Modelling and Improving Social Housing Markets

by

O. Maas

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday July 15th, 2020 at 15:30.
This research was undertaken in co-operation with Ymere.

Student number:	4606159	
Project duration:	October, 2019 – July, 2020	
Thesis committee:	Dr. M. M. de Weerdt,	TU Delft, supervisor
	Dr. N. Yorke-Smith,	TU Delft
	Dr. W. Brinkman,	TU Delft

TU Delft

Abstract

Social housing corporations in the Netherlands have a limited ability to influence the social housing market through offering houses to households outside of the regular waiting-list system, but often do not use an algorithmic strategy for utilising this freedom well. There is thus an open question of whether there exist algorithmic approaches that offer useful strategies for improving social housing markets.

To investigate this question, we introduce the Social Housing Market Problem, which integrates indifference, existing tenants, numerical preferences instead of ordinal ones, and a dynamic market, into the existing Housing Allocation problem. Here we distinguish between the constrained problem variant, where property chains and cycles are not allowed, and the unconstrained variant, where they are. We utilise an objective function that maximises the average numerical fit between all families in the market and their (lack of) houses. A variety of algorithms are adapted and developed for both problem variants; some of these serve as baselines meant to resemble current strategies; some are experimental; and others are of primarily mathematical interest. In particular, we develop an original and optimal algorithm for this problem which runs in exponential time. We compare our algorithms' performances on multiple problem types. The practical applicability of our research is limited by a number of design choices meant to simplify this complex real-world problem, but our results do suggest that some of our algorithmic approaches may indeed form the basis of better strategies for improving social housing markets. Finally, we suggest a number of improvements that may be made in future research to further develop our model and algorithms.

Preface

I would like to express my gratitude to Mathijs de Weerdt for his help, guidance, and valuable insights throughout this project; to Erik Smit, André van Diest, and the Match&Markt team at Ymere for their experience and time; to my family and friends for their encouragement; and to Matt Mikuš in particular for being my accountability partner in the latter months of this project. I'm grateful for your support.

O. Maas
Delft, July 2020

Contents

1 Introduction

In the social housing market in Amsterdam, the majority of house rentals are mediated through the WoningNet system, in which people may apply for houses themselves. When they sign up for this system, they are put on a waiting list; the duration of time they have spent on this list is a primary factor in how likely they are to be given the opportunity to rent a house. However, in a city as crowded as Amsterdam, a considerable portion of the total house supply is in high demand; as a consequence of this, the expected waiting time for families soars. Social housing corporations such as Ymere have a limited opportunity to influence the market by offering a small amount of houses to families themselves. Often, however, these corporations do not yet make use of this opportunity in a strategic manner, and when they do, they mostly do so on an informed but ad-hoc basis. There is thus an open question: **how might housing corporations better leverage their limited control over the market so as to improve a housing market, and to what extent might a housing market be improved to begin with?**

The mathematical literature around the topic of house allocation mostly takes the point of view of a university allocating dormrooms to students or faculty members [1, 17, 19] or treats the problem as a purely mathematical one [3, 10, 11]; the few works that actually view the house allocation problem as pertaining directly to social housing, are often not academic, and as such propose only basic algorithms [16]. As such, the majority of the house allocation strategies that the literature recommends, misses some key factor of the situation that housing corporations find themselves in, and though many of them have valuable features, they often offer few guarantees regarding to what extent, exactly, they help the housing market achieve its full potential. For example, many existing papers rely on households having a full comparative ranking of all houses on the market; in practice, it is not achievable to obtain such detailed information about households' preferences.

It would be valuable to answer the question at hand from a perspective nearer to that of a social housing corporation, so that they may be empowered to make better decisions in allocating houses to households, and thereby aid the social housing market. The purpose of this work is thus to come to an increased understanding of, firstly, what it might mean for a social housing market to be improved; secondly, through what strategies a social housing market might be improved along these dimensions; and thirdly, how effective these strategies are. To this end, we model the social housing market and compare the performance of various strategies—some of which are adapted from existing literature, and some of which are original—on both mathematical and semi-realistic problem instances.

The organization of this work is as follows. In Chapter 2, we first give a basic, non-mathematical overview of the problem that social housing corporations face. Next, we distil vital problem constraints from this general description and, through a perusal of related literature, arrive at a model choice as well as two relevant existing algorithms, namely, the Worker-Optimal Stable Matching Algorithm [6] and the Minimum-Cost Perfect Matching Algorithm [14]. In Chapter 3 we introduce and describe an original class of mechanisms. We moreover introduce, in Section 3.4, the **IR-Cycles** algorithm, which, though slow, we prove achieves a solution which scores at least as well as on our main scoring metric as any other mechanism that is subject to the same problem constraints—a guarantee we did not find replicated in the existing literature. Finally, we prove that the class of algorithms we defined, as well as IR-Cycles, are not included in the general Generalized Absorbing Top Trading Cycles family of algorithms for the Housing Allocation with Indifference problem, which, of the extant problems researched, most resembles ours. In Chapter 4 we describe our experimental set-up, including various methods used to generate data and measure scores, and describe our questions and expectations for the experiments' results; we next supply these results in Section 4.6 and discuss them in Section 4.7. In Chapter 5, we reflect on the limitations of our model choices and, with these in mind, offer practical insights based on our results. Finally, in Chapter 6 we summarise our theoretical and practical contributions, and suggest steps for future research on this problem.

The main contributions of this work are as follows. We introduce, define, examine, and contextualise the new Social Housing Market problem; we adapt several existing algorithms and create original strategies for this problem; and we compare their performances on a large variety of matching types. We find that: (i) the Minimum-Cost Perfect Matching Algorithm provides a quite reliable upper bound on algorithms' performances; (ii) taking future information into account may not make much difference in the quality of our decision-making; (iii) there may not be a large benefit, in a realistic setting, to pursuing 'cycles' and 'chains', two types of actions that may be taken by a social housing corporation; and (iv) our results indicate that making use of algorithms that are more sophisticated than a standard local strategy, may allow housing corporations to gain a sizeable increase in the matching's overall quality.

2 Problem Model & Related Literature

In this chapter we first introduce, in Section 2.1, the central problem of this work: the Social Housing Market Problem, whose key features are formalised in Section 2.2. Following this, in Section 2.3 we review some well-known related mathematical problems, and go over the reasons why these fall short of the complexity of the social housing market problem; thus we come up with our own formal problem description in Section 2.4. Nevertheless, insights for these problems might prove helpful; thus in Section 2.5 we go through related literature to find existing algorithms. Most of the algorithms that we find are a bad fit in some way, but two stand out: the Worker-Optimal Stable Matching Algorithm (WOSMA), and the Minimum-Cost Perfect Matching Algorithm (MCPMA).

2.1. The Social Housing Market Problem: An Informal Description

Let us start with a basic, informal description of this work's main topic: the **Social Housing Market Problem**.

We have a market with houses and families. For any house and any family, we may describe how well this family fits this house. We do this by looking at factors such as the family's income relative to the house's rent; the amount of rooms the house contains relative to the family's size; and whether the house is sufficiently accessible to families that should need it. This leads us naturally to our **primary goal**: namely, to find some allocation of houses amongst these families that leads to the highest possible average fit between the average family and their house.

Our problem is made more complex by the fact that it has a dynamic component. The market is ever-changing, with new houses and families appearing over time. A strategy for allocating houses that does well in the short-term, isn't guaranteed to perform satisfactorily in the long run. We want to be able to account for changes in the market.

Finally, unlike a university that allocates dormrooms to its students, a social housing corporation only has the power to recommend houses to families; it is up to the families themselves to decide whether they want to accept the corporation's offer, or try their luck on the market as a whole. Thus whichever house they are offered, must at the very least be better than the house they owned in the first place. Any strategy that can guarantee that this is always the case, is said to be *Individually Rational*.

