Stockholm School of Economics
Department of Economics
5350 Masters Thesis in Economics
Academic Year 2018–2019

Good Things Come to Those Who Wait

An analysis of student housing in Stockholm as a dynamic matching problem with an overloaded waiting list

Moritz von Schwartzenberg (41219)

Abstract

Motivated by the high demand for student housing in Stockholm, this thesis analyzes the allocation of student housing in Stockholm in light of recent developments in dynamic matching. Taking Leshno (2017) as a basis, we set up a model with an overloaded waiting list, a concept introduced by Leshno to describe waiting lists of sufficient length, such that arriving items can always be assigned. We show that the student housing allocation following the rules by Stiftelsen Stockholms Studentbostäder (SSSB), Stockholms biggest student housing provider, is outcome-equivalent to a First Come First Served (FCFS) mechanism. Subsequently, we analyze Leshno's Service in Random Order (SIRO) mechanism, a robust mechanism, which reduces the welfare loss in comparison to the FCFS mechanism by incentivizing agents to decline non-favorite items. We then compare the FCFS and the SIRO mechanism for various parameter settings and conclude that SIRO performs weakly better than FCFS for all settings considered. When extending the model to incorporate n-items as well as heterogeneous agents and allowing for an imbalance in supplied and demanded item types, we find that SIRO still outperforms FCFS in settings where agents are patient, but that benefits disappear for very impatient agents. Eventually, we conclude that the SIRO mechanism has the potential to increase the welfare in the allocation of student housing in Stockholm.

Keywords: Dynamic matching, overloaded waiting list, housing allocation, queuing, robust mechanism
JEL: C78, D47, D71, R31

Supervisor:	Albin Erlanson
Date submitted:	May 13, 2019
Date examined:	May 28, 2019
Discussant:	Anita Hafner
Examiner:	Mark Sanctuary

Acknowledgments

I would like to thank my supervisor Albin Erlanson for his help in finding the topic of this thesis, his constructive feedback and the valuable discussions.

Further, I want to express my gratitude for Thorben Frank's and Alireza Mirzaei's helpful feedback and comments.

I would like to thank Victoria Svensson for always having my back, sharing laughters and memories.

Lastly, I want to thank my parents for their unconditional love and support.

Contents

List of Figures

List of Tables

List of Abbreviations

FCFS	First Come First Served
IC	Incentive compatible
SIRO	Service in Random Order
SSCO	Stockholms Studentkårers Centralorganisation
SSSB	Stiftelsen Stockholms Studentbostäder
WFL	Welfare loss from misallocation

1 Introduction

In the first subsection of the introduction we will motivate our thesis and give an overview of our results. Subsequently, we will elaborate on student housing in Stockholm and conclude the introduction with our literature review.

1.1 Motivation and overview

Stockholm, Sweden's capital and biggest city, with its numerous educational institutions attracts many students from within Sweden as well as internationally. While degrees, backgrounds and personal situations differ, many students are unified in their need for affordable housing during the time of their studies. Budget constraints or staying in Stockholm only for a limited time, make renting instead of buying an apartment a common choice among students.

Apartments in Stockholm, however, are hard to find – and if one is restricted by a student's budget, even more so. Student housing, therefore, seems like a promising solution. The biggest student housing provider in Stockholm, Stiftelsen Stockholms Studentbostäder (SSSB), owns and rents out about 8,000 apartments. However, their waiting list comprises roughly 6 times as many students (SSSB, 2018).

This observation led to the idea of analyzing the allocation of student housing in Stockholm in the setting of an "overloaded waiting list", a concept introduced in Leshno (2017). The concept refers to a waiting list which is so long that, even when agents are allowed to reject items, the end of the list is never reached when offering available objects to the agents in the queue. In other words, no item will go to waste because there is always someone, who would accept it. Leshno points out two main benefits from examining an allocation problem as an overloaded waiting list: first, we can abstract from the arrival process of agents; second, we do not have to take waiting costs into account when comparing the performance of different allocation mechanisms. As every item will get assigned, independent of who the item is assigned to, there will be one less agent waiting. The performance of different mechanisms then simplifies to comparing how much welfare is generated (or lost) through the mechanism's allocation.

For an optimal allocation, each agent should receive her favorite item. However, as waiting is costly, it might be rational for some agents to accept an available non-favorite item now, instead of waiting to receive their favorite item at a later point in time. When agents receive a non-favorite item, the overall welfare decreases, as there would exist another agent on the overloaded waiting list who would have valued the item more. Leshno (2017) therefore states that, in the interest of a low welfare loss, it is desirable that agents decline non-favorite items. Incentivizing agents to decline non-favorite items is therefore one of the crucial aspects in our analysis.

Taking Leshno (2017) as a basis, we set up a 2-item model in which agents are on an overloaded waiting list to receive items. We then implement an allocation mechanism following the rules of SSSB. While the rules seem complex at first, we will show that they result in the same outcome as a First Come First Served (FCFS) mechanism, a mechanism that allocates items to agents in order of their arrival to the waiting list. This allows us to study the outcome of the FCFS mechanism and make inferences about SSSB's allocation.

For ease of analysis, we will then demonstrate that the FCFS mechanism can be depicted as a mechanism that holds a separate priority-queue, a buffer-queue, for both item types. That means agents on the general waiting list have the possibility to decline

an offered item and join the buffer-queue for the other item. Should then an item of this other kind arrive it will first get offered to the agents in the buffer-queue.

Building on this, we will elaborate on Leshno's Service in Random Order (SIRO) mechanism, which uses a similar setup. While under FCFS agents in the buffer-queue get served in order of their arrival, under SIRO the item is allocated to an agent at random. While an agent would not join a buffer-queue to be the last in line, the agent is more likely to do so if the chances of receiving the item are the same for all agents. SIRO, thus, offers the agents a higher expected value for later positions in the buffer-queue and, consequently, incentivizes them to decline non-favorite items. We additionally note that it is possible to include a simple learning scheme in the mechanism, which makes the SIRO mechanism independent of the model parameters and the mechanism designer's beliefs (we say the mechanism is "robust").

Having established the outcome-equivalence of FCFS and SSSB's mechanism as well as having introduced the SIRO mechanism, we then turn to a comparison and sensitivity analysis of FCFS and SIRO. We see that SIRO performs better than FCFS for nearly all parameter settings.

To see if this holds true in more realistic settings, we extend the model to incorporate n-item types, introduce heterogeneous mismatch values and allow for an imbalanced economy (i.e. long-term supply of and demand for item types differ). We elaborate on the implications for our model, such as strategic queuing for non-favorite items, and investigate the underlying reasons for the behavior we observe. We come to the conclusion that SIRO still has a lower welfare loss from misallocation than FCFS for settings with patient agents, whereas for impatient agents these benefits disappear.

After our analytical sections, we asses SSSB's allocation rules from a practical perspective and hypothesize about their mechanism design choices. Additionally, we look more closely at two of our assumptions, the overloaded waiting list and the possibility of having differentiating characteristics for the SIRO mechanism. We argue that SSSB's housing queue can be considered an overloaded waiting list and provide background and a suggestion for potential differentiating characteristics to be used with the SIRO mechanism. Furthermore, we acknowledge the importance of fairness considerations in the student housing allocation and present different perspectives.

In summary, this thesis shows that by regarding the waiting list for student housing from SSSB as an overloaded waiting list, we can establish that SSSB's allocation is outcome-equivalent to using an FCFS mechanism. This allocation, in turn, can be improved by using a robust mechanism (SIRO), which demonstrates promising results even when we extend the model to more complex settings.

The remainder of the thesis is structured as follows: We conclude the introductory section with a subsection on SSSB's rules and procedures for the housing allocation (1.2) followed by a literature review (1.3). In section 2 we first set up our model and preliminaries in subsection 2.1. Then in subsection 2.2 we introduce SSSB's mechanism and show that it is outcome-equivalent to an FCFS mechanism. After elaborating on the FCFS mechanism, we introduce Leshno's (2017) SIRO mechanism in subsection 2.3. In section 3 we conduct a comparison between the FCFS and SIRO mechanism as well as a sensitivity analysis using the theoretical misallocation rate (3.1) and the amount of agents who would decline a non-favorite item (3.2). To make our model more realistic, we subsequently implement extensions in section 4 and evaluate the mechanisms' performances. Afterwards we turn to our discussion of relevant issues, such as practical implications and fairness considerations (section 5), before concluding our thesis with section 6.

1.2 Student housing via SSSB

With 7985 student homes, SSSB provides a large share of the student housing in Stockholm (SSSB, 2018). Founded in 1958 by Stockholms Studentkårers Centralorganisation (SSCO), the federation of student unions in Stockholm, SSSB's goal is to provide students in Stockholm with affordable housing (SSSB, n.d.). We will in this section explain the most important parts of the rules and procedure for the allocation of student housing. The comprehensive rules are published as SSSB (2012) and are available for download on SSSB's website.

To receive student housing, a student has to join SSSB's housing queue. In order to be eligible to sign up to SSSB's housing queue, a student has to be a member of a student union, which is affiliated with SSCO, and additionally has to be between 16 and 54 years old.[1] Currently around 40 student unions are affiliated with SSCO, which together represent ca. 85,000 students (SSCO, n.d.).

Once a student has signed up for SSSB's housing queue, she starts to collect so-called "credit days". Per day spent queuing, one credit day is accumulated by the student. With these credit days, a student can then apply online to available apartments, which are posted on SSSB's website together with information such as monthly rent, location, size and number of rooms. After a few days the application period for an apartment ends and the student with the most credit days among the applicants will be offered to become the tenant of the apartment. Should the student then decline the apartment, it will get posted on SSSB's website again. SSSB penalizes declining an apartment that has been offered to a student by implementing that the student looses her credit days if she rejects an offered apartment twice. Students also loose their credit days once they sign a tenancy agreement via SSSB.

During the application period, every student can see the amount of credit days that the highest applicant for an apartment has accumulated. Also students are only allowed to apply to a maximum of three apartments at the same time. Should a student be the applicant with the highest points for more than one apartment, he will randomly receive an offer for one of them. However, once a student has applied to an apartment, it is also possible to withdraw again. Neither applying, nor withdrawing is costly.

In order to keep an apartment via SSSB, the student has to remain a member of a student union which is affiliated with SSCO and must be studying at least half-time. These requirements are regularly checked by SSSB and should a student not fulfill the requirements anymore, the tenancy agreement is terminated.

Aside from queuing for housing to receive an tenancy agreement, SSSB implemented the possibility for students to apply for priority treatment, which entails the offer of a tenancy agreement independent of a student's credit days. A committee, the "Bostadsdelegation", can grant priority treatment in cases where the housing has to fulfill certain requirements, location is of utmost importance or receiving a tenancy agreement is urgent. Additionally, medical reasons, severe social reasons, disability or other circumstances that are deemed relevant by the housing delegation, will allow a student to receive priority treatment. SSSB explicitly states, however, that too many other applicants or homelessness are no sufficient reasons for priority treatment.

[1] To actually become a tenant, however, a student has to be at least 18 years old. Additionally, there is the possibility of queuing already before being affiliated with one of the student unions for a maximum of 90 days.

1.3 Literature review

We will in this subsection give an overview of literature related to our analysis. Before reviewing the literature, however, it is helpful to clarify the semantics we are going to use. We follow Budish (2012), who distinguishes between matching and mechanism design. The author elaborates that in mechanism design the focus lies on maximizing some objective, for example welfare, with a mechanism. In matching, on the other hand, the focus lies on finding a mechanism that fulfills certain properties like fairness.

Put simply, mechanism design is more theoretical, while matching usually has an applied focus. However, the lines between both can be blurred. For example in the paper by Leshno (2017), which our analysis is built on, the goal is to minimize welfare loss and the author derives the optimal mechanism in the model settings. Nevertheless, the author also acknowledges the difficulties of implementing the optimal mechanism in practice and derives a more easily implementable alternative. Despite the sometimes blurred lines, the distinction between matching and mechanism design helps us to identify two strands of literature and understand their origins.

The origins of the literature on mechanism design are commonly attributed to Hurwicz, who in Hurwicz (1960) stated a broad definition of a mechanism and in Hurwicz (1972) defined incentive compatibility. The concept of incentive compatibility describes the characteristic of a mechanism under which agents fare best by acting truthfully. Both concepts paved the way for subsequent work in the field of mechanism design. For a detailed elaboration on the developments in the field of mechanism design, the reader is referred to the summaries of Maskin and Sjöström (2002) or Baliga and Maskin (2003).

Matching on the other hand has its origins in the work by Gale and Shapley (1962), who introduced an algorithm (deferred-acceptance) for college admissions. Initially the paper did not receive much attention, but, through Roth (1984), gained popularity (Sönmez and Ünver, 2011). Roth showed that the deferred-acceptance had been used in practice since 1952 to match medical interns with hospitals. Today, the canonical areas of application within matching include school choice (e.g. Abdulkadiroğlu and Sönmez, 2003), kidney exchange (e.g. Roth, Sönmez & Ünver, 2004) and housing allocation (e.g. Hylland and Zeckhauser, 1979). For an overview of matching in these settings, the reader is referred to the survey on matching theory by Sönmez and Ünver (2011).

While much of the matching literature is focused on a static setting, the paper by Leshno (2017) can be attributed to the field of dynamic matching. This field, which is concerned with allocations of items over multiple periods, has in recent years received more attention. This is likely due to the advancements made with regards to computational power, as many of the papers in dynamic matching use simulations. Leshno's paper introduces the concept of an overloaded waiting list, which is a novelty. Yet, there exist related papers and strands of literature, which we will introduce in the following.

The two papers, which are most closely related to Leshno (2017) are Thakral (2016) and Bloch and Cantala (2017). Thakral studies the assignment of housing, incorporates an FCFS buffer-queue and quantifies the benefit through an empirical analysis for the housing allocation in Pittsburgh. However, the author's approach is more focused on fulfilling axioms and incorporates uncertainty about the arrival of housing. Bloch and Cantala study the dynamical assignment of objects to agents in a queue, which is of constant size. Their main results are that all agents prefer FCFS over a lottery in the case of private values, but that waste is smaller if a lottery is used. Apart from not implementing the overloaded waiting list setting, their paper also differs from Leshno's (2017) as the authors consider agents whose preferences can change over time.

Additionally, the work by Baccara, Lee and Yariv (2018) is closely related to Leshno (2017) in its use of a Markov chain to describe the different states of the system. However, the authors consider a two-sided market, whereas Leshno sets up a model in a one-sided market.[2] The authors conclude that the optimal mechanism directly conducts optimal matches. If only suboptimal matches are possible, however, the mechanism accumulates agents until a threshold is reached and then conducts the matching.

This principle of accumulating agents which do not form perfect matches, commonly referred to as batching, increases the thickness of the market, i.e. the overall supply in the market. Examples for papers that investigate thickness in a market come to different conclusions about batching dependent on the exact settings. Ünver (2010), who models a dynamic kidney exchange market, has the aim to minimize agents' waiting times. He concludes that in his model efficient mechanisms conduct an exchange of kidneys as soon as it becomes possible. On the other hand, Akbarpour, Shengwu and Shayan (2014) set up a model in which agents stochastically depart from a waiting list and reach the conclusion that a thicker market improves welfare. Leshno (2017) does not investigate mechanisms that implement batching. He argues that batching would increase the overall waiting cost, which for high waiting cost might outweigh the benefit of batching, whereas for low waiting costs agents might still prefer to wait for their favorite item.

Additionally, there exists a literature for queuing models, which Hassin and Haviv (2003) examine in an extensive survey. Especially chapter two of their book that covers aspects such as random queues relate to Leshno (2017).

The allocation of public or student housing, has received much attention in the matching literature. One of the most notable contributions was made by Shapley and Scarf (1974), who introduced the concept of the top trading cycle algorithm (though attributed to David Gale). This algorithm creates a stable matching in a setting where agents are endowed with a houses.[3] Assignments that specifically use a waiting list have been the focus of Kaplan (1987), who even includes a dynamic aspect in his model and proposes a mechanism that conducts matchings based on which agent is most in need of an apartment at the time an apartment becomes available. However, he refrains from considering the strategical considerations on the side of the agents applying.

Lastly, the service in random order (SIRO) mechanism, which we will examine in later sections, has been introduced by Leshno (2017) with the aim to have a more robust alternative to the optimal mechanism in his model. Robustness is a term that is used to describe the property of a mechanism to fully or in parts be independent of model parameters. The SIRO mechanism is fully independent of model parameters, a property referred to as "detail-free" (p. 347) by Dasgupta and Maskin (2000). With regards to robust mechanism design Bergemann, Brooks and Morris (2016) remark that while the assumption of knowing all parameters is in most cases false, knowing nothing about the parameters is also unlikely. The authors conjecture that a compromise between both extremes will result in reasonable models and offer possibilities of future research. Nevertheless it is desirable to derive a detail-free mechanism for a problem if possible. Its robustness makes Leshno's SIRO mechanism especially appealing for application in a real-world setting.

[2] In two-sided markets, there are agents on both sides that have preferences over matches (e.g. employer and worker), whereas a one-sided market exists if each agent can be matched with any other agent or agents are matched with objects (e.g. students and apartments) (Niederle, Roth & Sönmez, 2008).

[3] A stable matching in this setting describes a matching in which agents cannot improve through exchange with another agent, who is willing to participate in an exchange.

2 Model and allocation mechanisms

In this section, we will explain the model as well as the objective when conducting the allocation and then introduce different mechanisms to solve the allocation problem. Throughout, we will use notation similar to Leshno (2017) and build on his model.

2.1 Model setup and preliminaries

The aim of this section is to specify the underlying model, clarify properties as well as definitions and introduce an example, which we will use to illustrate concepts throughout the coming sections.

2.1.1 Model setup

We consider a set of rational, risk-neutral agents $Q = \{1, 2, ...\}$ and a finite pool of indivisible items $H = \{h_1, h_2, ..., h_n\}$. Items are either of type A or type B. Agents are with probability p_α of type α and prefer A items, or otherwise with probability $p_\beta = 1 - p_\alpha$ of type β and prefer B items.

