instead of assuming that time is slotted, we can assume that customer-server couples arrive according to a Poisson process with unit rate. If the class of the incoming customers and servers are drawn independently at random, according to the probabilities λ_i for $i \in \mathcal{I}$ and μ_k for $k \in \mathcal{K}$, then the rate diagram of the continuous-time Markov chain describing the evolution of the sequences of unmatched items is identical to the transition diagram of the Markov chain introduced above. Consequently, the results recalled in Section 2.3 and those derived in Section 3 can be applied without any modification to this continuous-time Markov chain.

2.3 Stability conditions and stationary distribution

For purposes of later analysis, we now state the following theorem, which was proved in [3, Theorem 3] and [1, Lemma 2 and Theorems 2 and 8].

Theorem 1. *The stationary measures of the Markov chain associated with the system state are of the form*

$$\pi(c,d) = \pi(\varnothing) \prod_{p=1}^{n} \frac{\lambda_{c_p}}{\mu(\mathcal{K}(\{c_1,\dots,c_p\}))} \frac{\mu_{d_p}}{\lambda(\mathcal{I}(\{d_1,\dots,d_p\}))}, \quad (c,d) \in \Pi. \tag{1}$$

The system is stable, in the sense that this Markov chain is ergodic, if and only if one of the following two equivalent conditions is satisfied:

$$\lambda(\mathcal{A}) < \mu(\mathcal{K}(\mathcal{A})) \text{ for each non-empty set } \mathcal{A} \subsetneq \mathcal{I}, \tag{2}$$

$$\mu(\mathcal{A}) < \lambda(\mathcal{I}(\mathcal{A})) \text{ for each non-empty set } \mathcal{A} \subsetneq \mathcal{K}. \tag{3}$$

In this case, the stationary distribution of the Markov chain associated with the system state is given by (1)*, with the normalization constant*

$$\pi(\varnothing) = \left(\sum_{(c,d)\in\Pi} \prod_{p=1}^{n} \frac{\lambda_{c_p}}{\mu(\mathcal{K}(\{c_1,\dots,c_p\}))} \frac{\mu_{d_p}}{\lambda(\mathcal{I}(\{d_1,\dots,d_p\}))} \right)^{-1}. \tag{4}$$

The states of the two queues are not independent in general because their lengths are equal. However, (1) shows that these two queue states are *conditionally* independent *given* the number n of unmatched items. This property will contribute to simplify the analysis in Section 3.

Remark 3. The stationary measures (1) seem identical to the stationary measures associated with another queueing model, called an FCFS-ALIS parallel queueing model [2, 4]. The only (crucial) difference lies in the definition of the state space of the corresponding Markov chain. In particular, our model imposes that the lengths of the two queues are equal to each other. In contrast, in the FCFS-ALIS parallel queueing model, there is an upper bound on the number of unmatched servers, while the number of customers can be arbitrarily large. This difference significantly changes the analysis. The analysis that we propose in Section 3 is based on the resemblance with another queueing model, called a multi-server queue for simplicity, that was introduced in [9, 13].

3 Performance evaluation by state aggregation

We now assume that the stability conditions (2)–(3) are satisfied, and we let π denote the stationary distribution, recalled in Theorem 1, of the Markov chain of Section 2.2. Sections 3.2 to 3.4 provide closed-form expressions for several performance metrics, based on a method explained in Section 3.1. The time complexity to implement these formulas and the relation with related works [1, 3] are discussed in Section 3.5. The reader who is not interested in understanding the proofs can move directly to Section 3.2.

3.1 Partition of the state space

A naive application of (4) does not allow calculation of the normalization constant, nor any other long-run performance metric as a result, because the state-space Π is infinite. To circumvent this, we define a partition of the state space.

Partition of the state space Π Let $\mathbb{I}$ denote the family of sets $\mathcal{A} \subseteq \mathcal{I} \cup \mathcal{K}$ such that $\mathcal{A}$ is an independent set of the compatibility graph and the sets $\mathcal{A} \cap \mathcal{I}$ and $\mathcal{A} \cap \mathcal{K}$ are non-empty. Also let $\mathbb{I}_0 = \mathbb{I} \cup \{\emptyset\}$. For each $\mathcal{A} \in \mathbb{I}_0$, we let $\Pi_{\mathcal{A}}$ denote the set of couples $(c, d) \in \Pi$ such that $\{c_1, \ldots, c_n\} = \mathcal{A} \cap \mathcal{I}$ and $\{d_1, \ldots, d_n\} = \mathcal{A} \cap \mathcal{K}$; in other words, $\Pi_{\mathcal{A}}$ is the set of states such that the set of unmatched classes is $\mathcal{A}$. We can show that $\{\Pi_{\mathcal{A}}, \mathcal{A} \in \mathbb{I}_0\}$ forms a partition of Π, and in particular

$$\Pi = \bigcup_{\mathcal{A} \in \mathbb{I}_0} \Pi_{\mathcal{A}}.$$

The first cornerstone of the subsequent analysis is the observation that, for each $(c, d) \in \Pi_{\mathcal{A}}$, we have $\mu(\mathcal{K}(\{c_1, \ldots, c_n\})) = \mu(\mathcal{K}(\mathcal{A} \cap \mathcal{I}))$ and $\lambda(\mathcal{I}(\{d_1, \ldots, d_n\})) = \lambda(\mathcal{I}(\mathcal{A} \cap \mathcal{K}))$. In anticipation of Section 3.2, for each $\mathcal{A} \in \mathbb{I}$, we let

$$\Delta(\mathcal{A}) = \mu(\mathcal{K}(\mathcal{A} \cap \mathcal{I}))\lambda(\mathcal{I}(\mathcal{A} \cap \mathcal{K})) - \lambda(\mathcal{A} \cap \mathcal{I})\mu(\mathcal{A} \cap \mathcal{K}). \tag{5}$$