The challenge we face lies in first defining some mathematical way to describe the fit between each house and each family; next, in clarifying exactly how well any given social housing market could possibly score on our chosen goal; and finally, in either achieving or approximating this best-possible outcome as closely as possible using some allocation strategy that is individually rational.

A final important consideration for housing corporations is the question of when to pursue either of two types of actions: chains and cycles. **Chains** are actions in which some family moves to some empty house; some other family moves to the house that the first is leaving uninhabited; and so on, until finally some family leaves a house behind that is not immediately filled up by any family. **Cycles**, meanwhile, represent an action in which a sequence of families all move to the house currently owned by the next family in said sequence.

With this in mind, we distinguish between two different problem variants: the unconstrained problem variant, in which chains and cycles may be performed within a single timestep; and the constrained problem variant, in which a house that is left empty by some family may not immediately be filled by another family. In other words: in this latter problem variant, chains and cycles may not take place.

Having established the basics of the social housing market problem and its key factors in this chapter, we may now define a corresponding mathematical model for the problem. After this, we provide an overview of related literature, and highlight two algorithms that we include in our overall comparison.

2.2. Key Problem Features

When we gave our informal problem description in Section 2.1, we highlighted several key defining features of our problem. In order to search the literature for a problem that fits our problem, we briefly formalise these key features here so that we may better understand the requirements of the social housing market problem:

F.1 We have two classes of objects—houses and families—across which matches may occur.

F.2 Each (house, family)-pair is associated with some fit-score.

F.3 Our primary goal is to find a matching of houses and families such that average fit is maximized.

F.4 Any viable algorithm must provide an individually rational solution.

F.5 Our market is dynamic; the set of houses and the set of families both change over time.

Feature F.4 is a relatively relaxed constraint, compared to what some authors require from their algorithms in similar problems: they desire to have an algorithm that is not merely individually rational, but which moreover incentivizes all partipants not to lie about their desires, a quality which is known as strategy-proofness. However, in the case of housing corporations, most information that is available to them comes from objective measurements that provide little opportunity for strategic behaviour; thus the trait of strategy-proofness is not very relevant to our research. Hence we leave it out of consideration.

2.3. Related Problems

There are several well-known mathematical problems that contain some of these five key features. The Stable Marriage problem (introduced in the seminal paper by Gale and Shapley [9]), in which the goal is to find a matching between two sides containing mutual ordinal preferences, such that the resulting matching contains no pair of nodes, each from different sides of the graph, that both prefer each other over their current matc, has features F.1 and F.4, but lacks the others. The Maximum Matching problem, where one tries to find a bipartite matching of maximum size, can be treated as a Flow Network problem (both discussed by Kleinberg and Tardos [14]), neatly fits features F.1 and F.3, but relevant algorithms often fail feature F.4 and often lack the dynamic angle present in feature F.5.

Meanwhile, the House Allocation problem [17], in which houses are distributed amongst agents that each have ordinal preferences over houses, and in which sometimes there already exist tenants that will only take a house they prefer to their current house, only has features F.1 and F.4. Finally, the Online Bipartite Matching Problem looks a lot like the social housing market problem, but fulfills feature F.5 only in part: In most definitions of this problem that we could come across (e.g. Kesselheim et al. [12], Khuller et al. [13]), only one side of the bipartite graph is dynamic; never both[1]. Moreover, feature F.2 (and thus feature F.3) often is not present, and whenever a new node enters the graph, it must either be matched immediately and irrevocably, or forever be left unmatched; this is in contrast to the social housing market problem, which does not contain such a constraint.

The problem that comes closest to the social housing market problem is the House Allocation with Indifference problem; it lacks the dynamic angle and thus does not natively incorporate feature F.5, but provides an otherwise useful model. Jaramillo and Manjunath [11] describe the difference that the addition of indifference makes: the well-known Top Trading Cycles mechanism (which for the traditional House Allocation problem identifies a unique core assignment and which is the only rule which

[1] This is likely due to the fact that, as Kesselheim et al. [12] note, “for general weights and when [all] vertices arrive in adversarial order, every algorithm can perform arbitrarily bad.”

Family size	Yearly income (euros) between...	Maximum allowed monthly house rent costs (euros)
1 person	0-22,700	607,46
1 person	22,701-42,436	720,42
2 people	0-30,825	607,46
2 people	30,826-42,436	720,42
3 or more people	0-30,825	651,03
3 or more people	30,826-42,436	720,42

Table 2.1: Table showing the 2019 Dutch rules around income and maximum rent costs.

is individually rational, strategy-proof, and Pareto-efficient—the latter meaning that no changes can be made to the outcome to improve an agent's match without worsening another's), no longer achieves Pareto efficiency when indifference is present, and a core allocation is no longer guaranteed to exist. We refer the reader to their Example 1, a simple example case for which the possible assignments found by TTC are not Pareto-efficient; we have reproduced it in Chapter A for the reader's benefit. Crucially, indifference in preferences allows us to adapt House Allocation with Indifference algorithms that nominally make use of ordinal preferences, to fit the situation where instead we have a numerical fit-score for each possible match (feature F.2): households strictly prefer houses that give them a higher fit-score, and are indifferent across all houses that give them the same fit-score. In this manner, though feature F.5 is missing from this problem description, House Allocation with Indifference uniquely satisfies feature F.2, and thus serves as a solid basis for our exploration of the social housing market problem.

2.4. Mathematical Model

Though none of the problems we saw in Section 2.3 precisely match ours, some are quite similar; we use these as inspiration in creating our own formal problem description of the social housing market problem, which we base primarily on the key features we found in Section 2.2. Features F.1 to F.3 suggest the model choice of matching across a bipartite graph of houses and households, where edges between houses and households represent included matches (feature F.1) and are associated with a score representing the fit between said house and household (feature F.2). Then our goal is to find some matching on this bipartite graph that maximizes the average fit score (feature F.3).

In formalizing this model, let us distinguish between two versions of the social housing market problem, contingent wholly on whether or not we take feature F.5 into account: a **static** version, in which we are given a single market and asked to improve on it; and a **dynamic** version, in which our market changes over time and where the challenge is to find an approach that can handle these changes well.

Let us start by examining features F.2 and F.3. The basic units of our model are houses H and families F. Then a *fit evaluator* is a function that takes a house h and a family f, and returns some score $s \in [0, 1]$ denoting the quality of the *fit* between h and f. In other words, this is a measure of how well this house and this family fit together.

We currently distinguish between three binary types of fit evaluators:

1. $fit_{financial}(h, f)$, which measures the financial fit of a house and family according to whether they satisfy the conditions in Table 2.1.

2. $fit_{room}(h, f)$, which measures whether the house contains the right amount of rooms for the family.
   The house must have at least one room per family member, but no more than one extra. If either of these conditions fails, a zero score is returned.

3. $fit_{access}(h, f)$, which measures whether the house is sufficiently accessible for the family. If the head of the family is below the age of 65, any house fulfills this condition. If they are above that age, however, then either the living space must be located on the ground floor, or the building must contain an elevator; if both of these conditions are false, a score of zero is returned.

In order to calculate using fit-scores more efficiently, we distinguish between two types of comprehensive fit evaluation functions:

1. $fit_{avg}(h,f) = \left(fit_{financial}(h,f) + fit_{room}(h,f) + fit_{access}(h,f)\right)/3.$
2. $fit_{min}(h,f) = \min\left(fit_{financial}(h,f), fit_{room}(h,f), fit_{access}(h,f)\right).$

We sometimes use the generic function $fit(h,f)$ to denote a call of either $fit_{avg}(h,f)$ or $fit_{min}(h,f)$ when we are operating on a more abstract level in which the specific fit-function has not yet been chosen or specified. Wherever it is important to note which of the two we are using, we explicitly specify this.