Some of the agents are in a queue (general waiting list) to receive an item. For each time period $t = 0, 1, 2, ...$ spent in the queue, each agent incurs the same additive waiting costs $c > 0$. In each time period one item will become available to be assigned to the agents. The probability that an A item becomes available is p_A and correspondingly $p_B = 1 - p_A$ for a B item. We will for most parts of our analysis restrict attention to a balanced economy, meaning that the probability of an item being of a certain type is the same as the percentage of agents in the population that prefer such items (i.e. $p_A = p_\alpha$ and $p_B = p_\beta$).

We denote the value that agents derive from receiving an item as

$$v = \begin{cases} 1 & \text{if item } h_k \text{ is of the preferred type} \\ x & \text{otherwise,} \end{cases} \tag{1}$$

where $x < 1$. We denote an agent's value from item h_k as v_k and in case it is not of the preferred type as x_k.

Additionally, let $Q_t \subseteq Q$ denote the subset of agents queuing in period t and let then $|Q_t|$ denote number of agents in the queue at period t. At any point in time, we assume that $|Q_t| \geq M$, where $M \gg 0$ such that an available item can always be assigned to an agent on the list, even if agents are allowed to reject items. This is referred to by Leshno (2017) as an "overloaded waiting list".

2.1.2 Objective

We have now specified the underlying model for our analysis in which agents are waiting in a queue to receive an item. We would like to match the agents with the items and as the items arrive randomly over time, we talk about dynamic matching. A matching mechanism defines a set of rules according to which the agents get matched with the items.

Definition 1. A mechanism is an injective function $\mu_t(h_k) \mapsto i$ with $i \in Q_t$ that assigns an available item h_k to a queuing agent i in period t, i.e. the agent i and the item h_k are matched.

Definition 2. If an agent is matched with an item which is not of the agent's favorite type, the item is misallocated (or mismatched).

As mechanisms can be specified in different ways, we would like to be able to make comparisons. Therefore, we now define welfare loss, which will serve as the measure that we use to evaluate a mechanisms performance. Note that we do not have to take into account waiting costs, but only the welfare loss from misallocation (WFL). This is because in an overloaded waiting list, independent of which agent gets assigned to an item, the overall waiting costs are not affected. In any case there will be one less agent waiting.

Definition 3. For a mechanism μ, we define the long-run welfare loss from misallocation as

$$\text{WFL}_\mu = (1-x)\xi \tag{2}$$

$$\text{with } \xi = \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \xi_t \text{ and } \xi_t = \begin{cases} 1 \text{ if } \mu_t(h_k) = i \text{ is a mismatch} \\ 0 \text{ otherwise,} \end{cases}$$

where x is the mismatch-value from equation (1). Note that ξ is the long-run misallocation rate and ξ_t an indicator which equals 1 if the item's allocation through mechanism μ results in a mismatch and 0 otherwise.

The lower the welfare loss induced by a mechanism μ, the better. The objective in our analysis is to match the items and agents while minimizing the WFL.

2.1.3 Agent's decision

As the mechanisms that we will consider offer items to agents, i.e. allow the agents to decline an offered item, we will now look more closely at the agent's decision. If an agent is offered an item, she faces the decision to either accept the item or to wait instead.

Remark 1. *If the available item is of an agent's preferred type, the agent will always prefer the available item over waiting for another item.*

Remark 1 is an observation that directly follows from equation (1) in connection with the waiting costs c. As the agent will derive the highest possible value from an assignment to an item of her preferred type, waiting will be strictly worse due to the waiting costs, even if another item of her favorite kind became available and she was sure to get assigned to it.

Lemma 1. *If offered a mismatched item, an agent will prefer waiting (i.e. decline the offered item) if the expected waiting time for her favorite item does not exceed the maximum amount of periods that the agent is willing to wait for her favorite item.*

Proof. To proof Lemma 1, we will show that comparing the value of accepting with the expected value of rejecting is equivalent to comparing the expected waiting time for a favorite item with the maximum amount of periods that the agent is willing to wait for the favorite item type.

If an agent i is offered a non-favorite item h_k, she will reject that item only if the expected value from waiting for the favorite item is greater than or equal to the value from receiving the non-favorite item (x_k) now. Let another item $h_j \neq h_k$ be an item of the agent's favorite type and let t_x denote the periods until agent i is matched with item

Agent	Type	Item value		$\bar{w}$
		v_A	v_B	
1	β	0.4	1	6
2	β	0.4	1	6
3	β	0.4	1	6
4	β	0.4	1	6
5	α	1	0.4	6
6	α	1	0.4	6
7	α	1	0.4	6
8	β	0.4	1	6
9	α	1	0.4	6
10	α	1	0.4	6

Table 1: Baseline example: First 10 agents on waiting list. The maximum amount of periods that an agent is willing to wait $\bar{w} = \frac{1-x}{c}$ is calculated with $c = 0.1$ and the mismatch value $x = 0.4$.

h_j. With $E[t_x]$ denoting the expectation of t_x, we can establish that an agent will reject a mismatched item if it holds that

$$x_k \leq 1 - E[t_x] \cdot c \tag{3}$$

$$E[t_x] \leq \frac{1 - x_k}{c}$$

$$E[t_x] \leq \bar{w}, \tag{4}$$

where $\bar{w} = \frac{1-v_k}{c}$ is the maximum amount of periods that an agent is willing to wait for her favorite item, which we will from now on refer to as the agent's willingness to wait. We proved Lemma 1 by showing that equation (3) and equation (4) are equivalent. □

2.1.4 Introduction of a baseline example

Having set up our model and clarified the objective as well as the agent's decision, we now introduce a baseline example, which we will refer to in the coming sections when using examples for clarification.

We assume that we are in a setting with an overloaded waiting list, meaning that there are always at least as many agents in the queue, so that an available item can be assigned. Table 1 shows the first 10 agents on the waiting list, their types (α or β), the values they derive from the different item types (v_A and v_B) depending on their types and the maximum amount of periods that they would be willing to wait for their favorite item if offered a mismatched item ($\bar{w}$), i.e. their willingness to wait.

The arrival probabilities of the two item types are $p_A = 0.5$ and $p_B = 0.5$, the value of a non-favorite item, i.e. the mismatch value, is $x = 0.4$ and the waiting costs are $c = 0.1$. Therefore, the agents' willingness to wait is $\bar{w} = \frac{1-x}{c} = 6$. We will assume these values as well as the waiting list (Table 1) for most of our subsequent examples.

2.1.5 Applying a random allocation mechanism to the baseline example

We will now illustrate the model and concepts introduced in this section with an example of an allocation through a mechanism. As a basis we will assume the parameters and agents introduced in the baseline example in section 2.1.4.

Consider a mechanism that offers the available item in every period to one queuing agent that is randomly selected among the first 10 agents in the queue, who have not yet rejected the item. The selected agent then has the option to accept or decline the item. Should the agent accept the item, she will take the item and leave the waiting list. Should she decline the item, she will remain on the waiting list and the item will get offered to another agent that again gets randomly chosen from the first 10 agents, who have not rejected the item. We will refer to this mechanism as random allocation with declines or simply random allocation mechanism.

In period 1 an A item arrives. One of the first 10 agents gets randomly selected and is offered the item. In period 1 agent 9 is selected. As the agent is of the α type, the agent accepts the item and leaves the queue (cf. Remark 1). If we now looked again at the first 10 agents, agent 9 would be gone and another agent, e.g. agent 11, would be on the list of the first 10 agents.

In the second period a B item arrives, which is then offered to agent 5. The agent is of the α type and therefore compares the expected waiting time for an A item with her willingness to wait for that item (cf. Lemma 1). The agent's willingness to wait is $\bar{w} = 4$ periods, while she expects that she would receive an A item in $E(t_x) = 3$ periods.[4] Therefore, agent 5 rejects the item and remains on the waiting list, while the item gets offered to another randomly picked agent. It is offered to agent 2, who is of the β type and, thus, accepts the item and leaves the queue.

Period 3 commences with a B item arriving. The item is offered to agent 6, who is of type α. Her willingness to wait is also $\bar{w} = 4$ periods and she expects that she would receive an A item in $E(t_x) = 20$ periods. Therefore, she accepts the mismatched item and leaves the queue.

The result of the matching after the three described periods is reported in Table 2. In this example the average misallocation rate is

$$\begin{aligned}\xi &= \frac{1}{T}\sum_{t=1}^{T}\xi_t \\ &= \frac{1}{3}\cdot(0+1+0) \\ &= \frac{1}{3}\end{aligned}$$

and the average welfare loss

$$\begin{aligned}\text{WFL} &= (1-x)\xi \\ &= (1-0.4)\frac{1}{3} \\ &= 0.2.\end{aligned}$$

[4] For the sake of illustrating the proceedings of the mechanism, we do not use the real expected waiting time here. The real expected waiting time would be 20 periods, because there is a 10% probability of getting the item offered and a 50% probability that the item is of the favorite kind $\left(E(t_x) = \frac{1}{0.1\cdot 0.5} = 20\right)$. Note that every agent would accept any item in this scenario and we therefore do not have to take the impact of rejections into account.

		Item value					
Agent	Value	v_A	v_B	$\bar{w}$	Matched type	ξ_t	WFL_t
2	β	0.4	1	4	B	0	0
6	α	1	0.4	4	B	1	0.6
9	α	1	0.4	9	A	0	0
					Average:	0.33	0.13

Table 2: Matching results of random allocation mechanism used on baseline example. The maximum amount of periods that an agent is willing to wait $\bar{w} = \frac{1-x}{c}$ is calculated with $c = 0.1$ and $\text{WFL}_t = (1-x)\xi_t$. The averages of ξ_i and WFL_t are rounded to two decimals.

2.2 Allocation using SSSB's mechanism

The SSSB allocation mechanism assumes that per time period in the queue, each agent accumulates one credit day. We denote agent i's credit days at time t as d_{it}. The mechanism then proceeds as follows: Each period, the available item is announced to the queuing agents. The agents can then apply to be assigned to the item. At the end of the period, the item will be assigned to the applicant with the highest amount of credit days. If two or more agents have the same amount of credit days, the agent who signed up to the queue earlier is chosen. Remark 2 directly follows from this description of the mechanism.

Remark 2. *SSSB's mechanism matches an agent i with an available item h_k if and only if there exists no other agent $j \neq i$ with more credit days ($d_{jt} > d_{it}$), who has applied to be assigned to the item.*

2.2.1 Agent's decision in SSSB's mechanism

We will now investigate which decision the agents have to make in SSSB's mechanism and explain what determines their decision.

In each period each agent faces the decision if she would like to be matched to the available item or if she would rather want to wait. As there are no costs associated with applying to or withdrawing from an item, it seems rational that each agent always applies to an available item. However, SSSB penalizes declining an item after getting it offered, which is highly undesirable for agents. Therefore, agents will not simply apply, but decide if they want to be matched to an item or not before they do. In other words, only if agents would want to be matched to an available item (i.e. would accept the item if they are the successful applicant), they will apply to it. We will argue that it is possible for agents to make the decision, if they want to be matched to an item or not, independent from other agents' application decisions.

From Remark 1 we know that if an available item is of the agent's preferred type, an agent prefers that item over waiting and would therefore apply to it. If an item's type is not of the agent's favorite type, the agent's decision depends on the expected waiting periods until being assigned to her favorite item and her willingness to wait (cf. equation (4)). Recall that an agent's willingness to wait is $\bar{w} = \frac{1-v_k}{c}$. The agent knows

the maximum periods that she is willing to wait, because she has information about her valuation of the available item as well as the waiting cost.

For the expected waiting periods until being assigned to a favorite item, we take into account that the agents observe the amount of credit days of the highest applicant to an item. Observing these amounts over multiple periods allows the agents to form expectations regarding the needed credit days for an item type. Knowing their own credit days, the agents then, based on their observations, form an expectation about how much longer they would have to wait to be the successful applicant for an item of their favorite type. We assume that this learning converges to the true expected waiting time.[5]

In remark 3 we summarize the results from this subsection.

Remark 3. *In SSSB's mechanism, agents will not apply to an item if their willingness to wait is greater than or equal to the expected waiting time (i.e. if equation* (4) *holds). Agents base their expected waiting time on observations of past waiting times needed. This converges to the true expected waiting time* $E(t_x)$*.*

In the case of homogeneous mismatch values, agents of the type that do not prefer the available item will all make the same decision, which means that either every mismatched agent or no mismatched agent applies. While it is feasible to standardize the values of favorite items to 1, it is in reality an unlikely scenario that all agents have the same mismatch value. Therefore, as one of the extensions to the model we will relax the assumption of homogeneous mismatch values in section 4.2.

2.2.2 Assignment of the item

After deriving which agents are going to apply for the available item, we will now turn to how the assignment of the item takes place.

Claim 1. *The outcome of the assignment of an available item in one period can be described by the following mechanism:*

1. *Sort the general waiting list in descending order by the amount of credit days that an agent has accumulated (i.e. the agent with the most credit days is the first agent on the list).*[6]
2. *Offer the available item to the first agent on the list that has not yet rejected the available item. If the agent accepts the item, she takes it and leaves the queue. If she declines, the item will be offered to the next agent on the waiting list.*
3. *Repeat step 2 until the item is assigned.*

We call the mechanism proposed in Claim 1 a First Come First Served (FCFS) mechanism, where agents are allowed to decline items and keep their position in the queue i.e. it is an FCFS mechanism with declines. Based on the description of the FCFS mechanism in Claim 1, we can note Remark 4.

Remark 4. *The FCFS mechanism proposed in Claim 1 matches an agent* i *with an available item* h_k *if and only if there exists no other agent* $j \neq i$ *higher on the general waiting list, who accepted the item when she got it offered.*

[5] We will later in section 4.2.1 elaborate on how to mathematically implement such learning dynamics and will see that the agents' expectations convergence to the true values. This leads us to assume that agents have correct expectations.

[6] As in SSSB's mechanism, if two or more agents have the same amount of credit days, the agent who signed up to the queue earlier is chosen.

Proposition 1. *The outcome of SSSB's allocation mechanism is the same as the outcome of an FCFS mechanism with declines, for which the amount of accumulated credit days serves to determine the order of picks (with highest credit days corresponding to getting the first pick).*

Proof. We will prove Proposition 1 by showing that Remark 2 and Remark 4 are equivalent. To see that note the following. Firstly, an agent that is higher on the general waiting list is an agent with more credit days, as the FCFS mechanism sorts the waiting list by credit days in descending order. Secondly, in SSSB's mechanism, an agent who applies to the available item prefers that item over waiting and, thus, is an agent who would accept the item if it was offered to her (cf. section 2.2.1). Therefore, it is the same agent that receives the item in both mechanisms. As the available item is allocated in the same way in each period, we have established that the allocation of both mechanisms is equivalent. □

A more intuitive explanation for why Proposition 1 holds is that both mechanisms check the same properties, but proceed in a different order. Whereas SSSB's mechanism first checks who wants it and then checks who of those has the most points, the FCFS mechanism first checks who has the highest points and then if that agent wants it. The FCFS mechanism might have to check multiple agents before assigning the item, but eventually assigns it to the same agent.

2.2.3 Agent's decision in the FCFS mechanism

In the previous section we have established that SSSB's mechanism will result in the same outcome as an FCFS mechanism with declines. Therefore, to analyze the outcome of SSSB's mechanism, we will now look at the FCFS mechanism and especially the agent's decision in the FCFS mechanism.

The FCFS mechanism is one of the main mechanisms in queueing theory (cf. Hassin and Haviv, 2003) and is often used in practice. Claim 1 describes how the FCFS mechanism with declines proceeds. In the SSSB mechanism agents only know their credit days, but their expectations about the waiting time converge to the true expected waiting time $E[t_x]$ by observing the credit days of the highest applicants. This full level of information corresponds to agents knowing their position on the waiting list (or rank) in the FCFS mechanism, because if agents know their rank, they can form the correct expectations about their waiting time, as we will show with equation (5).

When looking at the agent's decision, it still holds that, as before, an agent who is offered her favorite item will accept it. If the agent is offered her non-favorite item, she will compare the expected waiting time to her maximum willingness to wait.

Remark 5. *An agent will only be offered an item if she is the first one on the waiting list or if all agents that are ahead of her in the waiting list have already rejected the available item. As agents would never reject their favorite type, any agent that is offered an item will know that all agents in front of her are waiting for an item of the other type than the type available.*

Making this observation, we can use the rank of the agent on the waiting list in connection with the arrival probability of the favorite item type to calculate the expected waiting time for the favorite item. Denoting an agent's rank by k, we can formulate this

as

$$E[t_x] = \frac{1}{p_x} \cdot k, \tag{5}$$

where p_x is the arrival probability of a certain item type. Consider example 1 for clarification.

Example 1. Assume we are in the baseline example and that the agents on the general waiting list are already sorted in descending order by their credit days. In the first period an A item arrives and is offered to agent 1 on the list reported in Table 1, a type β agent. If the agent declined the item, she would still be the first agent on the waiting list and would be offered the next item that arrives. With an arrival probability of 0.5 for type B items, the expected waiting time until she would be offered a type B item is $E[t_x] = \frac{1}{0.5} = 2$ periods. The agent, however, is willing to wait $\bar{w} = 6$ periods. Therefore, the first agent rejects the item and it is offered to the second agent on the waiting list.
The second agent is also of type β. As the first agent has rejected the item, the second agent knows that the first agent also prefers B items. Therefore, she knows that she would only get the second B item that becomes available. This is because the first type B item that becomes available will be taken by the first agent. So agent 2's expected waiting time for a B item is $E[t_x] = \frac{1}{0.5} \cdot 2 = 4$ periods. As $4 \leq 6$, the second agent rejects the mismatched item as well and the item gets offered to the next agent.
Same argumentation holds for the third agent on the waiting list, who is also of type β. She will also wait for her favorite item, as she expects to wait $E[t_x] = \frac{1}{0.5} \cdot 3 = 6$ periods, which is equal to the time she would be willing to wait.
Agent 4, who is offered the mismatched item expects to wait $E[t_x] = \frac{1}{0.5} \cdot 4 = 8$ periods, which is longer than her maximum willingness to wait. Therefore, agent 4 accepts the mismatched item and leaves the queue, which concludes the first period.

Now that we have established that SSSB's mechanism and the FCFS mechanism are outcome equivalent, in the next section we go on to introduce another mechanism, which aims to improve the allocation achieved through SSSB's and the FCFS mechanism.