One can verify that $\Delta(\mathcal{A}) > 0$ for each $\mathcal{A} \in \mathbb{I}$ if and only if the stability conditions (2)–(3) are satisfied. The product $\lambda(\mathcal{A} \cap \mathcal{I})\mu(\mathcal{A} \cap \mathcal{K})$ is the probability that an incoming client-server couple has its classes in $\mathcal{A}$, while the product $\mu(\mathcal{K}(\mathcal{A} \cap \mathcal{I}))\lambda(\mathcal{I}(\mathcal{A} \cap \mathcal{K}))$ is the probability that an incoming client-server couple can be matched with clients and servers whose classes belong to $\mathcal{A}$. By analogy with the queueing models in [10, 13], the former product can be seen as the "arrival rate" of the classes in $\mathcal{A}$, while the latter product can be seen as the maximal "departure rate" of these classes.

Partition of the subsets $\Pi_{\mathcal{A}}$ The second cornerstone of the analysis is a partition of the set $\Pi_{\mathcal{A}}$ for each $\mathcal{A} \in \mathbb{I}$. More specifically, for each $\mathcal{A} \in \mathbb{I}$, we have

$$\Pi_{\mathcal{A}} = \bigcup_{i \in \mathcal{A} \cap \mathcal{I}} \bigcup_{k \in \mathcal{A} \cap \mathcal{K}} \left(\Pi_{\mathcal{A}} \cup \Pi_{\mathcal{A} \setminus \{i\}} \cup \Pi_{\mathcal{A} \setminus \{k\}} \cup \Pi_{\mathcal{A} \setminus \{i,k\}}\right) \cdot (i, k), \tag{6}$$

where $\mathcal{S} \cdot (i, k) = \{((c_1, \ldots, c_n, i), (d_1, \ldots, d_n, k)) : ((c_1, \ldots, c_n), (d_1, \ldots, d_n)) \in \mathcal{S}\}$ for each $\mathcal{S} \subseteq \Pi$, $i \in \mathcal{I}$, and $k \in \mathcal{K}$, and the unions are disjoint. Indeed, for each $(c, d) \in \Pi_{\mathcal{A}}$, the sequence $c = (c_1, \ldots, c_n)$ can be divided into a prefix $(c_1, \ldots, c_{n-1})$ and a suffix $i = c_n$; the suffix can take any value in $\mathcal{A} \cap \mathcal{I}$, while the prefix satisfies $\{c_1, \ldots, c_{n-1}\} = \mathcal{A} \cup \mathcal{I}$ or $\{c_1, \ldots, c_{n-1}\} = (\mathcal{A} \setminus \{i\}) \cup \mathcal{I}$. Similarly, for each $(c, d) \in \Pi_{\mathcal{A}}$, the sequence $d = (d_1, \ldots, d_n)$ can be divided into a prefix $(d_1, \ldots, d_{n-1})$ and a prefix $k = d_n$; the prefix can take any value in $\mathcal{A} \cap \mathcal{K}$, while the prefix satisfies $\{d_1, \ldots, d_{n-1}\} = \mathcal{A} \cap \mathcal{K}$ or $\{d_1, \ldots, d_{n-1}\} = (\mathcal{A} \setminus \{k\}) \cap \mathcal{K}$.

3.2 Normalization constant

The first performance metric that we consider is the probability that the system is empty. According to (4), this is also the inverse of the normalization constant. With a slight abuse of notation, we first let

$$\pi(\mathcal{A}) = \sum_{(c,d) \in \Pi_{\mathcal{A}}} \pi(c, d), \quad \mathcal{A} \in \mathbb{I}_0.$$

To simplify notation, we adopt the convention that $\pi(\mathcal{A}) = 0$ if $\mathcal{A} \notin \mathbb{I}_0$. The following proposition, combined with the normalization equation $\sum_{\mathcal{A} \in \mathbb{I}_0} \pi(\mathcal{A}) = 1$, allows us to calculate the probability $\pi(\emptyset) = \pi(\varnothing)$ that the system is empty.

Proposition 1. *The stationary distribution of the set of unmatched item classes satisfies the recursion*

$$\Delta(\mathcal{A})\pi(\mathcal{A}) = \mu(\mathcal{A}\cap\mathcal{K})\sum_{i\in\mathcal{A}\cap\mathcal{I}}\lambda_i\pi(\mathcal{A}\backslash\{i\}) + \lambda(\mathcal{A}\cap\mathcal{I})\sum_{k\in\mathcal{A}\cap\mathcal{K}}\mu_k\pi(\mathcal{A}\backslash\{k\}) + \sum_{i\in\mathcal{A}\cap\mathcal{I}}\sum_{k\in\mathcal{A}\cap\mathcal{K}}\lambda_i\mu_k\pi(\mathcal{A}\backslash\{i,k\}), \quad \mathcal{A}\in\mathbb{I}. \tag{7}$$