Finally, $fit(\emptyset, f) = 0$ for any f and any fit-function. Having no house merits a low score.

Now that feature F.2 has been explained, we can move onto describing how features F.1 and F.3 factor into our model.

An instance of the **static** Social Housing Market problem is a weighted bipartite graph $G = (H \cup F, G_E)$ of houses H and families F. We have that $\forall h \in H, \forall f \in F$, there exists an edge $e = (h,f) \in G_E$ with weight $fit(h,f) \in [0,1]$ for some chosen fit-function.

A *matching* M_G is, for a given instance of the static social housing market problem G, a set of edges $M_G \subset G_E$ such that each house in G is connected to at most one family, and each family is connected to no more than one house. An edge $e = (h,f) \in M$ denotes that in this matching M, the family f lives in house h. A matching moreover contains knowledge about which families have no (social) housing, and which houses have no family living in them, thanks to its underlying problem graph G. The *size* $|M_G|$ of a matching M_G equals the amount of edges in M_G. Finally, we briefly abuse notation and define the function $M_G(h)$ to return h's matched family in M_G, or to return $\emptyset$ if h is unmatched. Similarly, $M_G(f)$ returns f's matched house in M_G, or $\emptyset$ if f is unmatched.

Feature F.3 naturally follows in our objective function, which, in conformance with our goal stated in Section 2.1, simply calculates the average fit of all families in the matching and returns a corresponding score between 0 and 1:

$$eval_{avg}(M_G) = \sum_{f \in F} \frac{fit(M_G(h), f)}{|F|}.$$

Now that the basics of how features F.1 to F.3 inform our model, are well-understood, it is prudent to describe how we integrate feature F.5.

In the **dynamic** matching social housing market problem, we have some dynamic matching (DM) which defines an initial matching m and an amount of timesteps $tCount$. Simulating time takes the form of advancing time one step (i.e. reducing $tCount$ by one) and adding one house and one family to the matching.

There are then two approaches to ’running’ this DM. The first, **PerStep**, performs one simulation step followed by the execution of some improvement strategy on the matching, $tCount$ times. The second, **AfterSteps**, performs $tCount$ simulation steps before executing some improvement strategy on the resulting matching. We have chosen this dual approach to examine how well the different algorithms that we compare, take advantage of the dynamic nature of the problem. In other words, we are asking the question: if the algorithm has no knowledge of the future (PerStep), how much worse does it then perform, compared to if it had perfect knowledge (AfterSteps)? An algorithm that performs well in AfterSteps, but does very poorly in PerStep, is likely not well-suited to the dynamic problem variant; an algorithm for which the two outcomes are both good and very similar, meanwhile, is clearly better able to deal with the changing nature of the market.

Finally, we carry over the distinction between the constrained and the unconstrained problem variants, which we brought up in Section 2.1. In the **constrained** problem variant, cycles and chains may not take place within a single timestep; after all, in reality, it is often difficult to neatly coordinate such chains of movement within a short amount of time. In the **unconstrained** problem variant, however, we *do* allow the instant execution of chains and cycles.

Now that we have a full understanding of our mathematical problem, we may delve into the literature to see what extant algorithms might fit our problem.

2.5. Related Algorithms

Kleinberg and Tardos [14] describe the efficient Minimum-Cost Perfect Matching algorithm, which takes a bipartite graph whose edges have nonnegative weights referred to as ‘costs’, and returns a perfect matching—that is to say, a subset of edges such that every node on either side is linked to some node on the other side—in which the sum of costs of the included edges is minimized. One small downside is that this algorithm requires an empty initial matching; moreover, it only works on bipartite graphs whose two sides have equal sizes. This latter requirement, however, is easily evaded through the addition of maximally high-cost dummy nodes to whichever side is smaller.

Fan et al. [7] give a complex psychological satisfaction model in which they compare the expectations of agents in a two-sided matching market with the quality of their final match; in this manner they set up an optimization problem in which the sum of agents’ elation is maximized. This optimization problem is reduced to a classical assignment problem, for which there exist several polynomial-time ($\mathcal{O}(n^3)$) algorithms.

Gan et al. [10] propose an efficient envy-freeness algorithm for use when the number of houses in a house allocation problem instance exceeds the number of families. Here envy-freeness refers to the quality that every family likes their own house at least as much as any other house that was assigned. They furthermore show that if the number of houses exceeds the number of families by a logarithmic factor, there is a high chance that the problem instance admits an envy-free assignment.

Erdil and Ergin [6] suggest the efficient ($\mathcal{O}(|F|^3 \cdot |H|)$) Worker-Optimal Stable Matching Algorithm, which makes use of Pareto-improvement cycles to improve a given matching, namely by exchanging houses amongst the households that own them such that no one is worse off, but at least one family prefers their new house to their old one. Notably, their algorithm admits problem instances in which families may be indifferent between multiple houses.

Sönmez and Ünver [17] propose a ‘You Request My House, I Get Your Turn’-mechanism to redistribute houses amongst households in a housing market that includes existing household-house links, where families are assumed to have strict ordinal preferences over all houses. Their mechanism is Pareto-efficient, individually rational, and strategy-proof.

Plaxton [15] introduce an efficient ($\mathcal{O}(n^3)$) and strategy-proof family of mechanisms for the House Allocation with Indifference problem; their family is included in the Generalized Absorbing Top Trading Cycle (GATTC) family of algorithms described by Aziz and De Keijzer [4], which is a set of mechanisms that are core-selecting (i.e. there can be no blocking coalition of agents within the graph) and Pareto efficient, but not necessarily strategy-proof. They note that it is as of yet an open question whether all Pareto efficient mechanisms (including those that also guarantee (strict) core stability) are included in GATTC.

The above algorithms consider solely a static problem variant. Our search for dynamic algorithms, sadly, was less fruitful. This is because, as noted in Footnote 1 we could find little research that deals with dynamic markets in which both sides (houses and families) display dynamic behaviour. Khuller et al. [13] and Kesselheim et al. [12], for example, both give ‘online’ algorithms which solve the dynamic weighted bipartite matching problem where nodes on one side are all known and present in advance, and nodes on the other side enter the market one at a time.

Akbarpour et al. [2] give a more fluid dynamic model where all nodes enter and leave the market stochastically; however, it only distinguishes a single type of node (rather than the bipartite model that befits the social housing market problem). Furthermore, its model of unmatched agents becoming critical over time, until they reach a point where they leave the market, is ill-suited to our problem. Finally, it models acceptable matches as being randomly determined, whereas in the real world, the quality of a match between some house and some household is not random. For these reasons, we assumed their insights would not translate into the social housing market problem.