2.3 Allocation using the SIRO mechanism

In this section we will conduct the matching of agents and items with the help of the Service in Random Order (SIRO) mechanism, following Leshno (2017).

2.3.1 Introducing buffer-queues

As the SIRO mechanism uses the concept of buffer-queues, we first introduce this concept in this subsection and, for an easier comparison, show that the FCFS mechanism from section 2.2.3 can be represented with the help of buffer-queues as well.

Consider a waiting list where agents get offered an item in sequence of their position on the general waiting list and also are allowed to decline an offered item, similar to the FCFS mechanism from the previous section. However, now if an agent declines an item, she can join a separate, prioritized queue for her preferred item, a buffer-queue. If then an item of the preferred kind arrives, it will first be offered to the agents waiting in the buffer-queue. Otherwise, if the buffer-queue of that item type is empty, the item is offered to the agents on the general waiting list. The buffer-queues are themselves operated under

a prespecified mechanism. We will here and in the following refer to the mechanism under which a buffer-queue is operated as a buffer-queue policy, whereas we reserve the term mechanism for the whole allocation process.

Note that using a buffer-queue to describe a mechanism is not necessary, but allows for a more convenient description of the mechanism's proceedings.[7] See Remark 6 to understand that the FCFS mechanism from the previous period can also be expressed by using a buffer-queue.

Remark 6. *The FCFS mechanism with declines from the previous section can also be expressed as a buffer-queue mechanism. Both, the type A and type B buffer-queues, would then be operated under an FCFS buffer-queue policy. That means agents who decline an item, will join the buffer-queue for their favorite item and once an item of the favorite type arrives, it will be offered to the agent who joined the respective buffer-queue first.*

Next we introduce the concept of incentive compatibility to be able to describe a special position in the buffer-queue, which will become important for our comparison of mechanisms.

Definition 4. A buffer-queue position is incentive compatible (IC) if joining the buffer-queue in that position will give the agent the most expected value, i.e. if it is rational for the agent to join in that position.

Definition 5. With K_A we denote the last position for which it is IC to join the buffer-queue for type A items. Analogously, let K_B denote the same for type B items.

Our definition is equivalent to defining K to denote the maximum IC buffer-queue length. Note that K is directly dependent on the waiting time for the favorite item that an agent expects when joining a buffer-queue.

Example 2. Remember Example 1, where in period 1 the first three β agents from the waiting list (see Table 1) rejected a mismatched A item. The fourth agent, however, accepted the mismatched item, because her expected waiting time was higher than her willingness to wait, i.e. it was not rational for her to wait for a B item. In this example, $K_B = 3$.

The concept of the maximum IC buffer-queue size allows us to specify Proposition 2.

Proposition 2. *Consider the mechanism μ_1 with maximum IC buffer-queue sizes K_{A1} and K_{B1} and mechanism μ_2 with maximum buffer-queue sizes K_{A2} and K_{B2}.*

If $K_{A1} \geq K_{A2}$ and $K_{B1} \geq K_{B2}$ and at least one of the equation holds with strict inequality, then we have that $WFL_{\mu_1} \leq WFL_{\mu_2}$.

Proof. A greater maximum IC buffer-queue size, i.e. higher value for K, means that it is rational for more agents to join the buffer-queue instead of accepting a misallocated item. For any finite K, there exists a positive probability that the imbalance between supplied and demanded items gets sufficiently big, such that K agents are in the buffer-queue. If K agents are in the buffer-queue, there also exists the probability of a mismatch, as it would not be IC anymore to join the buffer-queue. If now K was larger, the agent would have still joined the buffer-queue and the mismatch had not happened. Therefore, for a higher value of K, the probability of a mismatch decreases in the long-run. The decreased long-run misallocation rate then decreases the WFL. □

[7] Additionally, the buffer-queue representation allows for an easier depiction of the system as a Markov chain (cf. Appendix B)

Note that Proposition 2 is independent of the buffer-queue policy used, but solely relies on K. This is the starting point for Leshno's (2017) derivation of alternative buffer-queue policies, which we will elaborate more on in the next section.

2.3.2 SIRO as a robust buffer-queue policy

In this subsection we will introduce the SIRO buffer-queue policy, a robust buffer-queue policy that we will use to decrease the WFL in comparison to the FCFS mechanism.

As we can see in equation (5), the expected waiting time for an item type when using the FCFS mechanism increases linearly with each agent that waits for that type. This implies that agents that join the buffer-queue early have a relatively short expected waiting time, while agents who join later have a higher expected waiting time. Leshno (2017) tries to make it rational for more agents to decline an item and instead join a buffer-queue (i.e. to increase K), which would then result in less welfare loss in the long-run following Proposition 2. The author does so by introducing two other buffer-queue policies, namely the Load Independent Expected Wait (LIEW) and the Service in Random Order (SIRO) buffer-queue policy.

LIEW is the optimal buffer-queue policy, which minimizes the probability of misallocations by controlling the expected waiting time for each agent joining the buffer-queue. The policy imposes that the expected waiting time for each agent is exactly the maximum waiting time that an agent would wait for her favorite item. Therefore, regardless of the position at which an agent joins the buffer-queue, each agent has the same expected waiting time.[8] However, LIEW requires that the planner knows exactly the amount of periods that agents would wait for their favorite item. Additionally, there has to be a maximum of agents that can join the buffer-queue (i.e. a cap),[9] because an agent's expected waiting time depends among other factors on how likely it is that other agents join the buffer-queue as well. Also LIEW requires that agents have correct beliefs about subsequent agents joining the buffer-queue. These constraints make the LIEW buffer-queue policy tough to implement in practice and are the reason for Leshno (2017) to introduce the more robust SIRO buffer-queue policy.

Definition 6. Under the service in random order (SIRO) buffer-queue policy, each agent in the buffer-queue has the same chance of receiving an item.

Definition 7. The SIRO mechanism offers agents items in the order of their rank on the general waiting list and allows agents to join a buffer-queue of their choice if they decline an offered item. There exists a buffer-queue for each item type and all are operated under the SIRO buffer-queue policy.

An intuitive summary of the SIRO mechanism is that it "gives equal top priority to any agent who previously rejected an item" (Leshno, 2017, p. 31). As the quote captures well, the derivation of the mechanism and the proofs of its desired properties lead up to an intuitive final result. As for LIEW, the cap is needed for the calculation of the expected waiting time for a position in the buffer-queue, because it has to be taken into account

[8] Note that this does not mean that agents who are in the queue will get an arriving item with the same probability, but instead that in the moment of joining the queue, agents have the same expected waiting time.

[9] If a buffer-queue is full, the agent on the general waiting list, who is offered the available item, has no choice but to accept the item.

how many agents might join the buffer-queue afterwards.[10] SIRO reduces misallocation in comparison to FCFS by increasing the expected waiting time for agents that join the list early and decreasing the expected waiting time for agents that join the list later. By doing so, even agents that are offered an item when there are already some other agents in the buffer-queue are more likely to join the buffer-queue than accepting a mismatch, as we will show in Example 4.

In comparison to the LIEW mechanism, the SIRO mechanism does not minimize the misallocation, but still reduces it compared to the FCFS mechanism. The extend to how much SIRO reduces the misallocation in comparison to FCFS depends on the specific parameters. The following example illustrates how the SIRO mechanism and its buffer-queue policy work.

Example 3. Remember Example 1, where in period 1 the first three β agents from the waiting list (see Table 1) rejected the mismatched A item. Under the FCFS mechanism, these agents just remained on the waiting list. Now, under SIRO, these agents would have instead joined the buffer-queue for B items. So after period 1, there would be three agents in the buffer-queue for B items.
Then in period 2 a B item arrives. As the buffer-queue for type B items is non-empty, the item is first offered to the three agents in the buffer-queue. The SIRO buffer-queue policy implies that the item is assigned to one of the three agents with equal probability (ca. 33.3%), i.e. randomly.

The SIRO mechanism has some desirable properties. Leshno (2017) proofs that this mechanism is "Belief-Free Incentive Compatible". If a mechanism is incentive compatible, it means that it is in the agents' best interest to respond truthfully, i.e. responding truthfully is a weakly dominant strategy. Belief-free adds that the incentive compatibility holds independent of the agent's beliefs about the probabilities of other agents joining the buffer-queue after her and the belief about the arrival probability of the different kinds of items.

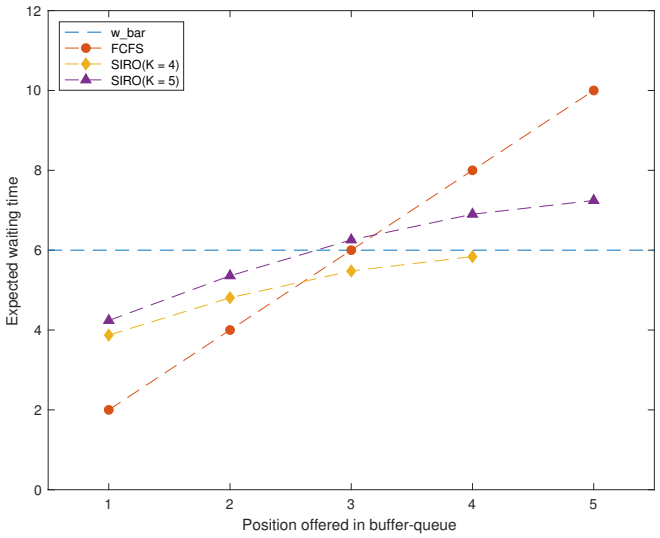
2.3.3 Agent's decision in the SIRO mechanism

Again, the agent faces a decision whether to accept or decline an offered item, which is based on equation (4). Now under the SIRO mechanism, the expected waiting time is different in comparison to the FCFS mechanism.

Figure 1 shows the expected waiting time for each position in the buffer-queue under different buffer-queue policies for equal arrival probabilities $p_A = p_B = 0.5$ and a willingness to wait of $\bar{w} = 6$. Note that if the SIRO buffer-queue is capped at 5 agents, it would not be IC to join on the fourth or fifth position ($K = 3$), whereas if the buffer-queue is capped at 4 agents, it is IC to join the buffer-queue even in the fourth position ($K = 4$). As the agents' expected waiting time in the FCFS mechanism is independent of how many agents join afterwards, there is no need to specify a cap for FCFS. In our example, position 3 is the last position that is IC for the FCFS mechanism. We illustrate how the SIRO mechanism impacts agents' decision making in comparison to the FCFS mechanism with Example 4.

Example 4. Recall Example 1, where with the FCFS mechanism in period 1 the first three agents rejected the mismatched A item and the agent 4 then accepted the mismatched

[10] We will at the end of subsection 2.3.3 argue that it is also possible and even desirable for higher robustness to implement the SIRO mechanism without specifying a cap.



Note: Adapted from "Dynamic Matching in Overloaded Waiting Lists", by Leshno, J., 2017, p. 36.

Figure 1: Expected waiting times for positions in the FCFS and SIRO buffer-queue. For the SIRO buffer-queue the figure shows the waiting times for two different caps K. The parameters are $p_A = p_\alpha = \frac{1}{2}$ and v and c such that the willingness to wait is $\bar{w} = 6$.

item because her expected waiting time was $\frac{4}{0.5} = 8$, which was greater than her willingness to wait of $\bar{w} = 6$ periods.
Now we consider the same example, but assume that a SIRO mechanism with a cap of four agents is implemented. Instead of 8, the expected waiting time for agent 4 would then be ca. 5.84 periods.[11] This is shorter than the maximum acceptable waiting time of $\bar{w} = 6$ periods and, thus, the agent would decide to join the buffer-queue instead of accepting the mismatched item.

Leshno (2017) shows that if there is a cap on the buffer-queues, it is possible to calculate an agent's expected waiting time by exploiting that the system can only be in a finite number of states (see Appendix C.1). Following Proposition 2, the mechanism's designer would optimally select the cap, which corresponds to the highest possible maximum IC buffer-queue length. However, if the designer does not have all necessary or wrong information, this can be difficult. Additionally, the mechanism would be less prone to errors if less parameters needed to be decided upon (i.e. the mechanism would be more robust).

Therefore, Leshno mentions the possibility of not imposing a cap on the SIRO buffer-queues. In that case, the agents would have to calculate the maximum IC buffer-queue length to be able to form an estimate about how many agents would join the buffer-queue after they have joined the buffer-queue. However, it is also a possibility to provide agents with the historical waiting times for the buffer-queue of their favorite item, depending on the position joined. Altman and Shimkin (1997) show that under the minor assumption that there exists a small fraction of agents in the population, that will always join the waiting list, the agents will learn from the previous waiting times and their estimates will converge to an equilibrium.

Therefore, we will assume that, similar to SSSB's mechanism, the agents learn from previous waiting times. Instead of observing the highest amount of waiting days, however, in the SIRO mechanism the agents have information on the average amount of periods that

[11] See Appendix C.1 for how to calculate the expected waiting time under a SIRO buffer-queue policy.

agents waited for their favorite item in the buffer-queue depending on the position that they joined. This information will get updated every period. Knowing their maximum willingness to wait and expecting a waiting time equal to the average of previous agents, who joined the buffer-queue in the same position, each agent can make a decision if she wants to accept an offered item or join the buffer-queue.

3 Comparison of the FCFS and SIRO mechanisms and sensitivity analysis

In this section we will investigate the performance of the FCFS and the SIRO mechanism by comparing their WFL. We start by deriving the theoretical misallocation rate, which is not mechanism-specific, but depends on the maximum IC buffer-queue lengths K for the implemented mechanism. Following this, we then explain how to compute the maximum IC buffer-queue lengths K, as this will be helpful to better understand the transmission between the variables in the rest of the section. Subsequently, we will change some of the model's parameters and shed light on their impact for each mechanism's performance. Based on the earlier results of this section, the long-run WFL for all comparisons can be computed using the maximum IC queue length K for the specified parameters.

3.1 Theoretical misallocation rate

To compare the performance of the allocation under the FCFS (and, thus, also SSSB's mechanism) and the SIRO mechanism, we will now look into the theoretical misallocation rate for the two mechanisms.

Remark 7. *Independent of the buffer-queue policy used, in every period there can only be one buffer-queue that contains agents.*

Remark 7 directly follows from the way buffer-queues are set up and becomes clear when considering the following example.

Example 5. Consider the type β agent in the first position on the general waiting list. The agent will only join the buffer-queue for type B items, if she is offered a type A item. However, she will only be offered a type A item, if the buffer-queue for type A items is empty, otherwise it would be offered to the agents in the buffer-queue first.

This implies that for finite K, the system consisting of the two buffer-queues can only be in a finite amount of states. Also, we can now say that the different states represent an imbalance between the supplied type of items and the waiting agents' types (i.e. their demand for a certain type). To gain a better understanding of the different states of the system consider Example 6.

Example 6. Consider general example under the FCFS mechanism. Initially, both buffer-queues are empty. When the A item arrives in the beginning of the first period, it is offered to the first agent on the list. The first agent is of type β and rejects the item. Now our system is in a state with one agent in the buffer-queue for the A item. The mechanism still tries to assign the A item. It is offered to the next agent on the list, who also rejects it and the state of the system is now that 2 agents are in the buffer-queue for a B item.

Evidently, misallocation can only occur in states of the system, where the imbalance has become sufficiently big. In our example with $K = 3$, only in a state where 3 agents are in the buffer-queue. If an item is then offered to an agent who does not prefer the available item, the buffer-queue will be too full so that it would be rational to join and, thus, misallocation occurs. Exploiting this property, Leshno (2017) derives the theoretical misallocation rate, subject to the maximum buffer-queue size by using a Markov chain representing the different states that the system can be in.[12] With $p = p_A = p_\alpha$, the author arrives at a long-run misallocation rate of

$$\xi = \frac{2p(1-p)}{(1-p)K_A + pK_B + 1}. \tag{6}$$

We can see that the long-run misallocation rate decreases for higher values of K, which is in accordance with Proposition 2. Note further that the long-run misallocation rate does not rely on mechanism-specific parameters, but that it is a probability of being in a certain state of the system, where it is rational for an agent to accept a mismatched item. From the long-run misallocation rate, we can directly calculate the WFL as specified in equation (2).

3.2 Calculation of maximum IC buffer-queue length

As the maximum IC buffer-queue lengths K directly impacts the WFL (cf. equation (6)), they play an important role when comparing the FCFS and SIRO mechanism. Therefore, we will use this subsection to investigate how to compute the maximum IC buffer-queue lengths K for the FCFS and the SIRO mechanism.

3.2.1 FCFS

To derive the agent's maximum IC buffer-queue length, recognize that we are trying to find out which position in a buffer-queue is the last position for which it would be rational to join. We start with equation (5), which gives us the agent's expected waiting time for their favorite item if offered a mismatched item. Substituting the expected waiting time into equation (4) and rearranging for the agent's position k results in

$$\begin{aligned} \frac{1}{p_A} \cdot k &\leq \bar{w} \\ k &\leq p_A \bar{w}. \end{aligned} \tag{7}$$

As the positions are noted in integer values, we rewrite equation (7) as

$$k \leq \lfloor p_A \bar{w} \rfloor, \tag{8}$$

where $\lfloor p_A \bar{w} \rfloor$ is the greatest integer smaller than $p_A \bar{w}$. Equation (8) means that in order for the agent to decline a mismatched item, the offered position in the buffer-queue has to be smaller than $p_A \bar{w}$. The last position is the greatest k and therefore exactly, when equation (8) holds with equality. Then we have

$$K_A = \lfloor p_A \bar{w} \rfloor. \tag{9}$$

Equation (9) gives us the maximum IC buffer-queue length for the item type A buffer-queue. The argumentation holds similarly for type B items. For clarification, we introduce the following example.

[12] Please refer to Appendix B for more background on the Markov Chain.

Example 7. If we again consider the baseline example from 2.1.4, we have that the willingness to wait is $\bar{w} = \frac{1-0.4}{0.1} = 6$. If we use the FCFS mechanism for allocation, then the maximum number of agents on the waiting list who have previously rejected an item is $K_B = 0.5 \cdot 6 = 3$. We have already seen that this is true: in Example 1 we conducted the allocation with the FCFS mechanism and have shown that there were 3 agents who rejected the mismatched item whereas the fourth agent accepted it.