Proof. Let $\mathcal{A}\in\mathbb{I}$. Substituting (1) into the definition of $\pi(\mathcal{A})$ yields

$$\begin{aligned}\pi(\mathcal{A}) &= \sum_{(c,d)\in\Pi_{\mathcal{A}}}\prod_{p=1}^{n}\frac{\lambda_{c_p}}{\mu(\mathcal{K}(\{c_1,\dots,c_p\}))}\frac{\mu_{d_p}}{\lambda(\mathcal{I}(\{d_1,\dots,d_p\}))},\\ &= \sum_{(c,d)\in\Pi_{\mathcal{A}}}\frac{\lambda_{c_n}}{\mu(\mathcal{K}(\mathcal{A}\cap\mathcal{I}))}\frac{\mu_{d_n}}{\lambda(\mathcal{I}(\mathcal{A}\cap\mathcal{K}))}\pi((c_1,\dots,c_{n-1}),(d_1,\dots,d_{n-1})).\end{aligned}$$

Then, by applying (6) and making a change of variable, we obtain

$$\mu(\mathcal{K}(\mathcal{A}\cap\mathcal{I}))\lambda(\mathcal{I}(\mathcal{A}\cap\mathcal{K}))\pi(\mathcal{A}) = \sum_{i\in\mathcal{A}\cap\mathcal{I}}\sum_{k\in\mathcal{A}\cap\mathcal{K}}\lambda_i\mu_k\big(\pi(\mathcal{A})+\pi(\mathcal{A}\backslash\{i\})+\pi(\mathcal{A}\backslash\{k\})+\pi(\mathcal{A}\backslash\{i,k\})\big). \tag{8}$$

The result follows by rearranging the terms. □

3.3 Waiting probability

The second performance metric that we consider is the waiting probability, that is, the probability that an item cannot be matched with another item upon arrival. The waiting probabilities of the customers and servers of each class can again be calculated using Proposition 1, as they are given by

$$\omega_i = \sum_{\mathcal{A}\in\mathbb{I}_0:\mathcal{A}\cap\mathcal{K}_i=\emptyset}\left(1-\sum_{k\in\mathcal{K}_i\backslash\mathcal{K}(\mathcal{A}\cap\mathcal{I})}\mu_k\right)\pi(\mathcal{A}), \quad i\in\mathcal{I},$$

$$\omega_k = \sum_{\mathcal{A}\in\mathbb{I}_0:\mathcal{A}\cap\mathcal{I}_k=\emptyset}\left(1-\sum_{i\in\mathcal{I}_k\backslash\mathcal{I}(\mathcal{A}\cap\mathcal{K})}\lambda_i\right)\pi(\mathcal{A}), \quad k\in\mathcal{K}.$$

If we consider the continuous-time variant described in Remark 2, these equations follow directly from the PASTA property. That this result also holds for the discrete-time variant of the model follows from the fact that the transition diagrams and stationary distributions of both models are identical.

Corollary 1 below follows from Proposition 1. It shows that the probability that both the incoming customer and the incoming server can be matched with present items (corresponding to transitions $-/-$) is equal to the probability that both the incoming customer and the incoming server have to wait (corresponding to transitions $+/+$). The proof is given in the appendix.

Corollary 1. *The following equality is satisfied:*

$$\sum_{(i,k)\in\mathcal{I}\times\mathcal{K}}\lambda_i\mu_k\sum_{\substack{\mathcal{A}\in\mathbb{I}:i\in\mathcal{I}(\mathcal{A}\cap\mathcal{K}),\\ k\in\mathcal{K}(\mathcal{A}\cap\mathcal{I})}}\pi(\mathcal{A}) = \sum_{\substack{(i,k)\in\mathcal{I}\times\mathcal{K}:\\ i\nsim k}}\lambda_i\mu_k\sum_{\substack{\mathcal{A}\in\mathbb{I}_0:i\notin\mathcal{I}(\mathcal{A}\cap\mathcal{K}),\\ k\notin\mathcal{K}(\mathcal{A}\cap\mathcal{I})}}\pi(\mathcal{A}). \tag{9}$$

This corollary means that, in the long run, the rate at which the queue lengths increase is equal to the rate at which the queue lengths decrease. Equation (9) is therefore satisfied by every matching policy that makes the system stable. This equation also has the following graphical interpretation. Consider a *busy sequence* of the system, consisting of a sequence of customer classes and a sequence of server classes that arrive between two consecutive instants when both queues are empty. We construct a bipartite graph,

whose nodes are the elements of these two sequences, by adding an edge between customers and servers that arrive at the same time or are matched with one another. An example is shown in Figure 2 for the compatibility graph of Figure 1a. If we ignore the customer-server couples that arrive at the same time and are also matched with one another, we obtain a 2-regular graph, that is, a graph where all nodes have degree two. Such a graph consists of one or more disconnected cycles. We define a left (resp. right) extremity as a vertical edge adjacent only to edges moving to the right (resp. left); such an edge represents a $+/+$ (resp. $-/-$) transition. One can verify that each cycle contains as many left extremities as right extremities. In the example of Figure 2, after eliminating the couple 1–A, we obtain two disconnected cycles. The cycle depicted with a solid line has one left extremity (2–A) and one right extremity (4–C). The cycle depicted with a dashed line also has one left extremity (3–E) and one right extremity (4–C). Since stability means that the mean length of a busy sequence is finite, combining this observation with classical results from renewal theory gives an alternative proof that (9) is satisfied by every matching policy that makes the system stable.