We provide a comprehensive overview of the algorithms noted here in Table 2.2. From this overview, we see that some algorithms which seem promising, are not a great fit after all. For instance, the model by Fan et al. [7] is designed for a two-sided matching market, and is thus an ill fit for our one-sided problem; furthermore, its focus on psychological satisfaction which takes into account agents’ expectations, is a bit too far removed from our problem. The algorithm suggested by Gan et al. [10] is ill-suited as well: firstly, it is rarely the case in the social housing market that there are more houses than families; secondly, envy-freeness is not a quality we necessarily want to strive for, since there will always

Name	**Authors**	**Advantages**	**Disadvantages**	**Dynamic?**
MCPMA	Kleinberg and Tardos [14]	Efficient, optimal, uses edge weights	Requires initially empty matching	No
Satisfied Matching	Fan et al. [7]	Efficient, can be modified for edge weights	Psychological satisfaction perhaps ill-fitting & very complex model	No
Envy-Free Assignment	Gan et al. [10]	Efficient, envy-free, can be modified for edge weights	In practice, houses rarely exceed families & envy-freeness is not a requirement for us	No
WOSMA	Erdil and Ergin [6]	Efficient, IR, can be modified for edge weights	N/A	No
YRMH-IGYT	Sönmez and Ünver [17]	Efficient, IR & strategy-proof	Requires strict ordinal preferences	No
Algorithm 1	Plaxton [15]	Efficient, IR, strategy-proof, can be modified for edge weights	N/A	No
Bipartite Online Matching & Online Perfect Matching	Kesselheim et al. [12], Khuller et al. [13]	Efficient, optimal, uses edge weights	Only one side is dynamic	Yes
Greedy & Patient	Akbarpour et al. [2]	Interesting dynamic properties	Not bipartite, entirely different dynamic model	Yes

Table 2.2: An overview of related algorithms from the literature. Wherever the authors did not name their algorithms, we supply a brief descriptive name.

be houses the average family in the social housing market would have preferred to get; and finally, envy-freeness only looks at houses that have been assigned, which might lead it to leave the best houses unassigned so as to avoid envy in other families. This is not something we want to happen in the real world, given that we're looking to optimize the average fit. The mechanism proposed by Sönmez and Ünver [17], requires that each family has strict preference ordinals over all houses, which is not precisely the case in the real world—and to the extent that it is, we do not have access to this data.

A few algorithms stand out, however: The Minimum-Cost Perfect Matching algorithm (MCPMA) [14] and the Worker-Optimal Stable Matching algorithm (WOSMA) [6], seem particularly viable for our problem, and thus merit greater attention. As noted before, one of MCPMA's failings—namely, that it only takes bipartite graphs where both sides have an equal number of nodes—may be circumvented easily. However, even then it is not usable in practice, since it requires a matching to be initially empty for its guarantees to hold; thus its solution is not individually rational, meaning MCPMA satisfies features F.1 to F.3 but fails features F.4 and F.5. However, since the algorithm provides a mathematically optimal solution (read: a weakly super-optimal one in the case of a non-empty input matching), it can serve as an upper-limit benchmark to compare other solution approaches with.

Meanwhile, WOSMA's Pareto-improvement step of carrying out vacancy chains and/or cycles is a natural fit for the housing market.

Finally, the algorithm put forth by Plaxton [15] seems promising, but in both Plaxton's algorithm and in GATTC mechanisms as a whole, it is assumed that $|H| == |F|$. Perhaps this requirement may be dealt with in as trivial a manner as we could do for MCPMA. At any rate, however, we sadly discovered Plaxton's algorithm too late into this project to incorporate it into our experiments.

2.5.1. The Minimum-Cost Perfect Matching Algorithm

The MCPMA is designed to choose a matching based on some bipartite graph, where firstly there exists as an equal amount of nodes on each side, and secondly, each edge $e = (h, f)$ is associated with a nonnegative *cost* $cost_{h,f}$. It generates specifically that matching which minimizes the total cost, summed over all edges included in the matching, while ensuring that every node is matched. As noted above, the outcome matching that it provides, is not guaranteed to be individually rational with respect to the mechanism's input matching; however, it may be used to very efficiently provide an upper-limit benchmark with which we may ground the performance of other algorithms.

There are a number of concepts used by MCPMA that are important to understand.

Firstly, it utilizes a concept known as *prices*, in which each node w in the input graph (i.e. each house and family) is associated with some price value $p(w)$. These price-values, which are updated at every iteration of the algorithm, help prove that the final matching has minimum cost.

Secondly, given some graph G, some matching M on G, plus appropriate prices p, we may construct G's *residual graph* R_G. R_G starts out as a graph that copies the house and family nodes from G, and adds a source node s and a sink node t. We add directed zero-weight edges from s to all unmatched houses across zero-weight edges, as well as from all unmatched families to t. Next, we include all (undirected) edges in G as directed edges: namely from h to f if $(h, f) \in M$, and from f to h otherwise. In the end, for each edge (h, f) (resp. (f, h) in R_G, we set its weight to equal $p(h) + cost_{h,f} - p(f)$ (resp. $p(f) + cost_{h,f} - p(h)$.

Thirdly, MCPMA makes use of *augmenting paths*: in this case, these refer to directed $s - t$-paths in R_G. When we find an augmenting path P in R_G, we may *augment* its matching M along this path P by matching all h, f for which $(h, f) \in P$, and removing all matches h, f for which $(f, h) \in P$.

Since all edges $(h, f) \in R_G$ were not present in its associated matching M, and all edges (f, h) were, this process increases the size of the matching M by 1. After all, the $s - t$-path necessarily starts with some edge (s, h) where h is unmatched, and ends in an edge (f, t) where f was unmatched. In-between h and f, the path will contain as many edges from H to F (which each add one match to M) as edges from F to H (which each remove one match from M). The net effect is that the resulting matching will have one more matched house and one more matched family, than the original matching M.

The step of carrying out an augmenting path in a given matching is highly unrealistic, since, when it matches n nodes on both sides, it unmatches $n - 1$ nodes on both sides. In real life, those $n - 1$ families would never simply give up their houses like this. So the augmenting path step is unsuited to a dynamic matching market; but it is furthermore unsuitable to a static matching market as well, even though there we would first run all of the algorithm's steps, and only then carry out its final solution.

MCPMA proceeds as described in Algorithm 1.

Algorithm 1: MCPMA.

```
MCPMA $(G)$
    Input  : $G$ : Bipartite graph (where $|H| = |F|$)
    Output: $bestMatching$ : Matching
    Initialize matching $bestMatching$ to be an empty matching on $G$.
    Define $p(h) = 0$ for all $h \in H$, and $p(f) = \min_{h \in H} cost_{h,f}$ for all $f \in F$.
    while $|bestMatching| != |H|$ do
        Construct $R_G$ based on $G$, $bestMatching$, and prices $p$.
        Use Dijkstra's algorithm to find a minimum-cost $s$–$t$-path $P$ in $R_G$.
        Augment $bestMatching$ along $P$ to obtain a new matching $bestMatching'$.
        Find a set of prices $newPrices$ compatible with $bestMatching'$ using
          $UpdatePrices(R_G)$.
        Set $bestMatching = bestMatching'$.
        Set $p = newPrices$.
    return bestMatching
```

Algorithm 2: UpdatePrices.

```
UpdatePrices $(R_G)$
    Input  : $R_G$ : Residual Graph
    Output: $newPrices$ : Prices
    Initialize empty $newPrices$ function.
    Run Dijkstra's algorithm on $R_G$ to get the shortest path $P(w)$ from $s$ to any other node $w$.
    For each house or family node $w \in R_G$, set $newPrices(w)$ to equal the weight along $P(w)$
      plus $p(w)$.
    return newPrices
```

There remains the final question of whether we might perhaps render MCPMA individually rational, by modifying it so that it only carries out augmenting paths that improve the situation of every agent, and letting it act on a matching that is not initially empty. However, if we did this, MCPMA would lose its (super-)optimality guarantee. Furthermore, we already have another algorithm that only carries out sequences of moves that are beneficial to all agents involved: namely, WOSMA.

2.5.2. The Worker-Optimal Stable Matching Algorithm

The basic mechanism step in WOSMA is to find and carry out *improvement cycles* amongst the households in a *two-sided* matching market. Improvement cycles can take one of two forms: either it is a cycle of households, each of which at least weakly prefers the house of the household next in the sequence to their own and one of which **strictly** prefers the next household's house—'strict' denoting here that for this house h and family f we have not merely $fit(f,h) \geq fit(f, M_G(f))$, but rather the stronger fact that $fit(f,h) > fit(f, M_G(f))$—or it is a chain in which one household prefers an empty house to their own, another household prefers the first household's house to their own, and so forth, until some household's house is left empty (and here again, at least one of these preference relation must be strict). Given this reliance on cycles and chains, WOSMA is only viable in the unconstrained problem variant.