3.2.2 SIRO

For the SIRO buffer-queue, deriving the maximum IC buffer-queue length K is more complex, because an agent's expected waiting time is dependent on how many agents will join the buffer-queue afterwards, whereas actions of later agents do not play a role for the expected waiting time in the FCFS mechanism.

Recall from the previous section, that we are looking for the last position, for which it would be rational to join the waiting list. It would be rational if the expected waiting time on the waiting list is lower than the agent's willingness to wait $\bar{w}$. Therefore, to derive the maximum IC buffer-queue length, we start by computing the expected waiting time of the last agent, who joins a buffer-queue that is capped at 1 agent and compare it with the agent's willingness to wait.[13] Should the willingness to wait be greater, we calculate the expected waiting time of the last agent, who joins a buffer-queue that is capped at 2 agents and again compare it to the agent's willingness to wait. We proceed with this until the expected waiting time in the buffer-queue for the favorite item for the first time exceeds the agent's willingness to wait. Then we know that the previous cap is the maximum IC buffer-queue length. Consider Figure 2, which reports the Matlab code for this algorithm, for clarification. The function "fun_expected_wait_SIRO" takes a given maximum IC buffer-queue length and computes the expected wait of the last agent joining that buffer-queue with the help of the dynamic equations (cf. Appendix C.1).

```
1      K_A = 0;
2          while exp_wait ≤ w_bar
3              K_A = K_A+1;
4              exp_wait = fun_expected_wait_SIRO(K_A);
5          end
6      K_A = K_A-1;
```

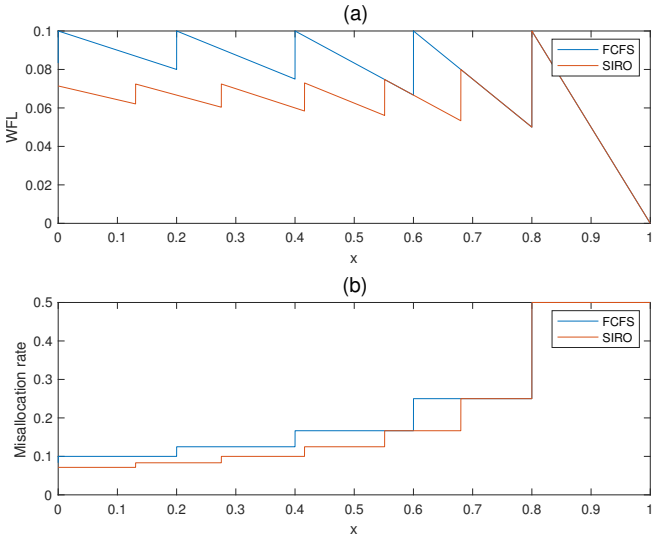
Figure 2: Matlab code summarizing the computation of the maximum IC buffer-queue length K for SIRO buffer-queues.

3.3 WFL and misallocation rate for different mismatch values

Now we will turn to evaluating the impact of changes in parameters on the allocation. In Figure 3 we plot FCFS' and SIRO's WFL (a) and the misallocation rate (b) for different mismatch values x.

We observe in Figure 3(a) that for mismatch values between 0 and ca. 0.55 the SIRO mechanism results in a strictly smaller WFL than the FCFS mechanism. While between mismatch values of ca. 0.55 and ca. 0.68 SIRO still performs weakly better than FCFS, both mechanisms result in the same WFL for mismatch values higher than ca. 0.68.

[13] The expected waiting time can be computed by using the dynamic equations (see Appendix C.1).



Note: (a) adapted from "Dynamic Matching in Overloaded Waiting Lists", by Leshno, J., 2017, p. 37.

Figure 3: Comparison of FCFS and the SIRO mechanism for changing mismatch values x. Figure (a) shows the WFL for different mismatch values and Figure (b) the corresponding misallocation rate for different mismatch values. Other parameters are arrival probabilities $p_A = p_\alpha = 0.5$ and waiting costs $c = 0.1$.

The discontinuity points ("jumps") mark the mismatch values for which a different maximum buffer-queue length becomes incentive compatible. For clarification refer to Figure 10 in the appendix, where we have added the maximum IC buffer-queue lengths for both mechanisms into Figure 3(a). For low mismatch values, the maximum IC buffer-queue length of both mechanisms is longer, because agents are willing to wait longer for their favorite item. As mismatch values increase, agents will not be willing to spend as much time queuing for their favorite item. At a value of ca. 0.8, we reach a point where agents will no longer join buffer-queues but instead accept a mismatch item in any case, because the mismatched item still gives them a relatively high value.

For low mismatch values the SIRO mechanism has a higher maximum IC buffer-queue length. Therefore, more agents join the buffer-queue and it will less often occur that an agent rejects an item, i.e. the misallocation rate is lower. We can see this in Figure 3(b). Again, the discontinuity points occur between different maximum IC buffer-queue lengths. Note that when the misallocation rate increases with the shorter buffer-queue lengths, we would also expect a higher WFL. However, the higher mismatch value counteracts this.

3.4 WFL for different waiting costs

In Figure 4 we depict for FCFS and SIRO the WFL for different waiting costs for a fixed arrival probability and fixed mismatch values on the left y-axis. Additionally, we also report the corresponding maximum IC buffer-queue lengths K on the right y-axis. Note that in a balanced economy, we have that the maximum IC buffer-queue lengths are the same for both item types ($K_A = K_B = K$).

With one exception at waiting costs of around 0.22, where both mechanisms result in the same WFL, SIRO performs better than FCFS until waiting costs of ca. 0.3. Afterwards, both mechanisms result in the same WFL.

Recall that the WFL is calculated according to Equation (6). Waiting costs c impact

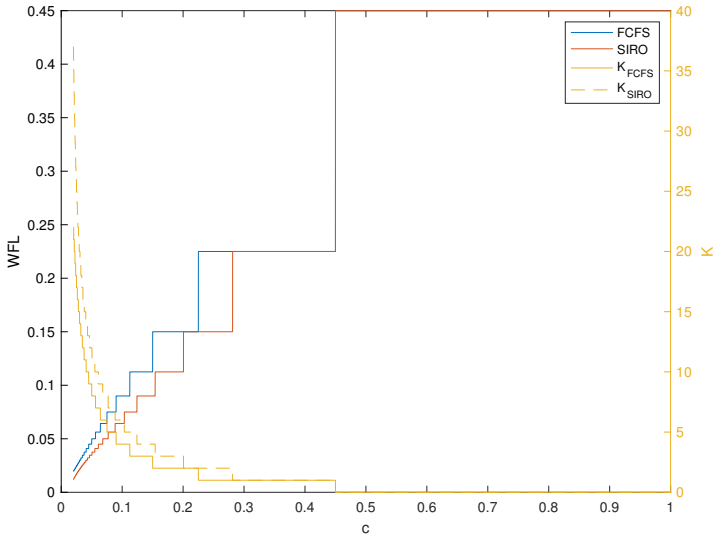


Figure 4: Comparison of the FCFS and the SIRO mechanism's WFL for different waiting costs c. Other parameters are arrival probabilities $p_A = p_\alpha = 0.5$ and mismatch value $x = 0.1$.

the WFL through the following transmission: Higher waiting costs ($c \uparrow$) decrease the willingness to wait ($\bar{w} \downarrow$), which, for a constant expected waiting time for the favorite item $E[t_x]$, makes it for less agents rational to join a buffer-queue ($K \downarrow$). The decreased maximum IC buffer-queue length then impacts WFL through the long-run misallocation rate (cf. equation (6)). We also reported K in Figure 4 to bring the relation of K and WFL to mind. In line with our argumentation, we observe again that the discontinuity points in the graph occur when there is a change in the maximum IC buffer-queue length K.

Note that the willingness to wait $\bar{w} = \frac{1-x}{c}$ is not defined for $c = 0$. However, as c approaches 0 we get for the willingness to wait that

$$\begin{aligned} \lim_{t\to\infty} \bar{w} &= \lim_{t\to\infty} \frac{1-x}{c} \\ &= \infty. \end{aligned}$$

This result is in line with what we would intuitively expect: If waiting costs approach zero, agents are willing to wait however long it takes to receive their favorite item. Thus, agents would never accept a mismatched item, which means that the maximum IC buffer-queue length K is not defined and that there will be no WFL. In Figure 4 we only report values starting at 0.02, because even for $c = 0.01$ finding the maximum IC buffer-queue length K for the SIRO mechanism is computationally demanding and the additional insight from it limited.

For different mismatch values x, we would still observe a similar shape that we observe in Figure 4, but see that for higher mismatch values the maximum WFL is lower while we reach the maximum WFL already for lower waiting costs. We can see this clearly in Figure 5, where we depict the FCFS mechanism's WFL for both, different waiting costs and mismatch values, at the same time.[14] Intuitively, this makes sense because

[14] The equivalent figure for the SIRO mechanism shows the same characteristics, therefore we exemplary only report the figure for FCFS here. Refer to Figure 11 in the Appendix for a figure containing both mechanisms.

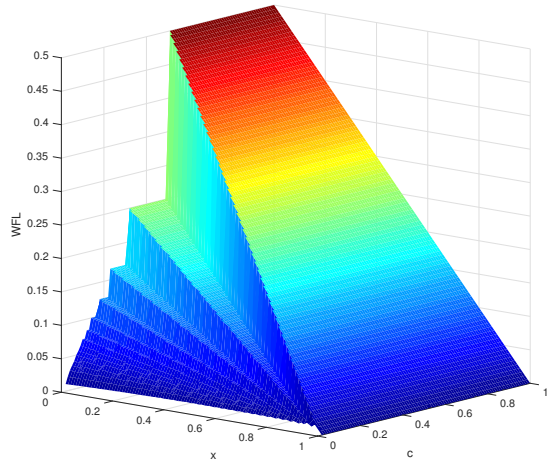


Figure 5: WFL for different mismatch values and waiting costs for the FCFS mechanism. Other parameters are $p_A = p_\alpha = 0.5$.

higher mismatch values make the agents less willing to wait, which in turn decreases the maximum IC buffer-queue length K already for lower waiting costs c. The increase in the misallocation rate increases the WFL, but is offset by the higher mismatch value x, which is in line with our findings from section 3.3.

3.5 WFL for different arrival probabilities

Lastly, we check the impact of different arrival probabilities on the WFL.

In order to measure the impact of different arrival probabilities, we have to change the waiting costs to $c = 0.2 \cdot p_A$. This setup entails that the expected waiting costs until the next item of the favorite type arrives are constant (in our case 0.2), independent of the changes in the arrival probability p_A. To see that, recall from equation (5) that the periods until an A item becomes available are in expectation $\frac{1}{p_A}$. If we let Γ denote the expected waiting costs until an item of the agent's favorite type becomes available, we can show that

$$\begin{aligned}\Gamma &= c \cdot \frac{1}{p_A} \\ &= 0.2 \cdot p_A \frac{1}{p_A} \cdot \frac{1}{\frac{1}{n}} \\ &= 0.2,\end{aligned}$$

which is constant and, thus, does not depend on the different arrival probabilities p_A. However, this also means that the costs c are changing for each specification of n.

If we had not adapted the waiting costs, but instead had just set c to a constant value, we would not measure the impact of changing arrival probabilities alone, but also the impact of changing waiting costs. This is for the reason that with constant waiting costs c and a higher (lower) arrival probability p_A, the time it in expectation takes for an A item to arrive decreases (increases). Then, the expected waiting costs until an agent receives an item of her favorite kind would decrease as well and we would measure this effect in addition to the effect of a higher arrival probability.

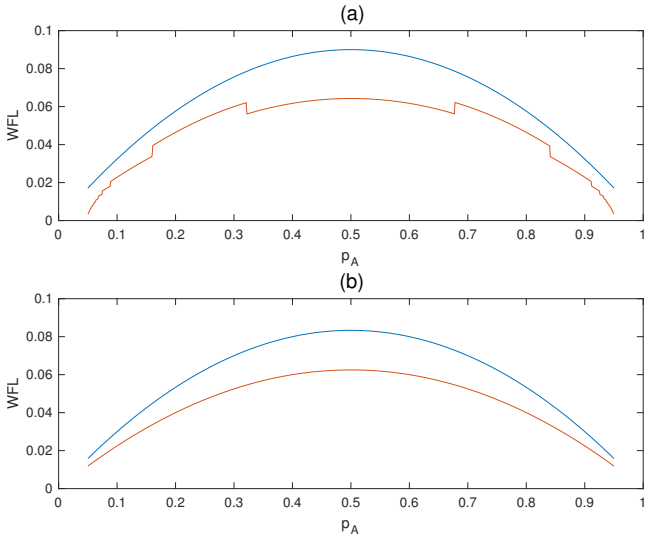


Figure 6: Comparison of the FCFS and the SIRO mechanism's WFL for different arrival probabilities p_A. In (a) the mismatch value is $x = 0.1$ and in (b) the mismatch value is $x = 0.5$. Waiting costs in both cases are $c = 0.2 \cdot p_A$.

In Figure 6(a) and (b), we see how WFL changes with regard to different arrival probabilities. Note that for an arrival probability of $p_A = 0.5$ both items arrive with the same probability and that an arrival probability of e.g. $p_A = 0.3$ implies that the other item arrives with a probability of $p_B = 1 - 0.3 = 0.7$. Due to this relation, we can observe that the graphs are mirrored at $p_A = p_B = 0.5$.

When comparing Figure 6(a) and (b), we see that the shape of the graph remains the same for the FCFS mechanism, but differs in magnitude of WFL. For the SIRO mechanism, we even see different shapes of the plot for WFL. The different shapes in the WFL plot for the SIRO mechanism can be explained by the influence that p_A has on the the maximum IC buffer-queue length through the dynamic equations (see Appendix C.1). Overall, the impact of different arrival probabilities p_A is tough to assess theoretically.

In both reported cases, however, SIRO performs better than the FCFS mechanism. To confirm this, we investigated the WFL for different arrival probabilities and different mismatch values at the same time and still conclude that SIRO performs weakly better than FCFS. A corresponding graph is reported as Figure 12 in the Appendix. Doing the same for different arrival probabilities and different waiting costs until a specific item arrives at the same time also confirms this assessment. The corresponding graph is reported as Figure 13 in the Appendix.

3.6 Summary

In summary, we can conclude that the SIRO mechanism outperforms the FCFS mechanism, and thus SSSB's mechanism, for most parameters. For some parameters, both mechanisms perform equally well, whereas FCFS never performs better than SIRO.

The result that the SIRO mechanism outperforms SSSB's mechanism is not surprising, if we recapitulate the line of argumentation so far: In section 2.2.2 we have shown that SSSB's mechanism is outcome equivalent to an FCFS mechanism. We then introduced buffer-queues in section 2.3.1 and explained that the FCFS mechanism can be represented as a buffer-queue mechanism. The representation as a buffer-queue allowed us to come

up with the intuitive motivation for the SIRO mechanism, which was to make it rational for more agents to join the buffer-queue (cf. section 2.3.2). Lastly, in section 3.1 we showed the correctness of the intuitive motivation, i.e. that increasing the maximum IC buffer-queue size decreases WFL.

The findings of our sensitivity analysis for the mismatch value are that an increase in mismatch value results in a lower WFL, but that WFL increases again when the mismatch value has increased enough so that the lower willingness to wait leads to a decrease in the maximum IC buffer-queue length. For waiting costs we found that higher waiting costs negatively impact the maximum IC buffer-queue size, which in return leads to a higher WFL. Arrival probabilities impact the WFL as well, but the variation is hard to assess, as the arrival probability appears multiple times in the equation of the theoretical misallocation rate as well as in the dynamic equations. If anything, we tend to see higher misallocation as well as a greater difference between the two mechanisms for more balanced arrival rates.

4 Extensions

In this section we will implement extensions to the introduced model to account for more realistic settings, which likely have an impact on the allocation of items. Leshno (2017) serves as a basis for the extensions to the n-item economy and heterogeneous mismatch values. We take Leshno's analysis further, by elaborating on the background of our observations and making a connection to the underlying concepts that we discussed in the previous section. For each of the extensions, we evaluate the impact on the FCFS and the SIRO mechanism and compare the mechanisms performances.[15]

4.1 n-item economy

We will start by extending the model to an n-item economy. First, we will discuss the changes to the model setup before we then use simulations to compare the mechanisms' performances. We want to investigate how WFL is impacted by having more than two different types of items and on which mechanism it has a bigger impact.

4.1.1 Implementation

Instead of only two item types, we now assume that there are n different types of items. The finite set $G = \{g_1, g_2, ..., g_n\}$ contains those different item types. The probability of an item being of type g_i is $p^G(g_i)$. Similar to the 2-item case, we assume that each agent prefers a single item type. Agents that prefer the same item type are denoted as the same agent type θ_i and the finite set $\Theta = \{\theta_1, \theta_2, ..., \theta_n\}$ contains these different types of agents. Consequently, there are just as many types of agents as there are item types ($n = |\Theta| = |G|$). Without loss of generality, we can assume that θ_i agents prefer type g_i items. The probability of an agent being of type θ_i is $p^\Theta(\theta_i)$. We again restrict attention to a balanced economy (i.e. $p^\Theta(\theta) = p^G(g)\ \forall \theta, g$). Additionally, we assume that all items arrive with the same probability $p^G(g) = \frac{1}{n}$.

[15] The simulations reported in this section have been conducted using the program MATLAB (version: R2018a). We report the code for our most extensive extension (section 4.3) in Appendix E. Other code can be provided upon request.

Remark 1, noting that agents always accept their favorite item, and Lemma 1, stating that agents compare the expected waiting time with their willingness to wait, both remain valid. As we still consider the same mismatch values for agents, an agent will never consider waiting for another item other than her favorite item. Because if the agent is offered a mismatched item, she would rather take the mismatched item than another mismatched item of same value in the future, due to the waiting costs $c > 0$.

Both mechanisms still proceed as presented earlier. Thus, the FCFS mechanism's allocation is still equivalent to the allocation of SSSB's mechanism. In both mechanisms, however, agents can now join n different buffer-queues. As an agent would never consider waiting for another item than her favorite item, agents will only join the buffer-queues for their favorite item type.