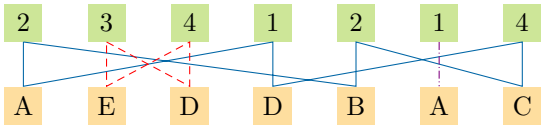


Figure 2: A busy sequence associated with the compatibility graph of Figure 1a. The arrival sequences are 2, 3, 4, 1, 2, 1, 4 and A, E, D, D, B, A, C. Each component of the corresponding bipartite graph is depicted with a different line style.

3.4 Mean number of unmatched items and mean waiting time

We now turn to the mean number of unmatched items. Proposition 2 gives a closed-form expression for the mean number of unmatched customers of each class. Proposition 3 gives a simpler expression for the mean number of unmatched customers (all classes included). The proofs are similar to that of Proposition 1, with a few technical complications, and are deferred to the appendix. Analogous results can be obtained for the servers by using the model symmetry.

Proposition 2. *For each $i \in \mathcal{I}$, the mean number of unmatched class-i customers is $L_i = \sum_{\mathcal{A}\in\mathbb{I}_0} \ell_i(\mathcal{A})$, where $\ell_i(\mathcal{A})/\pi(\mathcal{A})$ is the mean number of unmatched class-i customers given that the set of unmatched classes is $\mathcal{A}$, and satisfies the recursion*

$$\begin{aligned}\Delta(\mathcal{A})\ell_i(\mathcal{A}) = {} & \lambda_i\mu(\mathcal{A}\cap\mathcal{K})\left(\pi(\mathcal{A}) + \pi(\mathcal{A}\backslash\{i\})\right) + \lambda_i \sum_{k\in\mathcal{A}\cap\mathcal{K}} \mu_k \left(\pi(\mathcal{A}\backslash\{k\}) + \pi(\mathcal{A}\backslash\{i,k\})\right) \\ & + \mu(\mathcal{A}\cap\mathcal{K}) \sum_{j\in\mathcal{A}} \lambda_j \ell_i(\mathcal{A}\backslash\{j\}) + \lambda(\mathcal{A}\cap\mathcal{I}) \sum_{k\in\mathcal{A}\cap\mathcal{K}} \mu_k \ell_i(\mathcal{A}\backslash\{k\}) \\ & + \sum_{j\in\mathcal{A}} \sum_{k\in\mathcal{A}\cap\mathcal{K}} \lambda_j\mu_k \ell_i(\mathcal{A}\backslash\{j,k\}),\end{aligned} \tag{10}$$

for each $\mathcal{A} \in \mathbb{I}$ such that $i \in \mathcal{A}$, with the base case $\ell_i(\mathcal{A}) = 0$ if $i \notin \mathcal{A}$ and the convention that $\ell_i(\mathcal{A}) = 0$ if $\mathcal{A} \notin \mathbb{I}_0$.

Proposition 3. *The mean number of unmatched customers is $L_{\mathcal{I}} = \sum_{\mathcal{A}\in\mathbb{I}_0} \ell_{\mathcal{I}}(\mathcal{A})$, where $\ell_{\mathcal{I}}(\mathcal{A})/\pi(\mathcal{A})$ is the mean number of unmatched customers given that the set of unmatched classes is $\mathcal{A}$, and satisfies the recursion*

$$\begin{aligned}\Delta(\mathcal{A})\ell_{\mathcal{I}}(\mathcal{A}) = {} & \mu(\mathcal{K}(\mathcal{A}\cap\mathcal{I}))\lambda(\mathcal{I}(\mathcal{A}\cap\mathcal{K}))\pi(\mathcal{A}) \\ & + \mu(\mathcal{A}\cap\mathcal{K}) \sum_{i\in\mathcal{A}\cap\mathcal{I}} \lambda_i \ell_{\mathcal{I}}(\mathcal{A}\backslash\{i\}) + \lambda(\mathcal{A}\cap\mathcal{I}) \sum_{k\in\mathcal{A}\cap\mathcal{K}} \mu_k \ell_{\mathcal{I}}(\mathcal{A}\backslash\{k\})\end{aligned}$$

$$+ \sum_{i \in \mathcal{A} \cap \mathcal{I}} \sum_{k \in \mathcal{A} \cap \mathcal{K}} \lambda_i \mu_k \ell_{\mathcal{I}}(\mathcal{A} \backslash \{i,k\}). \tag{11}$$

for each $\mathcal{A} \in \mathbb{I}$*, with the base case* $\ell_{\mathcal{I}}(\emptyset) = 0$ *and the convention that* $\ell_{\mathcal{I}}(\mathcal{A}) = 0$ *for each* $\mathcal{A} \notin \mathbb{I}_0$*.*

By Little's law, the mean waiting time of class-i customers is L_i/μ_i, for each $i \in \mathcal{I}$, and the mean waiting time of customers (all classes included) is L. By following the same approach as [10, Propositions 9 and 10], we can derive, for each class, closed-form expressions for the distribution transforms of the number of unmatched items and waiting time. In the interest of space, and to avoid complicated notation, these results are omitted.