The algorithm first runs the well-known Deferred Acceptance algorithm [9] to gain a stable matching —one in which there exists no pair of nodes on either side of which both prefer the other to their current match, and for which at least one of these two preferences is strict—and carrying out an improvement cycle preserves stability in the matching; thus the final matching is stable[2] as well. Moreover, the final

[2]The definition of stability that is used here, applies to two-sided markets, where both sides have different preferences over the

matching is guaranteed to be Pareto efficient.

WOSMA makes use of so-called two-labeled graphs (TLGs)—named as such for their edges' binary weights. To create such a two-labeled graph TLG_M: given an input matching M_G, we initialize a graph TLG_M that includes all of G's family nodes, as well as an empty $\emptyset$-element. We then include three different types of edges in this graph:

1. $w \to v$. These we include whenever some family w prefers some other family v's house over their own. If this preference is strict, we give this edge a weight of 1; if they are indifferent between these houses, however, the edge is given a weight of 0.
2. $w \to \emptyset$. These we include whenever some family w prefers some empty house over their own. Again, this edge has a weight of 1 if this preference is strict, and a weight of 0 if they are indifferent.
3. $\emptyset \to w$. These we include whenever w has no house, or when no other family strictly prefers w's house. The weight of this edge is always 0.

Once we have such a TLG for a given matching, we find a cycle as follows. First we run Tarjan's path-based algorithm for identifying strongly-connected components of TLG_M. Then we look for strict cycles in the resulting strongly-connected components using a depth-first-search approach.

A functional description of WOSMA may be found in Algorithm 3.

Algorithm 3: WOSMA. The Deferred Acceptance step is not used in our implementation of WOSMA; hence it is included here in brackets.

```
WOSMA ($M_G$)
    Input  : $M_G$ : Matching, $fit$ : Fit Evaluator
    Output: $M_G$ : Matching
    [Run the Deferred Acceptance algorithm on $M_G$ to get a stable matching.]
    Create a two-labeled graph $TLG_M$ based on $M_G$ and $fit$.
    Try to find an improvement cycle[3] $cycle$ in $M_G$. Set $cycle$ to $null$ if none can be found.
    while $cycle != null$ do
        Execute $cycle$ in $M_G$.
        Create a new two-labeled graph $TLG_M$ based on the updated $M_G$.
        Try to find an improvement cycle $cycle$ in $M_G$. Set $cycle$ to $null$ if none can be found.
    return $M_G$
```

In this chapter we have given a mathematical description of the social housing market problem. Regrettably, we have also found that few extant algorithms apply: MCPMA provides merely theoretical value, and though WOSMA provides a Pareto efficient outcome, we have no guarantees of its quality. Both of these algorithms were moreover designed for problems that are not precisely ours, suggesting that perhaps there exist better algorithms that are specifically designed for the Social Housing Market problem. In particular, our problem still lacks an optimal algorithm. We are thus motivated in Chapter 3 to analyze the problem in greater depth and introduce an original algorithm, **IR-Cycles**, which gives us an outcome with an average household fit that is at least as high as the average household fit of *any* individually rational outcome.

other side. It does not translate directly to the one-sided type of market that we have, where only households have preferences over houses.

3
Theoretical Results

In Chapter 2, we noted that there is no literature that explicitly deals with the Social Housing Market problem; the Housing Allocation with Indifference problem comes close, but differs in some important ways. Thus, though we adapted algorithms from other problems—WOSMA and MCPMA—to our own problem, these algorithms were not designed for the Social Housing Market problem, leaving open the question of whether there exist algorithms that better fit our problem. In particular, we currently lack any kind of optimal algorithm. In this chapter we aim to first provide the reader with a better understanding of the Social Housing Market problem, after which we use this understanding to develop new algorithms that are better-suited to our problem.

We first provide the reader with a number of theoretical concepts and results that help us better understand the landscape of the social housing market problem. Next we introduce a new class of mechanism, the family of WOSMA-like mechanisms, which proves to be a quite general class of mechanisms. Following this, we comment on how difficult it can be to find the right WOSMA-like mechanism by showing how several intuitive WOSMA-like mechanisms perform suboptimally on small instances of a highly constrained problem variant; afterwards, we show that a simple adaptation of MCPMA solves this problem optimally. Finally, we introduce our original IR-Cycles algorithm, which is unique, amongst the approaches named in this thesis, in outputting an outcome which is not merely Pareto efficient but which also, of all outcomes that are individually rational given our initial state, scores highest on the goal function we described in Section 2.1—namely to maximize the average fit across families in a matching. In other words, IR-Cycles is optimal, thus helping us answer the research question we raised in Chapter 1 by allowing us to quantify to exactly what extent a given matching can be improved.

We end on an analysis of the WOSMA-like family of mechanisms and the IR-Cycles algorithms, proving that they are not part of the large Generalized Absorbing Top Trading Cycles (GATTC) family of mechanisms defined by Aziz and De Keijzer [4] for the Housing Allocation with Indifference problem.

In this chapter we make us of a model similar to that of the Two-Labeled Graph we saw in Section 2.5.2. Since chains are represented in these graphs as cycles, we simply use the term 'cycles' to refer to either cycles or chains in these graphs, except where otherwise noted.

3.1. Constrained Pareto Efficiency

We first introduce two concepts that are helpful for understanding the problem space at hand: constrained Pareto improving actions, and constrained Pareto efficiency. They are analogous to the familiar concepts of 'Pareto improving' and 'Pareto efficiency,' respectively. We define them as follows.

Normally, we use 'Pareto improving' to refer to a step that weakly improves a market for all agents, and strongly improves it for at least one agent. We now introduce **constraints** on the Pareto improving steps that may be taken. A constraint is any 'rule' which reduces (by definition weakly, but in most realistic cases, strongly) the set of Pareto improving steps that may be taken. Such constraints might take the form of: 'Cycles carried out may not have length larger than 3,' 'Cycles carried out must improve all involved agents' fitness by at least 0.2,' and so forth. We use the term **constrained Pareto improving actions** to denote some subset of Pareto improving steps as constrained by some set of constraints, a set possibly containing only one constraint (or indeed zero, meaning that statements about constrained Pareto improving steps are generalizable to Pareto improving steps).

It is possible that at some point there remain Pareto improving steps to be taken, but that simultaneously no more constrained Pareto improving steps are possible, corresponding to some set of constraints. The concept of **constrained Pareto efficiency** follows naturally from this situation. A market is constrained Pareto efficient, relative to some set of constraints, when there remain no more constrained Pareto improving steps to be taken (relative to this same set of constraints), regardless of whether any Pareto improving actions remain available.

We use the term **constrained Pareto improving mechanisms** to refer to mechanisms which, given some constraint, repeat the following routine: 'Look for a constrained Pareto improving step that we can take. If any is found, perform it. If none are found, end.' If such a mechanism is capable of finding all constrained Pareto improving steps at any time—i.e. it will at no point miss any—then we say that it is constrained and **strongly** Pareto improving. If, instead, it will always find *some* constrained Pareto improving step so long as there are any, then we call it constrained and **weakly** Pareto improving.

We finally introduce the concept of a **best achievable constrained Pareto efficient state**. Given some problem instance (as well as some set of constraints), take the set of all problem states that we can achieve by taking only (and possibly zero) constrained Pareto improving steps. Next, filter this set to leave only those states which are constrained Pareto efficient. Then the best achievable constrained Pareto efficient state is some state in this problem instance that is amongst these states, and which is at least as good as any other state in this set, according to some goal function of our choosing. Notably, however, here we do not consider the set of *all* constrained Pareto efficient states that this problem instance knows; merely those that can be attained solely through performing constrained Pareto improving steps from the initial given state.