4.1.2 Simulation

As we cannot use the theoretical misallocation rate as stated in section 3.1 anymore for $n > 2$, we will have to simulate the model and calculate the WFL as stated in Definition 3. Our simulation comprises 10 iterations, which each start with empty buffer-queues and then run for 1,000,000 periods (i.e. assign 1,000,000 items). Note that in order to simulate an overloaded waiting list, it is enough to implement that it is always possible to draw a new agent to assign an item to if needed.

We follow Leshno (2017) in specifying the waiting costs as $c = 0.1 \cdot \frac{2}{n}$. Recall section 3.5 where we also had to adapt the waiting cost to measure the ceteris paribus impact of different arrival probabilities. Here, the same reasoning applies: we have to alter the waiting costs in a way that the waiting costs until an agent's favorite item becomes available are the same for each different amount of item types n. If we, again, let Γ denote the expected waiting costs until an item of the agent's favorite type becomes available, we can show that

$$\begin{aligned}\Gamma &= c \cdot \frac{1}{p^G(g)} \\ &= 0.1 \cdot \frac{2}{n} \cdot \frac{1}{\frac{1}{n}} \\ &= 0.1 \cdot 2 \\ &= 0.2,\end{aligned}$$

which is constant and, thus, does not depend on the number of different item types n.

The result of the simulation is reported in Figure 7.[16] We can see that for the FCFS mechanism the maximum IC buffer-queue length K remains constant, whereas the K for SIRO first drops and then increases.[17] With constant K, the WFL for FCFS increases in n, but the marginal WFL decreases. As SIRO's K increases, the WFL decreases. Therefore, we come to the conclusion that for a higher amount of item types n the benefit from using SIRO, instead of FCFS, increases.

We see that for constant K the WFL increases with higher n (cf. FCFS for all n and SIRO between 3–6 and 7–8 item types). The reason for this is that with increasing n, agents on the general waiting list are more often offered a mismatched item (with

[16] The WFL reported is the average over the WFL of the 10 iterations. The standard error for each different value of n for both mechanisms was below 0.1%.

[17] Note that we had to adapt the dynamic equations to incorporate the n-item economy to compute K. We report the adapted dynamic equations in Appendix C.2.

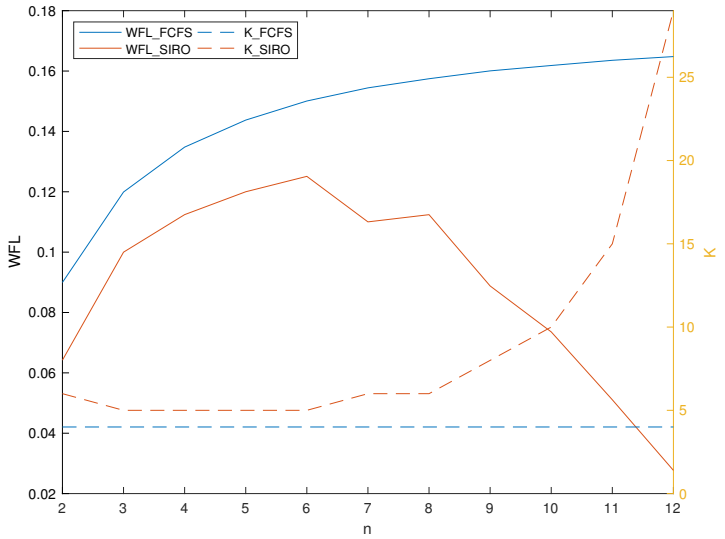


Figure 7: Comparison of the FCFS and the SIRO mechanism's WFL different amounts of item types n. Additionally, we show on the right y-axis, the corresponding maximum IC buffer-queue lengths K. Waiting costs are $c = 0.1 \cdot \frac{2}{n}$ and mismatch value $x = 0.1$.

probability $1 - \frac{1}{n}$). This leads to more agents joining a buffer-queue and consequently, that more agents are then in a position, where the maximum IC buffer-queue length is reached and they accept a mismatched item. The decreasing marginal effect can be explained by the fact that the probability of agents being offered a mismatched item also increases with decreasing marginal effect, i.e. the function for the probability $1 - \frac{1}{n}$ is convex for positive n.[18]

An examination of the willingness to wait $\bar{w}$ gives insight about the underlying reason for the respective maximum IC buffer-queue lengths, which then influence the shape of the WFL function for both mechanisms. In Figure 8 we report the agents willingness to wait $\bar{w}$ for different n. Additionally, we show the expected waiting time of the last agent, who joined a buffer-queue operated under the SIRO buffer-queue policy for multiple different maximum IC buffer-queue lengths. Recall that a buffer-queue is IC if the expected waiting time for the last agent is shorter than her willingness to wait $\bar{w}$. We see that for $n = 2$, a buffer-queue length of $K = 6$ is IC, whereas for n between 3–6, only a buffer-queue length of $K = 5$ is IC. This corresponds to our results from before and explains the shape of "K_SIRO" in Figure 7.

For the FCFS mechanism, we can show that the maximum IC buffer-queue lengths remain constant for each amount of different items n. From equation (9) that

$$K_g = \left\lfloor p^G(g) \cdot \bar{w} \right\rfloor$$

which, considering $p^G(g) = \frac{1}{n}\ \forall g$ and $c = 0.1 \cdot \frac{2}{n}$, we can alter so that

$$K_g = \left\lfloor \frac{1}{n} \cdot \frac{1-x}{c} \right\rfloor$$

$$K_g = \left\lfloor \frac{1}{n} \cdot \frac{1-x}{0.1 \cdot \frac{2}{n}} \right\rfloor$$

$$K_g = \left\lfloor \frac{1-x}{0.2} \right\rfloor,$$

[18] We can prove the convexity by showing $f'(x) = -\frac{n}{2}$ exists and $f''(x) = \frac{2n}{3} > 0\ \forall n \geq 2$.

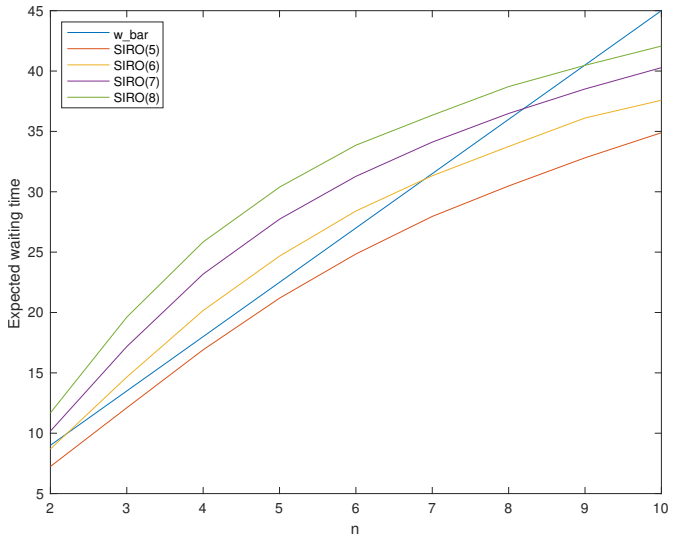


Figure 8: Comparison of willingness to wait $\bar{w}$ and the expected waiting time of the last agent, who joins a SIRO buffer-queue for different caps on the buffer-queue (cap in brackets). Waiting costs are $c = 0.1 \cdot \frac{2}{n}$ and mismatch value $x = 0.1$.

which is independent of the amount of different items n. The reason for this is that the costs decrease with the same rate as the probability of a favorite item becoming available. Thus, as the waiting costs remain the same, for FCFS the maximum IC buffer-queue length remains the same as well.

When looking at the impact of the different amounts of item types n for different mismatch values x at the same time, we find that for higher mismatch values, the shape of FCFS' welfare loss remains the same throughout, while for SIRO it takes longer until the WFL decreases towards zero. We report the this in Figure 14 in the Appendix. Our reasoning for the FCFS mechanism remains the same. The behavior of SIRO's WFL can be explained by the fact that higher mismatch values x, decrease the willingness to wait $\bar{w}$, i.e. decrease $\bar{w}$'s slope in Figure 8. The convex shape of SIRO's maximum IC buffer-queue lengths K is then the reason for why SIRO's WFL only decreases towards zero for higher n.

4.2 Heterogeneous mismatch values

As the second extension of our model, we will relax the assumption of homogeneous mismatch values. Again, we would like to investigate the impact of our extension on WFL and specifically compare the impact on the FCFS and the SIRO mechanism.

4.2.1 Implementation

To implement heterogeneous mismatch values, we alter equation (1), in which we specified the value v that an agent derives from an item. We now assume that the mismatch value x is drawn from a uniform distribution between 0 and 1 and that all draws are independent across agents as well as item types (i.e. an agent has the same valuation for items of the same type).[19]

[19] Note that Leshno (2017) assumes heterogeneous mismatch values for items of the same type. We, however, decided that for our application to the housing assignment, the characteristics of items of the

Additionally, we have to adapt Definition 3, in which we defined the long-run WFL, to the heterogeneous mismatch values, because the WFL is now dependent on the exact mismatch values of the misallocated items.

Definition 8. For a mechanism μ, we define the long-run WFL as

$$\text{WFL}_\mu = \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} (1 - v_{kt})\xi_t \tag{10}$$

$$\text{with} \quad \xi_t = \begin{cases} 1 \text{ if } \mu_t(h_k) = i \text{ is a mismatch} \\ 0 \text{ otherwise,} \end{cases}$$

where v_{kt} denotes the value of the available item h_k in period t and ξ_t is an indicator which equals 1 if the item is misallocated by $\mu_t(h_k)$ and 0 otherwise.

Note that the SIRO mechanism was not specifically made for this setting, because heterogeneous mismatch values might result in different IC buffer-queue lengths K for different agents and different item types. Therefore, we cannot use the dynamic equations anymore to compute the agents' expected waiting time as explained in section 3.2.2. As an alternative, however, we can implement that agents form their expectations based on the waiting time of previous agents.

When agents have to make a decision between accepting an item or joining the buffer-queue for their favorite item, the mechanism will provide them with information about the average waiting time of agents, who joined the buffer-queue in the position that they are offered. Comparing this to their own willingness to wait enables the agents to make their decision. To really compare the impact of the two mechanisms, we also assumed for the FCFS mechanism that the agents learn from the previous waiting times.[20] Note that in a balanced economy, the waiting time in each position of a buffer-queue is the same among all buffer-queues and that we can exploit this symmetry when implementing the learning.

That this implementation of learning converges to an equilibrium was shown by Altman and Shimkin (1997) under the assumption that there is a small fraction of agents that always joins the queue, independent of the expected waiting time. This assumption has to be implemented to ensure that agents continue to learn even in a case when the average waiting times have become extraordinarily high. We report an exemplary convergence path for the first position in a buffer-queue for the FCFS and the SIRO mechanism in Figure 9 and see that both mechanisms converge, but note that it takes more periods for the SIRO mechanism. We will elaborate on the impact of this assumption when elaborating on the results of the simulation.

4.2.2 Simulation

The results of our simulation are shown in Table 3.[21] We can see that as the SIRO mechanism decreases the WFL in comparison to FCFS for waiting costs up until $c = 0.30$.

same type should be evaluated similarly.

[20] As the expected waiting times under the FCFS mechanism are independent of how many agents join the buffer-queue afterwards, we could also compute the exact expected waiting times here. We could confirm that the expected waiting times with learning converged to the true values.

[21] Each value in the table is an independent simulation (i.e. 16 simulations) for the respective parameters of n and c as well as the mechanism used. Every simulation ran for two iterations, which are constituted of 1,000,000 periods each and the reported values are the average of the two independent iterations. Standard errors have been below 1% for WFL in each case.

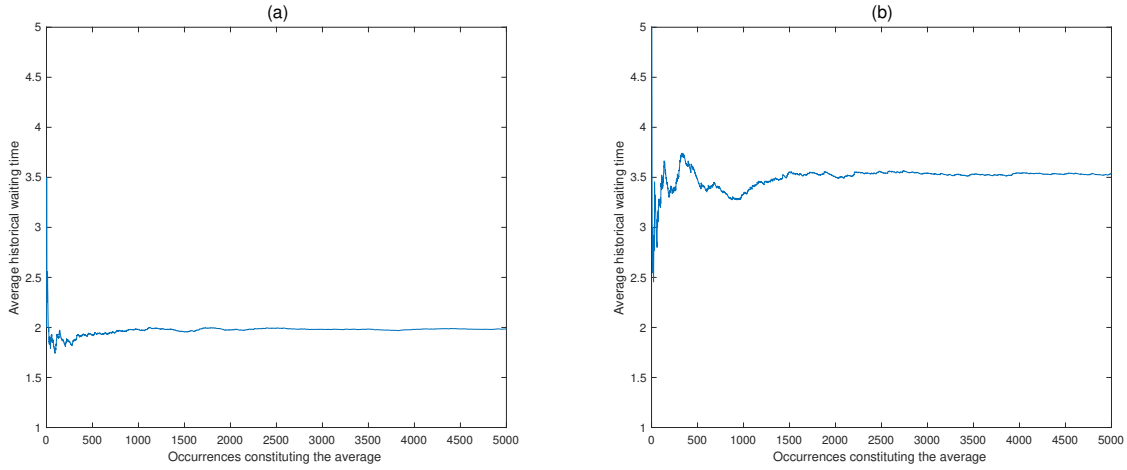


Figure 9: Convergence of the learning scheme. We show the convergence for position 1 in an (a) FCFS and a (b) SIRO buffer-queue. Waiting costs are $c = 0.05$.

	Mechanisms		
c	FCFS	SIRO	Difference in percent
0.05	2.49	2.21	−11.17
0.10	5.0	4.35	−13.11
0.15	7.44	6.54	−12.04
0.20	9.59	8.63	−10.05
0.25	12.44	10.88	−12.51
0.30	13.54	13.04	−3.68
0.35	15.09	15.25	1.01
0.40	17.47	17.51	0.2

Table 3: WFL in percent for heterogeneous mismatch values and different waiting costs c. In addition, we report the percentage difference in WFL that using SIRO instead of FCFS would result in.

For higher waiting costs, it performs slightly worse than FCFS.[22]

To understand why the benefit of using SIRO disappears for higher waiting costs c, it is instructive to look back at subsection 3.4. There we saw that for a generally high willingness to wait (i.e. low waiting costs) SIRO's impact on increasing the maximum IC buffer-queue length was high, whereas for a low willingness to wait, SIRO and FCFS had similar maximum IC buffer-queue lengths, and thus a similar WFL (cf. Figure 4). This argumentation applies in the setting of heterogeneous mismatch values as well.

However, it then appears puzzling that SIRO performs even worse than FCFS for high waiting costs. To understand why this happens, note that for waiting costs as high as $c = 0.35$ or higher, the maximum IC buffer-queue length of an FCFS buffer-queue will be $K = 1$.[23] For SIRO, however, it might for a small fraction of agents be IC to even accept positions in the buffer-queue when there is already one agent in it. While queuing in this situation decreases, it also increases the expected waiting time for the first position in the buffer-queue. The decrease in welfare loss by agents joining the second position, will then be counteracted by the agents, who observe a higher waiting time for taking the first spot in a buffer-queue and, therefore, reject the item. We did not see this phenomenon in the case of homogeneous mismatch values because the maximum IC buffer-queue length was the same for every agent.

Additionally, we observe that the WFL for both mechanisms increases for higher waiting costs. This is also in line with our results from previous sections, where we have seen investigated the sensitivity of the mechanisms with regards to higher costs (cf. subsection 3.4).

In order to implement that the agents could learn from the waiting times, we had to include a fraction of agents that would, if offered a mismatched item, always decide to join the queue for their preferred item independent of the expected waiting time. Having a fraction of always-queuers in the model decreases the WFL. This is because in situations when normal agents would have accepted a mismatch, the always-queuers would still join the queue.[24] We want to emphasize, however, that in our simulations only 1% of agents are always-queuers and, thus, that the impact is small.[25] Additionally, note that the always-queuers are implemented in both mechanisms.

4.3 Heterogeneous mismatch values in the n-item economy

As in reality, we are likely to encounter both, heterogeneous mismatch values as well as more than two different item types, we will in this section combine the two previous extensions.

[22] That we report the results in a table instead of a graph is due to the circumstances that computational efforts increase drastically when implementing the learning. Therefore, the fine increments in parameters needed for appropriate graphs would have taken an unreasonably long time to compute.

[23] The highest possible willingness to wait is $\frac{1}{c} = \frac{1}{0.35} \approx 2.86$ as $1 - x_k \leq 1$, which means that with $p_A = 0.5$, we get from equation 9 that $K_A = 1$. And similarly for B items.

[24] The impact of always-queuers becomes more noticeable for higher waiting costs, because higher waiting costs reduce the maximum IC buffer-queue length and always-queuers would, therefore, more often be in a position, in which normal agents would accept the mismatched object.

[25] When comparing our simulation results for FCFS to the theoretical results for $n = 2$ reported by Leshno (2017), we find that the differences are below 1%.

4.3.1 Implementation

Now, in addition to waiting for the favorite item type, we have the case that the agent could also wait for another non-favorite item type, which she values more.

Again, agents always prefer their favorite item type over waiting. For the case when agents are offered a mismatched item, however, we have to adapt Lemma 1 to the possibility of queuing for any other item.

Lemma 2. *An agent will decline a mismatched item, if her willingness to wait for another item of higher value is greater or equal to the respective expected waiting time. The agent will then join the buffer-queue for that item. If the condition is fulfilled by more than one item, the agent will join the buffer-queue for the item with greater difference between willingness to wait and expected waiting time.*

Proof. We have shown in Lemma 1 that comparing the value of accepting with the expected value of rejecting is equivalent to comparing the expected waiting time for a favorite item with the agent's willingness to wait. This can easily be adapted for the case of any other item by substituting 1, the value of the favorite item, by the mismatch value of an item h_j of type $g_j \neq g_k$, with g_k denoting the type of the available item h_k. If we let $E[t_j]$ denote the expected periods until an item of type g_j becomes available, we get that

$$\begin{aligned} x_k &\leq x_j - E[t_j] \cdot c \\ E[t_j] &\leq \frac{x_j - x_k}{c} \\ E[t_j] &\leq \bar{w}_j, \end{aligned} \tag{11}$$

where $\bar{w}_j$ is the agent's willingness to wait for an item of type j. The agent is only willing to wait for items of types that have a higher mismatch value than the available item (i.e. for which holds $x_j > x_k$).