3.5 Time complexity and related work

To conclude Section 3, we briefly discuss the merit of our approach compared to the expression derived in [3, Theorem 3] and rederived in [1, Theorem 7] for the normalization constant (equal to the inverse of the probability that the system is empty). This approach relies on a Markov chain called the *server-by-server FCFS augmented matching process* in [1, Section 5.4].

Flexibility The first merit of our approach is that it can be almost straightforwardly applied to derive other relevant performance metrics. Sections 3.3 and 3.4 provide two examples: the expression of the waiting probability is a side-result of Proposition 1, while the mean waiting time follows by a derivation along the same lines. Performance metrics that can be calculated in a similar fashion include the variance of the stationary number of unmatched items of each class, the mean length of a busy sequence, and the fractions of transitions of types $-/-$, $\pm/=$, $=/\pm$, $=/=$, and $+/+$. Our approach may also be adapted to derive an alternative expression for the matching rates calculated in [3, Section 3]. Indeed, upon applying the PASTA property, it suffices to calculate the stationary distribution of the *order* of first occurrence of unmatched classes in the queues (rather than just the *set* of unmatched classes); this distribution can be evaluated by considering a refinement of the partition introduced in Section 3.1.

Time complexity Compared to the formula of [3, Theorem 3], our method leads to a lower time complexity if the number of independent sets in the compatibility graph is smaller than the cardinalities of the power sets of the sets $\mathcal{I}$ and $\mathcal{K}$. This is the case, for instance, in d-regular graphs, where the number of independent sets is at most $(2^{d+1}-1)^{(I+K)/2d}$ [18]. To illustrate this, let us first recall how to compute the probability that the system is empty using Proposition 1. The idea is to first apply (7) recursively with the base case $\pi(\emptyset) = 1$, and then derive the value of $\pi(\emptyset)$ by applying the normalization equation. For each $\mathcal{A} \in \mathbb{I}$, assuming that the values of $\pi(\mathcal{A}\backslash\{i\})$, $\pi(\mathcal{A}\backslash\{k\})$, and $\pi(\mathcal{A}\backslash\{i,k\})$ are known for each $i \in \mathcal{A} \cap \mathcal{I}$ and $k \in \mathcal{A} \cap \mathcal{K}$, evaluating $\pi(\mathcal{A})$ using (7) requires $O(I \cdot K)$ operations, where I is the number of customer classes and K is the number of server classes. The time complexity to evaluate the probability that the system is empty is therefore given by $O(T + N \cdot I \cdot K)$, where N is the number of independent sets in the compatibility graph and T is the time complexity to enumerate all maximal independent sets. The result of [17] implies that the time complexity to enumerate all maximal independent sets in the (bipartite) compatibility graph $O((I+K) \cdot I \cdot K \cdot M)$, where M is the number of maximal independent sets. Overall, the time complexity to evaluate the normalization constant using Proposition 1 is $O(I \cdot K \cdot ((I+K) \cdot M + N))$.

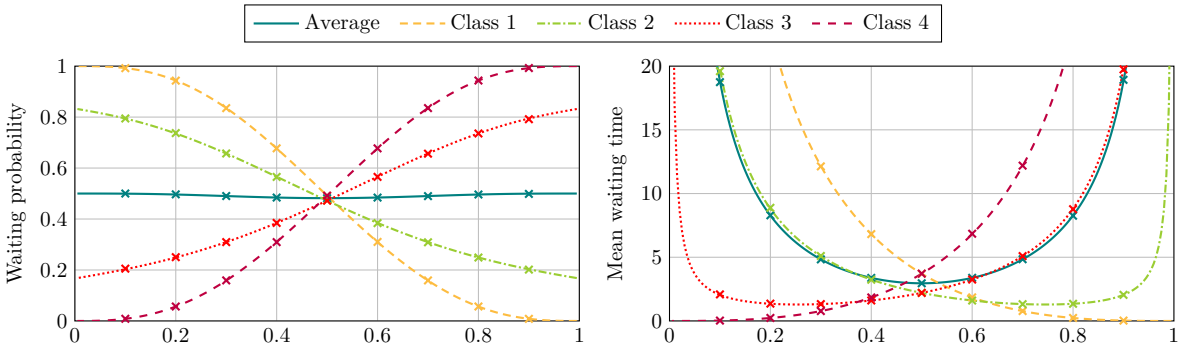
In comparison, the time complexity to evaluate the normalization constant using [3, Theorem 3] is $O((I+K) \cdot 2^{\min(I,K)})$ if we implement these formulas recursively, in a similar way as in [6]. Our method thus leads to a lower time complexity if the number of independent sets of the compatibility graph is small.

4 Numerical evaluation

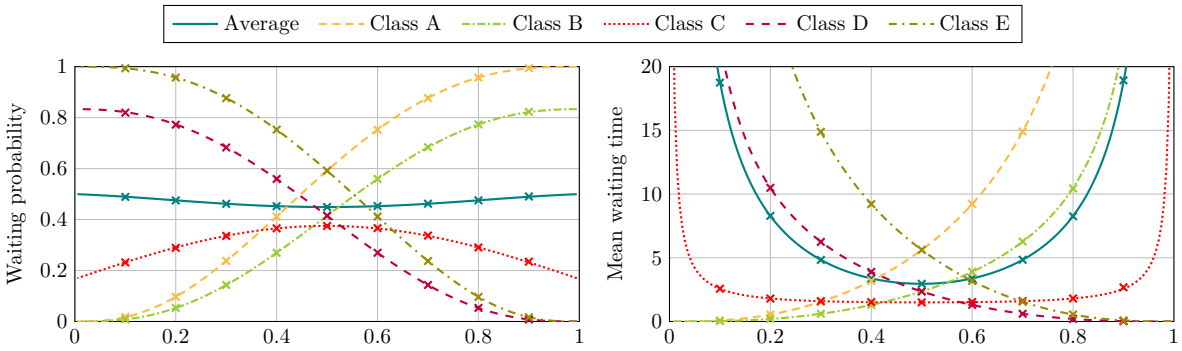
To illustrate our results, we apply the formulas of Section 3 to the toy example of Figure 1a. The arrival probabilities are chosen as follows: for any $\rho \in (0,1)$,