A number of basic results follow.

Lemma 3.1.1. *If we operate a constrained and strongly Pareto improving mechanism on some finite problem instance, it will eventually end in some constrained Pareto efficient state.*

Proof. Each step the mechanism takes improves the matching a little (because, being constrained Pareto improving, the step is at least Pareto improving), and the market, being finite, cannot be improved indefinitely; thus the mechanism eventually ends. (Here we make the important assumption that there are a finite number of values by which any improving step may improve the matching. For example, this is the case when we measure improvement to a precision of three digits after the decimal separator.) Furthermore, since at this point there are no more constrained Pareto improving steps to be found (and we know this since it would otherwise find any that are there), this final state is constrained Pareto efficient. □

Corollary 3.1.1.1. *Suppose we have a strongly Pareto improving mechanism that is, notably, not constrained. Then for any set of constraints, it follows that it is able to find a corresponding constrained Pareto improving step, provided that one exists.*

After all, at any point in the algorithm's execution, it is able to take any of all possible Pareto improving steps, and the set of these steps is a superset of the set of constrained Pareto improving steps; thus it can also find any step in the latter set.

As a consequence, any mechanism with this trait can easily be adapted to any constraint so as to produce a new mechanism which, relative to this constraint, performs constrained Pareto improving steps, and which is (by the above lemma) constrained Pareto improving.

Corollary 3.1.1.2. *A constrained and weakly Pareto improving mechanism is not necessarily constrained Pareto efficient.*

Corollary 3.1.1.2 follows from the fact that it is possible that a constrained and weakly Pareto improving mechanism would only ever find some Pareto improving step, but may at times miss some constrained Pareto improving step; that is to say, it may sometimes find only steps that are Pareto improving, but not constrained Pareto improving.

The original WOSMA is a constrained and weakly Pareto improving mechanism, since while looking for cycles, the first cycle that is found is also the one that is executed; in other words, it may find only a single Pareto improving step, which may not necessary be a constrained Pareto improving step. Given that it is weakly Pareto improving, it thus follows from Corollary 3.1.1.2 that it is not guaranteed to be constrained Pareto efficient given any set of constraints.

This begs the question: what would a constrained and strongly Pareto improving version of WOSMA look like?

3.2. Introducing WOSMA-like mechanisms

In this section we define a class of mechanisms that greatly resemble WOSMA, but which are constrained and strongly Pareto improving.

We first define here the notion of an 'improvement graph', which in general refers to a graph in which, given some problem instance G plus a corresponding matching M_G, all edge-wise improvements to M_G that adhere to some given set of constraints, are visible and may in cycles (or chains) be 'concatenated.' These cycles (or chains) may then be realized in the matching, provided they conform to our constraints. We may add constraints to this graph by simply removing those edges, or not considering those cycles, that do not satisfy them. In the case of the regular WOSMA without constraints (except that cycles be Pareto improving), the improvement graph is exactly the Two-Labeled Graph, as described in Section 2.5.2.

We may then introduce the family of 'WOSMA-like' algorithms. This family includes all algorithms of the following form: 'While there are cycles in the improvement-graph that conform to our given constraints, carry out any one of them.'

Theorem 3.2.1. *Given any set of constraints, there exists a WOSMA-like mechanism that finds a corresponding constrained Pareto efficient outcome. We may furthermore find such a constrained Pareto efficient WOSMA-like mechanism by simply using the default template of a WOSMA-like mechanism (which finds all Pareto improving steps) and modifying it so it only carries out constrained Pareto improving steps corresponding to these constraints.*

Proof. The original WOSMA is Pareto efficient [6], thus we know that if there is a Pareto improvement step, it can be found by looking for (and indeed is analogous to) some cycle in its improvement graph. It follows that if we find all possible cycles, we have found all possible Pareto improvement steps. Thus if we modify the original WOSMA to enumerate all cycles and to always select one that is, given some set of constraints, constrained Pareto improving (provided one such step is present) —then, by Corollary 3.1.1.1, we have a constrained Pareto efficient algorithm.

The class of WOSMA-like mechanisms is exactly the class of such modified versions of the original WOSMA, which we describe here. Thus there exists a WOSMA-like mechanism that finds a constrained Pareto efficient outcome given some set of constraints, and we have found its definition. □

Finally, for any set of constraints, given the corresponding WOSMA-like mechanism, there exists some action-selection strategy which gives us the best achievable constrained Pareto efficient outcome. After all, we know that the best achievable constrained Pareto efficient outcome is (by definition) achievable from the given initial state, and that it is achievable through some specific sequence of constrained Pareto improving steps. Then, given that the corresponding WOSMA-like mechanism (which we may find as described in Theorem 3.2.1) is capable of finding all possible constrained Pareto improving steps at any time, it follows that each of the steps from the desired sequence of steps may be found at the corresponding point in time. Thus for any set of constraints, the corresponding WOSMA-like mechanism can find the best achievable constrained Pareto efficient outcome; it is simply a matter of finding the right step-selection process strategy. Of course, this latter element is the hardest part; oftentimes strategies that one might expect would be optimal, turn out not to be. In our search for an algorithm that reaches the best achievable outcome for a highly constrained problem variant, we disqualified several seemingly valuable candidate algorithms. In the end, it was an adaptation of MCPMA which proved optimal.

3.3. Adapting MCPMA to the constrained problem variant

It can be hard to find a WOSMA-like mechanism that performs optimally, even when the set of possible actions is highly constrained. In this section we illustrate this idea by taking our 'constrained' problem variant[1] and showing that various intuitive WOSMA-like mechanisms perform suboptimally. Finally, we prove that the problem may be solved optimally through applying a modified version of MCPMA instead.

In the constrained problem variant our goal is to maximize the average fitness, with only chains of size 1 allowed; that is, any mechanism that acts on some input matching M^0 must give us an output matching M in which the only houses that have a different family (but which are not empty), are those that in M^0 were empty.

Our goal, then, is to find a mechanism which is individually rational and which furthermore returns the best possible individually rational outcome that conforms to the above constraint.

Let us choose some seemingly reasonable candidate heuristic, 'Greedy', which tries to make a seemingly optimal choice at every step while remaining more or less efficient. Firstly, it satisfies the problem variant's constraint, since it only moves families to houses that were initially empty. Secondly, in order to guarantee strong individual rationality, we define Greedy such that it at all times refuses to move families to houses they do not strictly prefer over their initial house.

In the discussion that follows, when we say that some family f_i who currently owns house h_i would 'benefit more' from some house h_x than another family f_j which currently inhabits house h_j, we do *not* mean to imply $fit(h_x, f_i) > fit(h_x, f_j)$. Rather, we mean that $fit(h_x, f_i) - fit(h_i, f_i) > fit(h_x, f_j) - fit(h_j, f_j)$. Similarly, when we say some house h_i benefits some family f_i more than some other house h_j benefits some other family f_j, we do not imply that $fit(h_i, f_i) > fit(h_j, f_j)$.)

Now let us go through several increasingly sophisticated versions of Greedy, each of which may be proven suboptimal by some counter-example.

3.3.1. Greedy-1

Greedy-1 proceeds as follows. We first identify the set of initially empty houses $H^\emptyset$ in the input matching M^0 with houses H and families F. (Here we have that $H^\emptyset \subseteq H$.) Now, so long as there exists some empty house in $H^\emptyset$ (i.e. not in $H \setminus H^\emptyset$) that any family $f \in F$ prefers to their current house, we pick some empty house $h_x \in H^\emptyset$ in the following way. First for each empty house $h_i \in H^\emptyset$ we define F_i' as the set of families for which h_i is a house they would benefit from at least as much as they would from any other empty house in $H^\emptyset$. Then pick some house h_x for which the set F_x'' is smallest (but non-empty), where $F_x'' \subseteq F_x'$ is defined as the subset of families in F_x' that would benefit from h_x at least as much as any other family in F_x' would. Finally, assign h_x to a random house in F_x''.