Now we have established the agent's willingness to wait for each item, which the agent then compares to the expected waiting time. However, there could be multiple item types, for which equation (11) holds. The agent would then have to decide which buffer-queue to join. The agent derives the highest value from queuing for the item that has the highest difference between the expected waiting time and the agent's willingness to wait, which we will prove now.

Note that an agent's willingness for an item h_j gives us the amount of periods it takes until the accumulated waiting costs have gotten as high as the additional value from item h_j in comparison to the available item h_k. In other words, it gives us the additional value of h_j denoted in periods. If we now subtract the expected waiting periods for item h_j, we are left with the expected additional value of receiving item h_j. Naturally, if an agent compares the expected additional value of receiving different items, she will decide for the one with the highest expected additional value. □

To illustrate the agent's decision, we introduce the following example.

Example 8. Consider an economy with $n = 3$ different item types. In addition to the A and B type items and α and β agents, let there be a third item type C, which is preferred by a third agent type γ. Assume waiting costs are $c = 0.1$.

Suppose an A item arrives and that the buffer-queue for A items is empty. The item is then offered to the first agent on the general waiting list, who is of type β and values the

	Difference in percent			
$\hat{c}$	$n = 2$	$n = 3$	$n = 5$	$n = 10$
0.05	−11.17	−13.81	−16.79	−21.43
0.10	−13.11	−13.25	−15.01	−17.29
0.15	−12.04	−12.43	−12.39	−13.47
0.20	−10.05	−10.04	−10.04	−9.5
0.25	−12.51	−11.53	−8.44	−4.61
0.30	−3.68	−2.94	−1.83	−0.77
0.35	1.01	0.36	0.34	1.06
0.40	0.2	1.8	0.78	1.44

Table 4: WFL difference between the FCFS and the SIRO mechanism in the n-item economy with heterogeneous mismatch values. Waiting costs for simulation are $c = \hat{c} \cdot \frac{2}{n}$.

A item at $x_A = 0.1$. The agent's willingness to wait for a B item is then $\bar{w}_B = \frac{1-0.1}{0.1} = 9$ periods. Additionally, the agent values type C items at $x_C = 0.6$, which results in a willingness to wait for type C items of $\bar{w}_C = \frac{0.6-0.1}{0.1} = 5$.

Assume now that the expected waiting time for both items is 3. The differences between the willingness to wait and the expected waiting time would then be $\bar{w}_B - E[t_B] = 6$ and $\bar{w}_C - E[t_C] = 2$ and the agent would decide to queue for type B items.

Alternatively, assume that the expected waiting time had been $E[t_B] = 8$ and $E[t_C] = 3$. Then the differences would be $\bar{w}_B - E[t_B] = 1$ and $\bar{w}_C - E[t_C] = 2$, which would mean that agent would queue for type C items.

When computing the WFL, we can use Definition 8 from the previous section. We also adapt the waiting costs from 4.1.2, as we want to keep the waiting costs for the next item constant (i.e. $c = \hat{c} \cdot \frac{2}{n}$). Additionally, both mechanisms will still proceed as presented earlier. Again, we can, therefore, draw conclusions about SSSB's mechanism by looking at the FCFS mechanism's allocation.

4.3.2 Simulation

In Table 4 we report the difference in percent between the FCFS and the SIRO mechanism analogous to the last column in Table 3. We do so as our focus lies on investigating the difference between the two mechanisms rather than the magnitude. For the underlying table, in which we report the magnitude, please refer to Table 7 in the Appendix.[26]

Similarly to our previous results in the 2-item economy, we see for each specification of the amount of item types that SIRO performs better than the FCFS mechanism as long as waiting costs are not too high. For high waiting costs ($\hat{c} = 0.35$ and $\hat{c} = 0.40$) we observe that the FCFS mechanism results in slightly less welfare loss than SIRO. Our explanation for the 2-item economy for this phenomenon also remains valid in the n-item economy.

Through the extension to an n-item economy, we can now examine how more item types change the mechanisms' performances. We can see that for waiting costs of $\hat{c} = 0.10$

[26] Our results are based on 64 independent simulations for the different parameters of n and c as well as the mechanism used. Otherwise the same holds as reported in footnote 21.

and $\hat{c} = 0.05$ the benefit of using the SIRO mechanism increases with the number of different item types, whereas for waiting costs of $\hat{c} = 0.25$ and $\hat{c} = 0.30$ we observe the opposite. For the waiting costs in between, we see that for fewer item types the difference between FCFS and SIRO remain roughly constant, whereas for $n = 10$ the difference between both mechanism decreases in $\hat{c}$.

We can relate these results well to the subsection in which we introduced the n-item economy for homogeneous mismatch values (cf. 4.1). There we also observe a significant decrease in SIRO's WFL for more item types and $\hat{c} = 0.1$ (cf. Figure 7). We reasoned for this by elaborating on the relation between the agents' willingness to wait and the maximum IC buffer-queue lengths. We then concluded that if agents' willingness to wait decreases, we will observe a decrease in SIRO's welfare loss only for more different item types (cf. Figure 14). This is due to the relation between willingness to wait and expected waiting time under SIRO, which we depicted in Figure 8.

Consequently, we would expect that we see a positive effect of SIRO if we consider more types of items, even for moderately high waiting costs. However, the computational effort increases with more item types, as agents can join more queues and will check all queues to find out which one yields them the highest expected value. In selective simulations with $n = 25$ item types, we could not confirm the expected effect,[27] but suggest investigations for higher amounts of different item types.

We conclude that much of our analysis from the individual examinations of the extensions also hold for the combined setting: For all amounts of item types considered, SIRO performs better for lower waiting costs, but the benefit of SIRO does not exist for high waiting costs. Additionally, we note that for low waiting costs, the benefit of using SIRO increases with more item types. We could not conclusively answer whether we would see a higher difference between SIRO and FCFS for much higher amounts of different items and note that this warrants further investigation, ideally with access to more computing power.

4.4 Imbalanced economy

Throughout the model, we have considered a balanced economy, where the long-term demand for the different item types is the same as the respective long-term supply. While it could be argued that SSSB strives to fulfill the students' demands, the building of new housing takes time and is constraint by available locations as well as money. It is therefore likely that some apartment types are more demanded than others. Investigating how such imbalances in the long-term supply and demand affect the allocation of items is the aim of this extension.

4.4.1 Implementation

We will now implement the imbalanced economy into the 2-item economy, where agents have heterogeneous mismatch values.

In order to do so, we implement that agents are of type α with a different probability than $p_\alpha = 0.5$, while items still arrive with probabilities $p_A = p_B = 0.5$. Changing the arrival probability of α agents directly affects the arrival probability of β agents, as $p_\alpha = 1 - p_\beta$. Note that in the simulations we cannot exploit the symmetry between the

[27] For $\hat{c} = 0.2$, our results were $\text{WFL}_{FCFS} = 12.21\%$ and $\text{WFL}_{SIRO} = 11.69\%$; for $\hat{c} = 0.3$, our results were $\text{WFL}_{FCFS} = 19.42\%$ and $\text{WFL}_{SIRO} = 19.92\%$.

	Agent arrival probability (p_α, p_β)				
c	(0.5, 0.5)	(0.4, 0.6)	(0.3, 0.7)	(0.2, 0.8)	(0.1, 0.9)
0.05	−11.17	−11.60	−10.65	−9.66	−8.84
0.10	−13.11	−12.32	−11.71	−11.25	−10.39
0.15	−12.04	−12.04	−11.38	−10.72	−10.07
0.20	−10.05	−9.75	−9.37	−9.00	−8.33
0.25	−12.51	−12.57	−12.07	−11.55	−10.98
0.30	−3.68	−4.10	−3.85	−3.57	−3.51
0.35	1.01	0.93	0.69	0.65	0.58
0.40	0.2	0.58	0.52	0.40	0.57

Table 5: WFL difference (in percent) between the FCFS and the SIRO mechanism in the 2-item economy for different specifications of imbalance.[29]

queues any longer if we are in an imbalanced economy, because the waiting times for the positions in the buffer-queues change with the agent's arrival probabilities.

The reason for why we implement the imbalance in the 2-item economy is that it allows us to investigate multiple degrees of imbalance. If the demand for one item is higher than its supply, we can directly infer that the demand for the other item is lower than its supply. Otherwise, if we were in a 3-item economy, and altered the demand for one item type, information about the other 2 items cannot directly be deduced, but their exact arrival probabilities determine the resulting WFL. Nevertheless, based on our results from the 2-item economy, we will argue for the impact of imbalance in the n-item economy in the next section.

4.4.2 Simulation

The results are reported in Table 5.[30] We observe the familiar result that also under different imbalance settings, the SIRO mechanism performs better for low waiting costs, but not for high waiting costs. Additionally, we see that with increasing imbalance, the difference between both mechanisms decreases. This is happens, because the much demanded item will cause a long, queue, which disproportionally many agents join. Expected waiting times for positions in the buffer-queue, therefore, increase with more imbalance and cause more mismatches for SIRO, whereas expected waiting times for positions in an FCFS buffer-queue remain the same. Note as well, that with increasing imbalance, the magnitude of the WFL increases for both mechanisms (cf. Table 8 in the Appendix).

While we could evaluate the impact of imbalance in the 2-item setting, this proves to be more difficult in the n-item economy. Not because imbalance cannot be introduced into this setting, but rather because it is difficult to make the welfare loss for the different specifications comparable. Still, based on our results, we can make some inferences for the n-item economy.

The increase in both mechanisms' welfare loss is something we will also observe if we introduce imbalance in the n-item economy. Consider the case when there are two

[30] Refer to Table 8 in the Appendix for the underlying values. The results are based on 80 independent simulations for the different parameters of p_α and c as well as the mechanism used. Standard errors have been below 0.1% for WFL in each case. Otherwise the same holds as reported in footnote 21.

items, which are imbalanced, whereas the long-term supply equals the long-term demand for remainder of the n items. For both mechanisms, the impact of the imbalance will decrease with more different item types, because the impact of the imbalanced types will decrease in comparison to the balanced item types. If we now assume that another two items are similarly in imbalance, while the remaining items are balanced, the proportion of imbalanced item types increases and so will the WFL.

We conclude that in the 2-item setting, a higher imbalance between both items results in a higher WFL for both mechanisms than in comparison to the balanced economy. Further, we again see that SIRO outperforms the FCFS mechanism for low waiting costs, but loses its advantage for higher waiting costs. From our results for the 2-item setting, we conclude that imbalance also negatively impacts the WFL, but that the exact WFL is dependent on the exact supply and demand for each item.

5 Discussion

The purpose of this section is to discuss SSSB's mechanism design, our assumptions, the fairness of the mechanisms as well as limitations of our model and recommendations.

5.1 Practical reasons for SSSB's mechanism design

In section 2.2 we have shown that the allocation with SSSB's mechanism results in the same allocation as an FCFS mechanism. It seems natural to investigate the question what the potential reasons could have been when designing SSSB's mechanism to not implement an FCFS mechanism instead.

One major difference between SSSB's mechanism and an FCFS mechanism is that the former just has to select the applicant with the highest credit days among the applicants, whereas the latter has to approach at least one, if not multiple agents to assign an item. In practice having to reach out to one or potentially even multiple students and awaiting their response is likely to significantly slow down the assignment process. That SSSB penalizes the rejection of an apartment, once it has already been assigned to a student, is also likely to have been implemented to counteract a slowing down of the assignment process.[31]

Another difference is that agents have no information on their rank in SSSB's mechanism, but only have information about their credit days as well as how many credit days the applicant with the highest amount for an item has. In the FCFS mechanism, the agents know the true expected waiting time for their favorite item when they have to decide whether to accept or decline a mismatched item. For SSSB's mechanism we assumed that agents form expectations based on the observations of the highest applicants' credit days, which then converge to the real amount of credit days needed. Thus, both mechanisms are equivalent with regards to the expected waiting time. Note, however, that if students' expectations are incorrect, there exists the possibility that students take decisions that are not optimal for them.

However, informing the students about their rank might also not be informative of their expected waiting time as there might be other students which have a higher position, but are not actively looking for accommodation anymore. Knowing the credit days, thus,

[31] Note that we do not have to take the punishments into account in our model, because we model the agent's decision so that an agent only applies to an item if she would accept it.

might give a better approximation of how much longer the agent will have to wait for a certain type of housing. Additionally, it seems worth mentioning that having the credit days build up instead of seeing a high queuing position might be more motivating for the students. Nothing is won if a student sees her position and gives up on trying to receive student housing via SSSB.

We conclude that, while there is room for improvement of the overall allocation, SSSB's mechanism has features that increase the speed of the assignment process in comparison to an FCFS with declines mechanism. Additionally, it is possible to make an argument for credit days being more informative than rank in SSSB's mechanism.

5.2 Impact of main assumptions

Now we turn to the assumptions we have made in our model and shed light on the underlying reasons as well as elaborate on their impact for the allocation.

5.2.1 Overloaded waiting list

We have made the assumption that the mechanisms can at all times allocate an available item, due to the high amount of agents in the queue. SSSB states in their 2017 annual report that of there was a roughly constant number of 50,000 persons in the queue during the year, of which 21,000 were eligible to receive student housing (SSSB, 2018, p. 22). The discrepancy between total number of persons in the queue and eligible ones results from the possibility of queuing before starting one's studies and that credit days can be parked for 36 months, while not being affiliated with one of the SSCO's member student unions (e.g. after graduation). The high demand for student housing via SSSB is not surprising as the average rental prices are the highest in the Greater Stockholm area compared to the rest of Sweden (Statistics Sweden, 2018). At the end of 2017, SSSB owned 7,985 apartments and reported 3,532 newly signed tenancy agreements during the year (SSSB, 2018, p. 3). Comparing the number of new tenancy agreements with the high number of persons queuing, we arrive at the conclusion that describing SSSB's housing queue as an overloaded waiting list is a reasonable assumption.

The benefits of an overloaded waiting list are that we can abstract from an arrival process of the agents to the queue and, thus, can compare the welfare of different mechanisms more easily. This is for the reason that the overall waiting time will always be the same, independent of who gets assigned to the object. In any case, after an object is assigned, there will be one less agent waiting. Therefore, we only have to consider the welfare loss incurred through agents not having been assigned to their preferred object (Leshno, 2017).

A necessary assumption for assessing the welfare in such a way is that all agents incur the same waiting costs, which in reality might not hold true. If agents incurred heterogeneous waiting costs, the welfare loss would be minimized by minimizing the welfare loss through misallocation and preferring agents with high waiting costs for an assignment, independent of how long they have waited. However, a practical concern would be how to adequately capture the waiting costs. If those costs could not be captured adequately, basing preferential treatment on them is likely to be perceived as unjust by agents with low waiting costs, who would be adversely affected by the preferential treatment of agents with high waiting costs. Assuming the same waiting costs appears to be a feasible way of

dealing with not having a reliable way to adequately capture waiting costs in practice.[32]

5.2.2 Differentiating characteristic for the buffer-queues

An underlying assumptions to be able to implement the SIRO mechanism are that there are two different types of items and that each agents has a favorite item type. In order to implement the SIRO buffer-queue mechanism for the Stockholm student housing market, we have to decide on the characteristic that should be used to differentiate the houses from one another.

Rental price comes to mind, but bears problems. While the housing objects could be differentiated by the rental price and agents have different amounts of money to spend on housing, all agents would prefer to spend as little as possible. Additionally, SSSB's goal is to provide students in Stockholm with "prisvärt" (SSSB, 2018, p. 42) housing, which translates to inexpensive or good value for money. Considering the comparably high rental prices in Stockholm (Statistics Sweden, 2018) makes it reasonable to assume that students who have the option to rent accommodation via SSSB will save money.

A better distinguishing factor is location. SSSB divides its housing areas into three different clusters: North of the city, South of the city and city center. Therefore, a feasible distinguishing factor for the two-item economy would be to denote housing objects outside of the city center as one type and housing objects in the city center as the other type. For the agents that implies that one agent type prefers living in the city center, while the other prefers living outside of the city center.

Still, the division into two different types of housing objects might leave out other important characteristics of the housing objects. Therefore, in section 4.1, we elaborated on an extension of the model to an n-item case and have seen that SIRO still outperforms the FCFS mechanism as long as agents are not very impatient. If agents are more patient, the benefit from using SIRO even increases in the amount of different item types. When we investigated the n-item economy wiht heterogeneous mismatch values, these results remained valid.

Practically, it could, for example, be feasible to differentiate between the 26 different housing areas where SSSB rents out housing objects (see Appendix A for a full list) or the different amount of rooms of an apartment (SSSB offers one, two, three and four-room apartments). Even a combination of both could be implemented, so that housing objects would be differentiated by area as well as room count, resulting in $26 \cdot 4 = 104$ different types.

Overall, with the extension to the n-item economy, it possible to implement the SIRO mechanism in more complex settings. Our suggestion to implement the differentiating buffer-queue characteristics based on location and potentially another factor appears to be a feasible solution. While admittedly the differentiating characteristics are somewhat limiting, we are of the opinion that for the sake of a better allocation of housing, this is a trade-off worth considering.

[32] Note that SSSB's priority treatment (cf. section 1.2) is basically a preferential treatment for agents who are deemed to have exceptionally high waiting costs.

5.3 Fairness

We will take this subsection to discuss fairness considerations of SSSB's and the SIRO mechanisms.[33]

Fairness in our case can be described as a moral assessment regarding the allocation or the mechanism, often referred to as distributive justice in the literature. Consequently, there exist several theories of what is to be regarded as a morally just distribution.[34] While the philosophical questions of distributive justice are certainly interesting, a specific examination is only of limited relevance for this thesis. For the purposes of this thesis, it is most instructive to follow Haidt (2013), who distinguishes between two main types of distributive fairness: equality and proportionality. In the following, we will examine the allocation and mechanisms with regards to those two types.

Proponents of the maxim of equality assess an allocation as fair if everyone receives an equal share. For two reasons we cannot transfer this to our problem: First, our items are indivisible and there exist less items than agents. Second, we are considering a dynamic matching problem, where not all items are available from the start, but arrive over time.