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{4}, \qquad \mu_A = \frac{\rho}{4}, \qquad \mu_B = \mu_C = \mu_D = \frac{1}{4}, \qquad \mu_E = \frac{1-\rho}{4}. \tag{12}$$

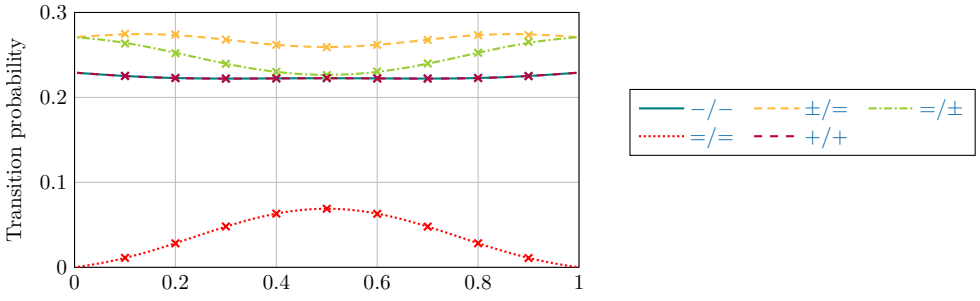
Figure 3 shows several performance metrics. The lines are plotted using the results of Section 3. To verify these results, we plotted marks representing simulated values based on averaging the results of 20 discrete-event simulation runs, each consisting of 10^6 transitions after a warm-up period of 10^6 transitions. The



(a) Customer-oriented performance metrics



(b) Server-oriented performance metrics



(c) Transition probabilities

Figure 3: Numerical results associated with the graph of Figure 1a. The abscissa is the parameter ρ defined in (12).

standard deviation of the simulated waiting times (resp. probabilities) never exceeded 1.9 (resp. 0.008) per experiment, validating the reliability of the results.

Due to the parameter settings, performance is symmetrical around $\rho = \frac{1}{2}$. Figure 3a and 3b show that classes 1, 2, 3, C, D, and E become unstable, in the sense that their mean waiting time tends to infinity, as $\rho \downarrow 0$. This is confirmed by observing that $\Delta(\mathcal{A}) \downarrow 0$ for $\mathcal{A} \in \{\{4, A\}, \{4, A, B\}, \{4, A, B, C\}, \{3, 4, A\}, \{3, 4, A, B\}, \{2, 3, 4, A\}\}$ when $\rho \downarrow 0$. We conjecture that this limiting regime can be studied by adapting the heavy-traffic analysis of [10, Section 6.2], although the behavior is different due to the concurrent arrivals of customers and servers.

Even if classes 1, 2, 3, C, D, and E all become unstable as $\rho \downarrow 0$, we can distinguish two qualitatively-different behaviors: the waiting probabilities of classes 1 and E tend to one, while for classes 2, 3, C, and D the limit is strictly less than one. This difference lies in the fact that the former classes have degree one in the compatibility graph, while the latter have degree two. Especially class C is intriguing, as the monotonicity of its waiting probability and mean waiting time are reversed, and would be worth further investigation.

Figure 3c shows that the probabilities of transitions $-/-$ and $+/+$ are equal to each other (as announced by Corollary 1) and are approximately constant. The probabilities of transitions $\pm/=$, $=/\pm$, and $=/=$, which impact the imbalance between classes but not the total queue lengths, vary with ρ. In particular, the probability of transitions $=/=$ is maximal when $\rho = \frac{1}{2}$, which may explain why $\rho = \frac{1}{2}$ minimizes the average waiting probability and mean waiting time.

References

[1] Ivo Adan, Ana Busic, Jean Mairesse, and Gideon Weiss. Reversibility and further properties of FCFS infinite bipartite matching. *Math. Oper. Res.*, 43(2):598–621, 2017.

[2] Ivo Adan, Igor Kleiner, Rhonda Righter, and Gideon Weiss. FCFS parallel service systems and matching models. *Perform. Evaluation*, 127-128:253–272, 2018.

[3] Ivo Adan and Gideon Weiss. Exact FCFS matching rates for two infinite multitype sequences. *Oper. Res.*, 60(2):475–489, 2012.

[4] Ivo Adan and Gideon Weiss. A skill based parallel service system under FCFS-ALIS — steady state, overloads, and abandonments. *Stoch. Syst.*, 4(1):250–299, 2014.

[5] S.A. Berezner and A. E. Krzesinski. Order independent loss queues. *Queueing Syst.*, 23(1):331–335, 1996.

[6] Thomas Bonald, Céline Comte, and Fabien Mathieu. Performance of balanced fairness in resource pools: A recursive approach. *Proceedings of the ACM on Measurement and Analysis of Comput. Syst.*, 1(2):41:1–41:25, 2017.