Lemma 3.3.1. *Greedy-1 is not guaranteed to return the best achievable solution.*

Proof. We give a proof by counter-example. Suppose that we have the following situation with two families (f_1, f_2) and two empty houses (h_1, h_2):

Neither family currently owns a house, and f_2 is indifferent between h_1 and h_2.

$$fit(f_1, h_1) = fit(f_2, h_1) + 0.1$$
$$fit(f_1, h_2) = fit(f_2, h_2) + 0.2$$

Then $F_1' = F_2' = F$, but $F_1'' = F_2'' = \{f_1\}$. Thus Greedy-1 may first choose either h_1 or h_2 at random. If it chooses to assign h_1 first, it would assign it to f_1, and next assign h_2 to f_2. However, the opposite allocation gives a higher average fitness. Thus Greedy-1 is not guaranteed to give an optimal outcome. □

3.3.2. Greedy-2

Greedy-2 proceeds similarly to Greedy-1. However, whenever there are multiple houses in $H_x = \{h_{x_1}, h_{x_2}, ...\}$, where H_x is defined as the set of houses h_{x_i} for which the set F_{x_i}'' is at least as small as any other F_{x_j}'' (for any $h_{x_j} \in H_x$), we prioritize a random house from that subset of H_x that contains all houses in H_x which would offer at least as high a benefit to any family as any other house in H_x. It is trivial to verify that Greedy-2 performs optimally in the counter-example to Greedy-1.

Lemma 3.3.2. *Greedy-2 is not guaranteed to return the best achievable solution.*

[1] In the constrained problem variant, cycles and chains may not take place within a single timestep; in other words, only chains of size 1 are allowed. See Section 2.4.

Proof. We give a proof by counter-example. Consider a situaton where we have two empty houses (h_1, h_2) and two families (f_1, f_2). The initial fit for each family with their own house is 0. Let us describe the fit between each house and each family as follows:

	h_1	h_2
f_1	0.6	0.5
f_2	0.6	0.1

Here $F_1' = F_1'' = \{f_1, f_2\}$ whereas $F_2' = F_2'' = \emptyset$. Then Greedy-2 would first pick h_1 to assign a family to, and would assign it randomly to either f_1 or f_2. The choice of assigning f_1 to h_1, however, is sub-optimal. Thus Greedy-2 is also not guaranteed to give an optimal outcome. □

3.3.3. Greedy-3

Greedy-3 proceeds similarly to Greedy-2. However, given some empty house h_x, and defining $\bar{H^\emptyset}$ to be the set of initially empty houses that are no longer empty, we now define F_x'' in the following way: given a house h_x, F_x'' is the set of families amongst those who would most benefit from h_x compared to their current house, whose marginal extra benefit from h_x compared to any other house h_y amongst the houses that were initially and are still empty, is at least as high as said marginal benefit is for any other family f_j. Intuitively, F_x'' is the set of families who would directly suffer the most if they were to lose the opportunity to move into h_x. Formally, we have:

$$\begin{aligned} F_x'' = \{ & f_i \in F_x' \\ & \mid \forall h_y \neq h_x \in H^\emptyset \setminus \bar{H^\emptyset} : \\ & (fit(h_x, f_i) - fit(h_y, f_i)) \geq 0 \\ & \wedge (fit(h_x, f_i) - fit(h_y, f_i)) \geq \max_{f_j \in F \setminus \{f_i\}} (fit(h_x, f_j) - fit(h_y, f_j))\} \end{aligned}$$

It may easily be verified that Greedy-3 performs optimally in the counter-example to Greedy-2.

Lemma 3.3.3. *Greedy-3 is not guaranteed to return the best achievable solution.*

Proof. We give a proof by counter-example. Let us have three empty houses (h_1, h_2, h_3) and three families (f_1, f_2, f_3). The initial fit for each family with their own house is 0. Let us describe the fit between each house and each family as follows:

	h_1	h_2	h_3
f_1	1.0	0.7	0.7
f_2	1.0	0.7	0.5
f_3	1.0	0.8	0.6

In the first step we would have that $F_1' = F, F_2' = F_3' = \emptyset$. Thus Greedy-3 first assigns h_1. We have that $F_1'' = \{f_1, f_2\}$. Greedy-3 thus randomly assigns h_1 to either family in F_1''. Suppose that it chooses to assign h_1 to f_1. Next under consideration is h_2 with $F_2' = F_2'' = \{f_3\}$; thus h_2 goes to f_3. Finally, h_3 goes to f_2. This gives us a final average fitness of $\frac{1.0+0.8+0.5}{3} = \frac{2.3}{3}$. However, the alternative assignment where f_1 gets h_3, f_2 gets h_1, and f_3 gets h_2, has an average fitness of $\frac{0.7+1.0+0.8}{3} = \frac{2.5}{3}$. Thus Greedy-3, too, is not guaranteed to give an optimal outcome. □

For a given house h_x, Greedy-3 first checks which families would most benefit from h_x (i.e. F_x'), and only then searches within this set for those families who would most suffer from not getting it (i.e. F_x''. One might wonder if doing the reverse might work better: namely to first check, given some house h_x, which families would most suffer from not getting it, and then amongst those choose some family that would most benefit from getting h_x. This slightly changed mechanism, however, would still fail on the counter-example to Greedy-3. We refer readers who wish to verify this to Chapter B.

Thus the WOSMA-like mechanisms that we came up with, were unable to perform optimally even on this very constrained problem variant. Fortunately, it is possible to modify MCPMA to perform optimally on the constrained problem variant.

3.3.4. Improvement-MCPMA as an optimal mechanism for the constrained problem variant

We may perform three alterations to MCPMA to create a new mechanism, **Improvement-MCPMA**, that efficiently solves the constrained problem variant to local optimality.

First, it is trivial to adapt MCPMA such that it chooses a matching in a bipartite graph where the sum of the weights along their edges is maximized, rather than minimized (so long as each edge's range of values is constrained). This we may achieve by adapting MCPMA to minimize not the sum of all included edges' weights, but rather to minimize $\sum_{(h,f)\in M}(1 - fit(h, f))$. This can be done simply by defining, for each edge $e = (h, f)$, a cost $cost_{h,f} = 1 - fit(h, f)$. Since 1 is the best weight an edge may have, minimizing the above sum now gives us a matching that maximizes the average fit across families.

The second modification is as follows. Given some problem instance of our problem variant, let us create a bipartite graph that contains nodes for all families on one side, and for all *empty* houses on the other side (rather than for all houses, as MCPMA would do), and which contains an edge between some empty house h and some family f if f would benefit from being allowed to move to h (compared to staying in their current house). Given the family's current house h_f, we assign each such edge a weight equal to $1-(fit(h, f)-fit(h_f, f))$, where 1 is the maximum value any fit can have. Thus each edge $e = (f, h)$ represents 1 minus the increase in fitness that the f would get from moving to h.

Then Improvement-MCPMA finds some allocation of these empty houses to families, such that the sum of all families' benefits is maximized. Since this maximizes the total increase in fitness, it also maximizes the average increase in fitness. Furthermore, since families would only be assigned some empty house if they strictly preferred said house over their current house, this outcome is moreover individually rational. Thus, given that our goal is to maximize the average increase in fitness, Improvement-MCPMA finds the best achievable individually rational outcome.