Another possible aspect of equality is the equality of chances, a demand often made in connection with education. Both, SSSB's mechanism as well as the SIRO mechanism are impartial, because the mechanisms' rules are clearly specified and the allocation is not subject to preferential treatment. If we consider two agents that join the queue at the same time, both have the same chances. However, under the SIRO mechanism, as Leshno (2017) argues, agents have a more equal expected waiting time for their favorite item than under the FCFS (and, thus also SSSB's) mechanism. Still, the author acknowledges a higher variation in the agents' actual waiting times.

If two agents join the same queue at different points in time, though, their chances of receiving an item are different as well. This holds true for SSSB's and the SIRO mechanism and using time as a differentiating factor could be questioned when following the maxim of equality. Still, using time as a distinguishing factor seems to be regarded as fair or at least acceptable, considering the popularity of first come first served in our society (e.g. being served in a fast-food restaurant, buying tickets for events or making appointments at the doctor). We will elaborate more on time as a distinguishing factor, when discussing the second type of fairness next.

Proportionality describes an allocation that gives an agent a proportion of what is to be allocated, based on certain characteristics. These characteristics, however, can be manifold. An often used characteristic for argumentations in favor of this view on fairness is effort. Those who exert more effort should also receive more.

For sequential allocations of one item per agent, the maxim of proportionality can be used to determine an order in which picks can be made. All three mechanisms (SSSB, FCFS and SIRO) we have introduced in this thesis use arrival time to the list as a deciding factor to determine the order of picks.[35] As mentioned earlier, the popularity of using time as a distinguishing factor suggests the social acceptance of it. Still, the appropriateness in

[33] Note, that we abstract from the additional rules that SSSB follows, as we did not derive such rules for the SIRO mechanism and as the main goal of this section is a comparison of both mechanisms.

[34] The interested reader is referred to Lamont and Favor (2017) for an overview of different theories regarding distributive justice and to Thomson (2011) for a review of fair allocation rules in various economic settings.

[35] While FCFS and SIRO right away determine the order of the general waiting list based on the arrival to the queue, SSSB first checks which agents are interested, but then also uses the time spent queuing as a deciding factor for who receives the item (cf. 2.2.2).

our setting could be questioned: Whereas in physical settings an agent has to be physically present for the duration of waiting in the queue, queuing online as implemented by SSSB's only requires the agent to sign up. In the physical setting agents who value the reward for queuing more, are more willing to stand in the queue for a longer time. However, due to the unknown and unsteady time intervals that it takes until an apartment becomes available, physical queuing is not a feasible solution for SSSB's housing queue in reality.

Another possible characteristic for the proportionality could be neediness. It could be argued that a fair allocation is one that gives the item to the agent, who needs it the most, or gives that agent the first pick in a dynamic allocation. For a need-based order of picks it would still be necessary to specify what is meant with need. Possible specifications for need could, for example, be monetary constraints (of parents) or moving to Stockholm from another city.

To conclude this subsection, we note that neither SSSB's nor the SIRO mechanism is superior to the other from a fairness perspective. That we do not get a clear result is hardly surprising given the moral-based notion of fairness. Still, including a discussion of fairness highlights the importance of distributional justice in this setting and aims to foster understanding for different perspectives on the topic.

5.4 Limitations

For a full assessment of our analysis, we will now turn to the limitations of our model.

We assumed that one item arrives per period and that the period ends with the assignment of the available item. In reality, however, it is the case that multiple items can be available at the same time. This would mean that agents can make use of applying to multiple items at the same time, which introduces another strategic aspect into the allocation, because other agents might apply for multiple items as well. As each agent can only receive one item, being the agent with the second-most points for an item could still lead to receiving the item. While these strategic considerations are tough to incorporate in a model as information for agents is limited (i.e. only credit days of highest applicant known), they are of relevance in reality.

Another aspect that is not considered in our model is that the students are studying in programs with different lengths. The study length might impact the agents decision through the estimated time that a student will still be in Stockholm and actually make use of the apartment. Still, it is not obvious how this impacts an agents decision. Consider, for example, a bachelor student who also wants to do a masters and, thus, is planning on staying in Stockholm for at least five years. It is not obvious whether that bachelor student would prefer waiting a long time for her favorite apartment or rather try to receive one as soon as possible to make use of it for a longer period.

Further, we acknowledge that the SIRO mechanism comes with the limitation that a mechanism designer has to decide on one ore more measurable characteristics to classify the apartments. This in turn limits the students' choices to the different types that the mechanism designer selected. However, by introducing the extension to the n-item economy, we lessened the impact of this limitation.

6 Conclusion

Motivated by the tense housing market in Stockholm and the seemingly complex allocation procedure of SSSB, the biggest provider of student housing in Stockholm, this thesis set

out to investigate the student housing allocation in the light of recent contributions to the dynamic matching literature.

Using Leshno's (2017) paper as a foundation, we set up a 2-item model with an overloaded waiting list and showed that the current allocation mechanism by SSSB is outcome-equivalent to an FCFS mechanism with declines. After explaining the concept of buffer-queues and that the FCFS mechanism could be presented as a buffer-queue as well, we introduced Leshno's SIRO mechanism, a robust mechanism that increases welfare by incentivizing agents to decline non-favorite items. In comparison to FCFS, agents are more likely to decline non-favorite items under the SIRO mechanism, because they have higher chances of receiving their favorite item in later positions in the buffer-queue (and thus a higher expected value). We investigated the sensitivity of both mechanisms to various changes of parameters and conclude that SIRO performs strictly better for most parameters and otherwise still weakly better than FCFS.

Subsequently, we introduced extensions to our model and thoroughly examined the underlying reasons for the differences that we observed in the mechanisms' performances. Our conclusion is that the SIRO mechanism still outperforms the FCFS mechanism, even in more complex settings such as an imbalanced n-item economy in which agents have heterogeneous mismatch values. However, the benefits from using SIRO decrease with higher waiting costs.

In addition to our theoretical analysis, we provided a more practical assessment of SSSB's mechanism design and reconciled our assumptions with reality. We also discussed fairness considerations and provided an overview about limitations of our model, which suggests that further research should investigate additional strategic aspects through overlapping availability of objects or time left of making use of student housing. Additionally, research on students' willingness to wait should be considered. We note further that the setting of an overloaded waiting list appears to be a promising direction for future work in dynamic matching.

In summary, the introduced SIRO mechanism outperforms FCFS in most settings and stands out due to its straightforward description and robustness. We believe that implementing the SIRO mechanism for the allocation of student housing in Stockholm could contribute to a better allocation.

References

Abdulkadiroğlu, A. and Sönmez, T. (2003). School choice: A mechanism design approach. *American economic review*, 93(3):729–747.

Akbarpour, M., Li, S., and Gharan, S. O. (2014). Dynamic matching market design. *CoRR*, abs/1402.3643.

Altman, E. and Shimkin, N. (1997). Learning and equilibrium in processor sharing systems. techreport CC212, Department of Electrical Engineering, Technion.

Baccara, M., Lee, S., and Yariv, L. (2018). Optimal dynamic matching. Discussion Paper DP12986, Centre for Economic Policy Research.

Baliga, S. and Maskin, E. (2003). Mechanism design for the environment. In *Handbook of environmental economics*, volume 1, pages 305–324. Elsevier.

Bergemann, D., Brooks, B. A., and Morris, S. (2016). Informationally robust optimal auction design. Discussion Paper 2065, Cowles Foundation.

Bloch, F. and Cantala, D. (2017). Dynamic assignment of objects to queuing agents. *American Economic Journal.Microeconomics*, 9(1):88–122.

Budish, E. (2012). Matching versus mechanism design. *ACM SIGecom Exchanges*, 11(2):4–15.

Dasgupta, P. and Maskin, E. (2000). Efficient auctions. *The Quarterly Journal of Economics*, 115(2):341–388.

Gale, D. and Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15.

Haidt, J. (2013). Of freedom and fairness. *Democracy*, (28).

Hassin, R. and Haviv, M. (2003). *To queue or not to queue: Equilibrium behavior in queueing systems*, volume 59. Springer Science & Business Media.

Hurwicz, L. (1960). Decentralization and computation in resource allocation processes. In Kenneth J. Arrow, S. K. and Suppes, P., editors, *Mathematical methods in the social sciences*. Stanford Univ. Press.

Hurwicz, L. (1972). On informationally decentralized systems. In McGuire, C. B. and Radner, R., editors, *Decision and Organization: A Volume in Honor of J. Marschak*, pages 297 – 336. North-Holland Publishing Company.

Hylland, A. and Zeckhauser, R. (1979). The efficient allocation of individuals to positions. *Journal of Political economy*, 87(2):293–314.

Kaplan, E. H. (1987). Tenant assignment policies with time-dependent priorities. *Socio-Economic Planning Sciences*, 21(5):305–310.

Lamont, J. and Favor, C. (2017). Distributive justice. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2017 edition.

Leshno, J. (2017). Dynamic matching in overloaded waiting lists. Revise and Resubmit, Econometrica.

Maskin, E. and Sjöström, T. (2002). Implementation theory. In Arrow, K. J., Sen, A. K., and Suzumura, K., editors, *Handbook of Social Choice and Welfare*, volume 1 of *Handbook of Social Choice and Welfare*, chapter 5, pages 237–288. Elsevier.

Niederle, M., Roth, A. E., and Sönmez, T. (2008). Matching. In Durlauf, S. N. and Blume, L. E., editors, *The New Palgrave Dictionary of Economics*. Palgrave Macmillan, 2 edition.

Roth, A. E. (1984). The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of political Economy*, 92(6):991–1016.

Roth, A. E., Sönmez, T., and Ünver, M. U. (2004). Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488.

Shapley, L. and Scarf, H. (1974). On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37.

Sönmez, T. and Ünver, M. U. (2011). Matching, allocation, and exchange of discrete resources. In *Handbook of social Economics*, volume 1, pages 781–852. Elsevier.

Statistics Sweden (2018). Minor rent change in dwelling stock.

Stockholms Studentbostäder (2012). Instruktion för studentbostadskön och innehav av studentbostad.

Stockholms Studentbostäder (2018). Årsredovisning 2017.

Stockholms Studentbostäder (n.d.). About us — SSSB. Retrieved from: `https://www.sssb.se/en/about-sssb/`. Accessed: 2019-05-09.

Stockholms studentkårers centralorganisation (n.d.). About SSCO — SSCO. Retrieved from: `http://www.ssco.se/en/om-ssco/`. Accessed: 2019-05-09.

Thakral, N. (2016). The public-housing allocation problem. Technical report, Harvard University.

Thomson, W. (2011). Chapter twenty-one - fair allocation rules. In Arrow, K. J., Sen, A., and Suzumura, K., editors, *Handbook of Social Choice and Welfare*, volume 2 of *Handbook of Social Choice and Welfare*, pages 393 – 506. Elsevier.

Ünver, M. U. (2010). Dynamic kidney exchange. *The Review of Economic Studies*, 77(1):372–414.

Appendix

A Housing areas

SSSB categorizes their 26 housing areas into three different clusters, which we report in the following table.

North of the city	City center	South of the city
Freja	Apeln	Skrmarbrink
Frsunda	Domus	Frigg Sickla
Kungshamra	Forum	Balder
Lappkrrsberget	Fyrtalet	Embla
Pax	Hugin & Munin	Flemingsberg
Strix	Idun	
	Jerum	
	Kurland	
	Lucidor	
	Mjlner	
	Nyponet	
	Roslagstull	
	Vtan	
	Marieberg	
	Tanto	

Table 6: Areas of SSSB housing as shown on `https://www.sssb.se/en/our-housing/`.

B Markov Chain

In this section, we will present background on the Markov Chain, based on Leshno (2017).

A Markov Chain describes a system that can only be in a finite amount of different states, in which the next state only depends on the current state and where we have stochastic transitions. This captures the properties of the buffer-queues in the 2-item economy very well (cf. Remark 7).

Assume a buffer-queue that is capped at a maximum of K agents. Then we can describe the set of possible states as $S = \{-K_\beta, ..., -1, 0, 1, ..., K_\alpha\}$, where $k \geq 0$ denotes the amount of agents in the buffer-queue for A items and $k \leq 0$ the amount of agents in the buffer-queue for B items.

Leshno (2017) first establishes the probabilities to end up in a certain state, depending on the starting state. Then, however, he introduces an alternative description of the Markov Chain, in which all moves are between adjacent states, for which he notes the

following transition probabilities.

$$
\begin{aligned}
P(u|(k,\phi)) &= \begin{cases} p_A & u=(k-1,\phi) \\ p_B & u=(k,B) \end{cases} && k>0 \\
P(u|(-k,\phi)) &= \begin{cases} p_A & u=(-k,A) \\ p_B & u=(-(k-1),\phi) \end{cases} && k>0 \\
P(u|(0,\phi)) &= \begin{cases} p_A & u=(0,B) \\ p_B & u=(0,B) \end{cases} \\
P(u|(k,B)) &= \begin{cases} p_\alpha & u=(k+1,B) \\ p_\beta & u=(k,\phi) \end{cases} && 0 \le k < K_\alpha \\
P(u|(K_\alpha,B)) &= p_\alpha + p_\beta \quad u=(K_\alpha,\phi) \\
P(u|(k,A)) &= \begin{cases} p_\alpha & u=(k,\phi) \\ p_\beta & u=(-(k+1),A) \end{cases} && -K_\beta < k \le 0 \\
P(u|(K_\alpha,A)) &= p_\alpha + p_\beta \quad u=(-K_\beta,\phi).
\end{aligned}
$$

The first three equations assign arriving items or transition to another state that checks the type of a new agent, depending on the type of item that becomes available and the state of the buffer-queues, whereas the last 4 equations check the type of a new agent and either add her to the buffer-queue or assign her the available item and transition back to one of the first three states.

Based on the transition probabilities Leshno (2017) derives the stationary distribution, i.e. the long-run distribution, of the Markov Chain, which he then uses to derive the theoretical misallocation rate, which we report in equation (6). For a more detailed treatment of the used Markov Chain and the formal derivation of the theoretical misallocation rate, we refer to Leshno's Appendix A.

C Dynamic equations for the SIRO mechanism

C.1 2-item economy

We will in the following for the ease of explanation only consider the buffer-queue for A items. Note, however, that the description holds similarly for B items. The dynamic equations are based on Leshno (2017), who establishes them in a more general setting, whereas we will simplify them specifically for the SIRO mechanism. The Dynamic equations assume that all agents are truthful.

Under the SIRO policy, an agent joining a buffer-queue knows that everyone in the buffer-queue will get assigned an arriving item of their preferred kind with equal probability. The expected waiting time, however, also depends on how many agents will join the buffer-queue after the agent has joined. With a maximum number of agents that are allowed to join the buffer-queue, we have a finite amount of possible positions that an agent can be in. Let m be the amount of agents in the buffer-queue for A items. Then

we can describe each possible state in this SIRO buffer-queue with

$$\begin{aligned} W(m) &= p_B \cdot W^B(m) + p_A \cdot m^{-1} \cdot 0 \\ &\quad + p_A \cdot (1 - m^{-1}) \cdot (W(m-1) + 1) \end{aligned} \tag{12}$$

$$W^B(m) = \begin{cases} p_\beta \cdot (W(m) + 1) + p_\alpha \cdot W^B(m+1) & \text{for } m < K \\ W(m) + 1 & \text{for } m = K, \end{cases} \tag{13}$$

where $W(m)$ denotes the expected waiting time of an agent in the buffer-queue before an item arrives and $W^B(m)$ denotes the expected waiting time of an agent on the general list, who is offered a B item.

Equation (12) describes the expected waiting time of an agent in the buffer-queue for items of type A before an item arrives. With a probability of $1 - p_B$, the item is not of the favorite kind and the expected waiting time will then be $W^B(m)$. With a probability of p_A the an item of the favorite type arrives.

As the buffer-queue is operated under a SIRO policy, the probability of each agent in the queue to be assigned to the available item is m^{-1}. So with probability m^{-1} the agent has a waiting time of 0. With probability $1 - m^{-1}$ another agent in the buffer-queue will be assigned to the available item. This would result in an expected waiting time of $W(m-1) + 1$, as there is one less agent in the queue and the agent has to wait until the next item becomes available.

Equation (13) then describes the expected waiting time of an agent on the general list, who is offered a B item. If the buffer-queue is not at its maximum capacity, there is a probability of p_α that the next agent who the item will be offered to is an α agent and joins the buffer-queue as well. Additionally, there is a probability of p_β that the next agent is of type β, which means that the expected wait would then be the expected wait of someone in the queue once the next item has arrived.

Solving the dynamic equations for random start values gives us the expected waiting time for each position in the buffer-queue. In most cases, our interest is $W^B(K)$, the expected waiting time of the last agent that joined the buffer-queue. This values is the value that the function "fun_expected_wait_SIRO", which we show in Figure 2, gives back.

C.2 Adapted for n-item economy with homogeneous mismatch values

For the n-item economy with homogeneous mismatch values, we had to adapt the dynamic equations. With W^X denoting any other item not of type A, the dynamic equations become

$$\begin{aligned} W(m) &= (1 - p_A) \cdot W^X(m) + p_A \cdot m^{-1} \cdot 0 \\ &\quad + p_A \cdot (1 - m^{-1}) \cdot (W(m-1) + 1) \end{aligned} \tag{14}$$

$$W^X(m) = \begin{cases} (1 - p_A) \cdot (W(m) + 1) + p_\alpha \cdot W^X(m+1) + (1 - p_x - p_\alpha) \cdot & W^X(m) \\ & \text{for } m < K \\ W(m) + 1 & \text{for } m = K. \end{cases} \tag{15}$$

In equation (14), we substituted p_B with $(1 - p_A)$ for arrivals of any other item type than A. and p_β with $(1 - p_\alpha)$.

Similarly, we changed p_β to p_X in equation (15). The main difference to the 2-item economy is that we had to include $(1-p_x-p_\alpha)\cdot W^X(m)$, which accounts for the possibility that there might be agents arriving that neither prefer the item type that is offered, nor prefer item type A. Recall the assumption for the dynamic equations that all agents are truthful. Then, those agents will join another buffer-queue and we are still in the state, where an X item is going to be assigned to an agent W^X and no agents have joined the buffer-queue for A items.