[7] Ana Busic, Varun Gupta, and Jean Mairesse. Stability of the bipartite matching model. *Adv. in Appl. Probab.*, 45(2):351–378, 2013.

[8] René Caldentey, Edward H. Kaplan, and Gideon Weiss. FCFS infinite bipartite matching of servers and customers. *Adv. in Appl. Probab.*, 41(3):695–730, 2009.

[9] Céline Comte. Resource management in computer clusters: Algorithm design and performance analysis. *Ph.D. thesis. Institut Polytechnique de Paris.*, 2019.

[10] Céline Comte. Stochastic non-bipartite matching models and order-independent loss queues. 2021. To appear in Stoch. Models.

[11] Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of Weighted Automata*. Monographs in Theoretical Computer Science. An EATCS Series. Springer-Verlag, 2009.

[12] Kristen Gardner and Rhonda Righter. Product forms for FCFS queueing models with arbitrary server-job compatibilities: an overview. *Queueing Syst.*, 96(1):3–51, 2020.

[13] Kristen Gardner, Samuel Zbarsky, Sherwin Doroudi, Mor Harchol-Balter, and Esa Hyytia. Reducing latency via redundant requests: Exact analysis. *ACM SIGMETRICS Perform. Evaluation Review*, 43(1):347–360, 2015.

[14] A. E. Krzesinski. Order independent queues. In Richard J. Boucherie and Nico M. Van Dijk, editors, *Queueing networks: A fundamental approach*, number 154 in Internat. Ser. in Ope. Res. & Manag. Sci., pages 85–120. Springer US, 2011.

[15] Jean Mairesse and Pascal Moyal. Stability of the stochastic matching model. *J. Appl. Probab.*, 53(4):1064–1077, 2016.

[16] Pascal Moyal, Ana Busic, and Jean Mairesse. A product form for the general stochastic matching model. *J. Appl. Probab.*, 58(2):449–468, 2021.

[17] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.*, 6(3):505–517, 1977.

[18] Yufei Zhao. The number of independent sets in a regular graph. *Combinatorics, Probability and Computing*, 19(2):315–320, 2010.

Appendix: Proofs of the results of Section 3

Proof of Corollary 1. Summing (8) over all $\mathcal{A} \in \mathbb{I}$ and rearranging the sum symbols yields

$$\sum_{(i,k)\in\mathcal{I}\times\mathcal{K}} \lambda_i\mu_k \sum_{\substack{\mathcal{A}\in\mathbb{I}:i\in\mathcal{I}(\mathcal{A}\cap\mathcal{K}),\\ k\in\mathcal{K}(\mathcal{A}\cap\mathcal{I})}} \pi(\mathcal{A}) = \sum_{\substack{(i,k)\in\mathcal{I}\times\mathcal{K}:\\ i\nsim k}} \lambda_i\mu_k \sum_{\substack{\mathcal{A}\in\mathbb{I}:\\ i\in\mathcal{A},k\in\mathcal{A}}} \big(\pi(\mathcal{A}) + \pi(\mathcal{A}\backslash\{i\}) + \pi(\mathcal{A}\backslash\{k\}) + \pi(\mathcal{A}\backslash\{i,k\})\big). \tag{13}$$

The left-hand side of this equation is the left-hand side of (9). The right-hand side can be rewritten by making changes of variables. For instance, for each $i \in \mathcal{I}$ and $k \in \mathcal{K}$ such that $i \nsim k$, replacing $\mathcal{A}$ with $\mathcal{A}\backslash\{i,k\}$ in the last sum yields

$$\sum_{\mathcal{A}\in\mathbb{I}:i\in\mathcal{A},k\in\mathcal{A}} \pi(\mathcal{A}\backslash\{i,k\}) = \sum_{\substack{\mathcal{A}\subseteq\mathcal{I}\cup\mathcal{K}:i\notin\mathcal{A},k\notin\mathcal{A},\\ \mathcal{A}\cup\{i,k\}\in\mathbb{I}}} \pi(\mathcal{A}) = \sum_{\substack{\mathcal{A}\in\mathbb{I}_0:i\notin\mathcal{A},k\notin\mathcal{A},\\ i\notin\mathcal{I}(\mathcal{A}\cap\mathcal{K}),k\notin\mathcal{K}(\mathcal{A}\cap\mathcal{I})}} \pi(\mathcal{A}).$$

The second equality is true only because $i \nsim k$. By applying changes of variables to the other terms, we obtain that the right-hand side of (13) is equal to

$$\begin{aligned}
&\sum_{\substack{(i,k)\in\mathcal{I}\times\mathcal{K}:\\ i\nsim k}} \lambda_i\mu_k \Bigg(\sum_{\substack{\mathcal{A}\in\mathbb{I}_0:i\in\mathcal{I},k\in\mathcal{A},\\ i\notin\mathcal{I}(\mathcal{A}\cap\mathcal{K}),k\notin\mathcal{K}(\mathcal{A}\cap\mathcal{I})}} \pi(\mathcal{A}) + \sum_{\substack{\mathcal{A}\in\mathbb{I}_0:i\notin\mathcal{I},k\in\mathcal{A},\\ i\notin\mathcal{I}(\mathcal{A}\cap\mathcal{K}),k\notin\mathcal{K}(\mathcal{A}\cap\mathcal{I})}} \pi(\mathcal{A}) \\
&\qquad + \sum_{\substack{\mathcal{A}\in\mathbb{I}_0:i\in\mathcal{I},k\notin\mathcal{A},\\ i\notin\mathcal{I}(\mathcal{A}\cap\mathcal{K}),k\notin\mathcal{K}(\mathcal{A}\cap\mathcal{I})}} \pi(\mathcal{A}) + \sum_{\substack{\mathcal{A}\in\mathbb{I}_0:i\notin\mathcal{A},k\notin\mathcal{A},\\ i\notin\mathcal{I}(\mathcal{A}\cap\mathcal{K}),k\notin\mathcal{K}(\mathcal{A}\cap\mathcal{I})}} \pi(\mathcal{A}) \Bigg) \\
&= \sum_{(i,k)\in\mathcal{I}\times\mathcal{K}:i\nsim k} \lambda_i\mu_k \sum_{\mathcal{A}\in\mathbb{I}_0:i\notin\mathcal{I}(\mathcal{A}\cap\mathcal{K}),k\notin\mathcal{K}(\mathcal{A}\cap\mathcal{I})} \pi(\mathcal{A}).
\end{aligned}$$

□

Proof of Proposition 2. Let $i \in \mathcal{I}$. We have $L_i = \sum_{\mathcal{A}\in\mathbb{I}_0} \ell_i(\mathcal{A})$, where

$$\ell_i(\mathcal{A}) = \sum_{(c,d)\in\Pi_{\mathcal{A}}} |c|_i \pi(c,d), \quad \mathcal{A} \in \mathbb{I}_0.$$

Let $\mathcal{A} \in \mathbb{I}$. If $i \notin \mathcal{A}$, we have directly $\ell_i(\mathcal{A}) = 0$ because $|c|_i = 0$ for each $(c,d) \in \Pi_{\mathcal{A}}$. Now assume that $i \in \mathcal{A}$ (so that in particular $\mathcal{A}$ is non-empty). The method is similar to the proof of Proposition 1. First, by applying (1), we have

$$\ell_i(\mathcal{A}) = \sum_{(c,d)\in\Pi_{\mathcal{A}}} |c|_i \frac{\lambda_{c_n}}{\mu(\mathcal{K}(\mathcal{A}\cap\mathcal{I}))} \frac{\mu_{d_n}}{\lambda(\mathcal{I}(\mathcal{A}\cap\mathcal{K}))} \pi((c_1,\dots,c_{n-1}),(d_1,\dots,d_{n-1})).$$

Then applying (6) and doing a change of variable yields

$$\begin{aligned}
&\mu(\mathcal{K}(\mathcal{A}\cap\mathcal{I}))\lambda(\mathcal{I}(\mathcal{A}\cap\mathcal{K}))\ell_i(\mathcal{A}) \\
&= \lambda_i \sum_{k\in\mathcal{A}\cap\mathcal{K}} \mu_k \Bigg(\sum_{(c,d)\in\Pi_{\mathcal{A}}} (|c|_i+1)\pi(c,d) + \sum_{(c,d)\in\Pi_{\mathcal{A}\setminus\{i\}}} (0+1)\pi(c,d) \\
&\qquad\qquad + \sum_{(c,d)\in\Pi_{\mathcal{A}\setminus\{k\}}} (|c|_i+1)\pi(c,d) + \sum_{(c,d)\in\Pi_{\mathcal{A}\setminus\{i,k\}}} (0+1)\pi(c,d) \Bigg), \\
&\quad + \sum_{j\in(\mathcal{A}\setminus\{i\})\cap\mathcal{I}} \sum_{k\in\mathcal{A}\cap\mathcal{K}} \lambda_j\mu_k \Bigg(\sum_{(c,d)\in\Pi_{\mathcal{A}}} |c|_i\pi(c,d) + \sum_{(c,d)\in\Pi_{\mathcal{A}\setminus\{j\}}} |c|_i\pi(c,d) \\
&\qquad\qquad + \sum_{(c,d)\in\Pi_{\mathcal{A}\setminus\{k\}}} |c|_i\pi(c,d) + \sum_{(c,d)\in\Pi_{\mathcal{A}\setminus\{j,k\}}} |c|_i\pi(c,d) \Bigg), \\
&= \lambda_i \sum_{k\in\mathcal{A}\cap\mathcal{K}} \mu_k \left(\ell_i(\mathcal{A}) + \pi(\mathcal{A}) + \pi(\mathcal{A}\setminus\{i\}) + \ell_i(\mathcal{A}\setminus\{k\}) + \pi(\mathcal{A}\setminus\{k\}) + \pi(\mathcal{A}\setminus\{i,k\})\right) \\
&\quad + \sum_{j\in(\mathcal{A}\setminus\{i\})\cap\mathcal{I}} \sum_{k\in\mathcal{A}\cap\mathcal{K}} \lambda_j\mu_k \left(\ell_i(\mathcal{A}) + \ell_i(\mathcal{A}\setminus\{j\}) + \ell_i(\mathcal{A}\setminus\{k\}) + \ell_i(\mathcal{A}\setminus\{j,k\})\right).
\end{aligned}$$

The result follows by rearranging the terms. □

Proof of Proposition 3. Equation (11) follows by summing (10) over all $i \in \mathcal{I}\cap\mathcal{A}$ and simplifying the result using (8). □