D Additional Tables and Figures

D.1 Comparison of the FCFS and SIRO mechanism and sensitivity analysis

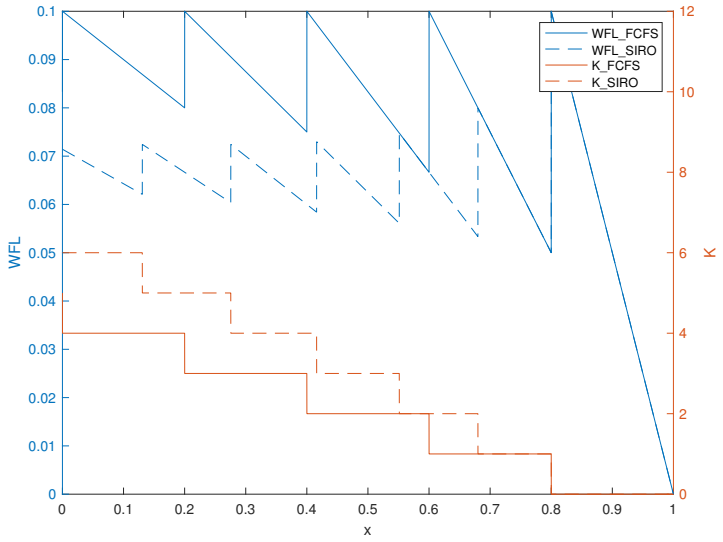


Figure 10: Comparison of the FCFS and the SIRO mechanism. On the left y-axis the WFl for different mismatch values. On the right y-axis we show the maximum IC buffer-queue lengths for the different mismatch values. Other parameters are $p_A = p_\alpha = 0.5$ and $c = 0.1$.

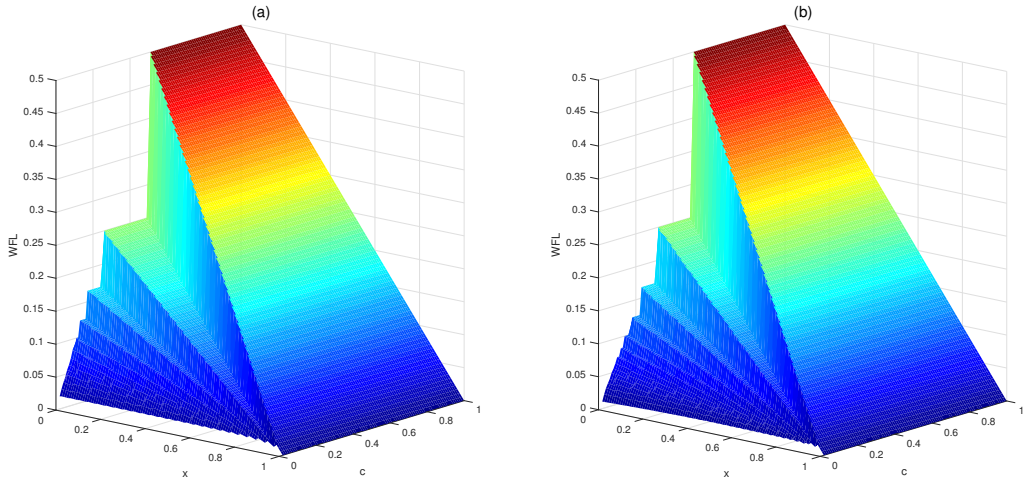


Figure 11: WFL for different mismatch values and waiting costs for the (a) FCFS and (b) SIRO mechanism. Other parameters are $p_A = p_\alpha = 0.5$.

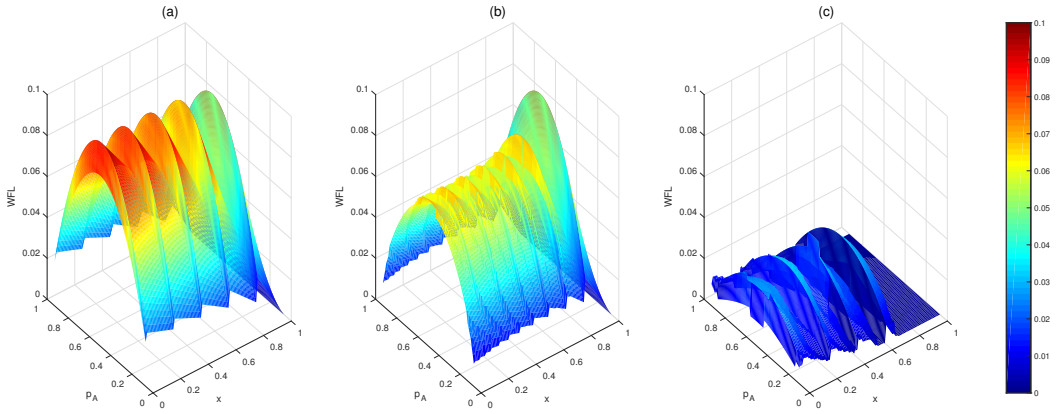


Figure 12: (a) WFL of FCFS, (b) WFL of SIRO and (c) difference in WFL between both mechanisms for different arrival probabilities and mismatch values. Waiting costs are $c = 0.2 \cdot p_A$.

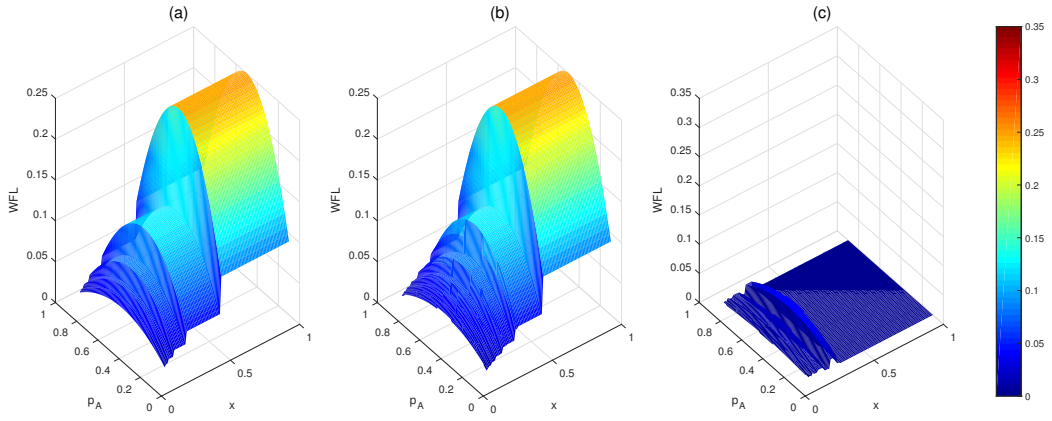


Figure 13: For different arrival probabilities and waiting costs, we show the (a) WFL of FCFS, (b) WFL of SIRO and (c) difference in WFL between both mechanisms. Mismatch values is $x = 0.5$.

D.2 Extensions

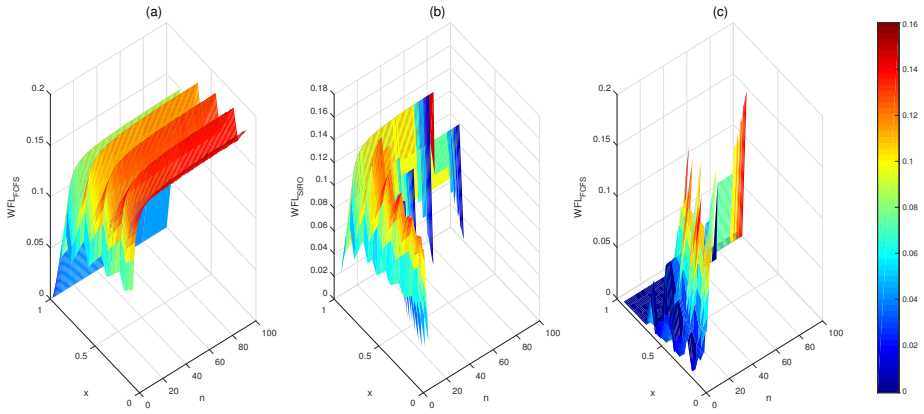


Figure 14: (a) WFL of FCFS, (b) WFL of SIRO and (c) difference in WFL between both mechanisms for different amounts of item types and mismatch values. Waiting costs are $c = 0.1 \cdot \frac{2}{n}$.

The missing values in Figure 14(b) and (c) are due to the maximum IC buffer-queue lengths becoming too high in the SIRO mechanism as that it could be computed within a reasonable amount of time.

	$n=2$		$n=3$		$n=5$		$n=10$	
$\hat{c}$	FCFS	SIRO	FCFS	SIRO	FCFS	SIRO	FCFS	SIRO
0.05	2.49	2.21	3.21	2.77	3.72	3.1	3.93	3.08
0.10	5.0	4.35	6.4	5.55	7.26	6.17	7.47	6.18
0.15	7.44	6.54	9.5	8.31	10.68	9.36	10.9	9.44
0.20	9.59	8.63	12.45	11.2	14.04	12.63	14.2	12.85
0.25	12.44	10.88	15.83	14.01	17.49	16.02	17.22	16.42
0.30	13.54	13.04	17.44	16.93	19.92	19.55	20.55	20.39
0.35	15.09	15.25	19.87	19.94	23.17	23.25	24.79	25.05
0.40	17.47	17.51	22.95	23.36	27.47	27.69	30.14	30.58

Table 7: Resulting welfare loss in percent for the heterogeneous mismatch values and different amounts of item types n. Waiting costs are $c = \hat{c} \cdot \frac{2}{n}$.

	Agent arrival probability (p_α, p_β)									
	(0.5, 0.5)		(0.4, 0.6)		(0.3, 0.7)		(0.2, 0.8)		(0.1, 0.9)	
$\hat{c}$	FCFS	SIRO	FCFS	SIRO	FCFS	SIRO	FCFS	SIRO	FCFS	SIRO
0.05	2.49	2.21	2.88	2.54	3.65	3.26	4.74	4.28	6.06	5.53
0.10	5.0	4.35	5.31	4.65	5.98	5.28	6.98	6.20	8.22	7.37
0.15	7.44	6.54	7.71	6.78	8.26	7.32	9.12	8.14	10.25	9.21
0.20	9.59	8.63	9.85	8.89	10.32	9.36	11.07	10.07	12.04	11.03
0.25	12.44	10.88	12.59	11.01	12.96	11.39	13.59	12.02	14.44	12.86
0.30	13.54	13.04	13.70	13.14	14.03	13.49	14.55	14.03	15.28	14.75
0.35	15.09	15.25	15.22	15.36	15.52	15.63	15.95	16.05	16.54	16.63
0.40	17.47	17.51	17.52	17.62	17.71	17.81	18.05	18.12	18.46	18.57

Table 8: Resulting welfare loss in percent in the 2-item economy for different specifications of imbalance.

E Matlab code for simulation

In the following we report the Matlab code for our simulation of the n-item economy for different waiting costs (cf. Table 7). As both mechanisms only differ in their buffer-queue policy, we first report the code for the FCFS mechanism and then, secondly, how SIRO differs.

```
%% Preamble
clear;
clc;

%% Parameters
T = 1000000;
iterations = 2;
c_hat_input = 0.05:0.05:0.4;
```

```
n_input = [2, 3, 5, 10];
WFL_FCFS = zeros(numel(c_hat_input), numel(n_input));
WFL_SIRO = WFL_FCFS;
welfare_std_FCFS = WFL_FCFS;
welfare_std_SIRO = WFL_FCFS;

%% Loop FCFS
for n = n_input
    p = 1/n;
for c_hat = c_hat_input
    c = c_hat * 2 / n;
    vec_welfare_iter = zeros(iterations, 1);
    record_convergence_FCFS = zeros(T, numel(n_input));
for i = 1:iterations
    % set up queues
    Q = zeros(10000, n);
    queues_values = Q;
    count_favorite = 0;
    count_mismatch = 0;
    count_queue = 0;
    vec_welfare = zeros(T,1);
    hist_wait = [1:10000]';
    hist_entries = zeros(10000,1);

for t = 1:T

    % item arrives
    item = randi(n);
    assigned = 0;

    % check if there are values in the queue
    q_n = Q(:, item);
    if q_n(1) ~= 0
        % record historical waiting time (-1 because they start with 1)
        waiting_time = floor(q_n(1))-1;
        % floor because it needs to be an integer value for hist_wait
        position_joined = round((q_n(1) - (waiting_time+1)) * 10000);
        weight = hist_entries(position_joined);
        hist_entries(position_joined) = weight +1;
        hist_wait(position_joined) = ...
            (weight * hist_wait(position_joined) ...
            + 1 * waiting_time) ...
            / (weight+1);
        if position_joined == 1
            record_convergence_FCFS(t) = hist_wait(1);
        end

        % change queue accordingly
        Q(1, item) = 0;      % take out first agent
        Q(:, item) = [Q(2:end, item); 0];   % correct positions of others
        count_favorite = count_favorite + 1;

        % record value
        vec_welfare(t) = queues_values(1, item);
        queues_values(1, item) = 0;
        queues_values(:, item) = [queues_values(2:end, item); 0];
    else
        while assigned == 0
```

```
67          % draw agent (and have the possibility of always queuers)
68          always_queuer = rand;
69          if always_queuer > 0.01
70              x = 0;
71          else
72              x = 1;
73          end
74          agent = randi(n) + 0.1 * x;
75          agent_type = round(agent);
76          % check if item is of favorite kind
77          if agent_type == item
78              count_favorite = count_favorite + 1;
79              assigned = 1;
80              vec_welfare(t) = 1;
81
82                 else % otherwise check value from all item types
83                 mismatch_values = rand(1, n);
84                 %correct mismatch value for favorite item
85                 mismatch_values(agent_type) = 1;
86
87                 % check which items the agent would consider waiting for
88                 w_bar = (mismatch_values - mismatch_values(item))/c;
89
90                 k_offered = zeros(n,1);
91                 for j = 1:n
92                     k_offered(j) = nnz(Q(:,j)) + 1; % offered spot in each
93                                                    % queue
94                 end
95
96                 Expected_wait = zeros(n, 1);
97                 for j = 1:n
98                 Expected_wait(j) = hist_wait(k_offered(j)); % E(t_x) per
99                                                            % item type
100                end
101                %Expected_wait = 1/n *k_offered; % True expectation
102
103                % check for all items, for which it would be rational to
104                % join the buffer-queue
105               accept = zeros(n,1);
106                for j = 1:n
107                    if w_bar(j) ≥ 0 % if positive value
108                        if Expected_wait(j) ≤ w_bar(j) %
109                            accept(j) = 1;
110                        else
111                            accept(j) = 0;
112                        end
113                    end
114                end
115
116                % check which one is preferred in case of multiple ...
                      possible
117                % ones
118                if sum(accept) > 1
119                    % select the one, where  the difference is higher
120                    difference = w_bar - Expected_wait';
121                    highest_diff = max(difference);
122                    join_q = find(difference == highest_diff);
123                else
```

```
124                     join_q = find(accept);
125                 end
126
127                 % always queuers queue
128             if agent - agent_type > 0
129                     count_queue = count_queue + 1;
130                     Q(k_offered(agent_type), agent_type) = ...
131                         1 + k_offered(agent_type) * 0.0001;
132                     queues_values(k_offered(agent_type), agent_type) ...
                            = ...
133                         mismatch_values(agent_type);
134
135             % accept mismatch or join best buffer-queue
136             elseif isempty(join_q)
137                 count_mismatch = count_mismatch + 1;
138                 vec_welfare(t) = mismatch_values(item);
139                 assigned = 1;
140             % otherwise join best queue
141             elseif isempty(find(Q(:,join_q), 1, 'last')) == 1
142             % add the additional value to capture the spot at which
143             % the agent joined the queue
144                 Q(1, join_q) = 1 + k_offered(join_q) * 0.0001;
145                 queues_values(1, join_q) = mismatch_values(join_q);
146                 count_queue = count_queue +1;
147             else
148                 Q(k_offered(join_q), join_q) = ...
149                     1+ k_offered(join_q) * 0.0001;
150                 queues_values(k_offered(join_q), join_q) = ...
151                     mismatch_values(join_q);
152                 count_queue = count_queue +1;
153             end % ends comparison with historical waiting times
154         end % ends if statement that checks if item is of favorite type
155         end % ends assignemnt of item
156     end % ends if statement that checks queue
157     Q(Q≠0) = Q(Q≠0) + 1; % add plus one to queuing agents to capture
158                          % waiting time
159 end % ends t
160     disp(['FCFS     Items:' num2str(n) '; c_hat:' num2str(c_hat)...
161     '; Iteration: ' num2str(i)])
162 vec_welfare_iter(i) = mean(vec_welfare);
163 end % ends iterations
164 welfare_std_FCFS(c_hat_input == c_hat, n_input == n) = ...
165     std(vec_welfare_iter);
166 WFL_FCFS(c_hat_input == c_hat, n_input == n) = 1-mean(vec_welfare_iter);
167 end % ends c_hat values
168 end % ends items
```

For SIRO, the code is similar, but instead of the reported lines 42–64, the following code is implemented.

```
1         % select random agent from buffer-queue
2           selected_agent = randi(nnz(q_n));
3
4           % record historical waiting time (-1 because they start with 1)
5           waiting_time = floor(q_n(selected_agent))-1;
6           % floor because it needs to be an integer value for hist_wait
7           position_joined = round((q_n(selected_agent) - ...
```

```
            (waiting_time+1)) * 10000);
        weight = hist_entries(position_joined);
        hist_entries(position_joined) = weight +1;
        hist_wait(position_joined) = ...
            (weight * hist_wait(position_joined) ...
            + 1 * waiting_time) ...
            / (weight+1);

        if position_joined == 1
            record_convergence_SIRO(t) = hist_wait(1);
        end

        % change queue accordingly
        Q(selected_agent, item) = 0;     % take out selected agent
        Q(:, item) = [Q((1:selected_agent-1), item); ...
            Q((selected_agent+1:end), item); 0];   % correct ...
            positions of others
        count_favorite = count_favorite + 1;

        % record value
        vec_welfare(t) = queues_values(selected_agent, item);
        queues_values(selected_agent, item) = 0;
        queues_values(:, item) = [queues_values(1:selected_agent-1, ...
            item); queues_values(selected_agent+1:end, item); 0];
```