A minor limitation of MCPMA is that it only works in problem instances where $|H| = |F|$. To mitigate this, we may perform the third and final modification, which we hinted at in Section 2.5: we simply add dummy empty houses or dummy households to the above bipartite graph until we have a graph in which the amount of empty houses plus (possibly 0) dummy houses equals the amount of households plus (possibly 0) dummy households. Each dummy house (resp. household) is connected to all households (resp. houses), dummy or no, with an edge that has weight 1, representing the fact that moving a household to a dummy house (or, meaninglessly, moving a dummy household to a house) gives zero benefit. In the end, when we get Improvement-MCPMA's outcome matching, all houses owned by a dummy household remain empty, and all households that own a dummy house do not move at all.

Improvement-MCPMA performs optimally in static instances of the constrained problem variant. However, it does not translate intuitively to the natural dynamic version of this problem variant. This is because, if Improvement-MCPMA were applied only after all additions to the matching had been made (i.e. in the AfterSteps scenario noted in Section 2.4), then it would only be able to move households to houses that are empty at that moment, whereas a repeated application of Improvement-MCPMA, once at every timestep (i.e. in PerStep) would 'refresh' its list of free houses at every timestep, and thus be able to move households to different houses at different times, some of which are unavailable to the mechanism's AfterSteps variant. Meanwhile, it would also not do to naively execute the algorithm as many times as there are timesteps in the AfterSteps scenario. After all, suppose a house only arrives in the final timestep; then any optimal solution would only allow it to be part of a single move. In contrast, this naive plural execution of Improvement-MCPMA might let the house participate in as many moves as there are timesteps.

We have thus found an optimal PerStep algorithm for the static constrained problem variant, and the standard MCPMA constitutes a weakly super-optimal (though non-IR) algorithm for the static social housing market problem. Unfortunately, we are still missing a mechanism for the static social housing market problem that is precisely optimal as well as individually rational. Our attempts to find such an algorithm within the family of WOSMA-like mechanisms have failed; but if we move beyond this family, we may find exactly the algorithm we are looking for.

3.4. Introducing the IR-Cycles Algorithm

IR-Cycles is an individually rational mechanism for the social housing problem that resembles TTC and WOSMA(-like algorithms), but which, uniquely amongst them, returns, for the static problem variant, the best achievable individually rational outcome (as measured by our primary goal function which we described in Section 2.1, namely, to maximize the average fitness in the matching). It is capable of this by virtue of its ability to take actions which neither TTC nor WOSMA(-like algorithms) may take.

IR-Cycles proceeds in the following fashion: Until there are none left, execute any cycle (or chain) which gives an increase in the matching's average fitness, and which leaves each family either strictly more well-off than their initial state, or assigns it its initial house (if the family had one). Individual rationality is trivially guaranteed, and so long as the problem instance is finite (where we we again assume that there exist a finite number of values by which any improving step may improve the matching), it is trivial to show that the algorithm finishes in all cases.

It thus differs from existing algorithms: TTC only matches families with one of their favourite available houses, and WOSMA(-like algorithms), at best, only match families with a house they prefer to their current house. This difference lies in the fact that IR-Cycles may also take actions which (conform with the requirements noted above) decrease a family's fitness relative to their current state.

Theorem 3.4.1. *IR-cycles returns the best achievable individually rational outcome.*

Proof. We perform a proof by contradiction by showing that any alternative individually rational and achievable outcome which is better than that which IR-Cycles found, may be reached from that latter state solely through carrying out cycles of the type which IR-Cycles is allowed to carry out. We thus get a contradiction: It must be the case that such cycles are still present in the outcome returned by IR-Cycles, but by definition, IR-Cycles would only have returned some outcome in which such cycles are no longer present.

To perform our proof by contradiction, let us assume that an execution of IR-cycles on some given matching M^0 gave us some matching M, but that there exists some other matching M^* which is also individually rational with regard to M^0, but which has a higher average fitness than M does.

Let us remove, without loss of generality, all the families and corresponding houses that have the same match in both M and M^*, from our consideration. (That is to say, we redefine M and M^* to exclude these houses and families.) Now we have the following situation:

$$\begin{aligned}
&\forall i, M(f_i) \neq M^*(f_i)\\
&\forall i, fit(f_i, M(f_i)) > fit(f_i, M^0(f_i))\\
&\forall i, fit(f_i, M^*(f_i)) > fit(f_i, M^0(f_i))\\
&\exists j, fit(f_j, M^*(f_j)) > fit(f_j, M(f_j))
\end{aligned}$$

This fourth condition is guaranteed by the fact that M^* has a higher average fitness than M does.

Now, in M, consider the following cycle or chain: We move f_j to $M^*(f_j)$; we move the family currently inhabiting this house, $M(M^*(f_j))$, to the house which said family is inhabiting in M^*, namely $M^*(M(M^*(f_j)))$, and so forth, until we move some family to either $M(f_j)$ (making this a cycle) or to an empty house in M (making this a chain). (Of course, if $M^*(f_j)$ is empty in M to begin with, then we simply end there, which gives us a chain of length 1.) Since each family still in M and M^* is assigned a different house in M than they are in M^*, each family that is involved in the above cycles or chain would indeed get a different house in M through the above process; hence we have a valid cycle or chain in M.

Since each family in this cycle or chain would, if we were to carry it out from M, be moved from their position in M to that house which they own in M^*, and since M^* is individually rational with respect to M^0, it follows that this cycle maintains M's individual rationality with respect to M^0. After all, the third condition of the four noted above, is valid for all families, and therefore it is valid for those families that are part of this cycle.

There are now two distinct cases: Either carrying out this cycle or chain in M would improve the average fitness, or it would not. In the second case, let us not carry it out. We may then furthermore remove these families and their houses in M^* from our consideration in both M and M^*; after all, even after their removal, it must still be the case that this new M^* has a higher average fitness than this new M, since even if we gave these families in M the houses they have in M^*, the average fitness of M has not increased and thus must necessarily still be lower than the average fitness of M^*. Then we have the same situation we had before, except with a smaller pool of houses and families. We can then again find a similar cycle or chain and choose (based on whether it improves the matching's average fitness or not) if we should carry it out or not. As this process continues, one of two things will happen: either we eventually find some cycle or chain that, if carried out in M, would increase the average fitness, or we keep removing houses and families from our consideration as described until there are no more houses and families to be removed. In this latter case, however, our initial assumption fails: There is then clearly no family f_j in M^* anymore for which $fit(f_j, M^*(f_j)) > fit(f_j, M(f_j))$. This violates our assumption that M^* has a higher average fitness than M.

Thus the former case is true: at some point, the cycle or chain that we find in the manner described above (starting with a new f_j), must improve our matching's average fitness, and is furthermore individually rational. This violates our inital assumption that M was achieved by IR-cycles, which already has performed all such cycles or chains that may be found. Thus we have a contradiction.

Hence, given some matching M found by IR-cycles which is individually rational with regard to M^0, there exists no other matching M^* which is both individually rational with regard to M^0 and which has a higher average fitness than M does. □

To summarize: we have the surprising result that the best achievable individually rational outcome is *always* achievable solely by carrying out cycles which increase the matching's average fitness. IR-Cycles merely exploits this fact.

It is important to qualify IR-Cycles' optimality with the following observation: it indeed finds the best achievable individually rational outcome, but it only does so on a local basis, that is to say, in the static problem variant. In a dynamic matching it will at every timestep find the best possible matching that may be reached from the preceding outcome; but this does *not* guarantee that, given some dynamic matching at timestep 0, it finds at timestep t the best outcome it was possible to get at timestep t.

Given that IR-Cycles gives us a locally optimal outcome, it is the case that, if we run it in the After-Steps setting, we may gain an exact understanding of to what extent a given dynamic matching may be improved in an individually rational manner. After all, its AfterSteps outcome is trivially reachable from a PerStep perspective: in the worst case, a PerStep strategy which at any non-final timestep does nothing, and which at the final timestep takes those actions that the AfterSteps strategy would have taken, achieves exactly the same outcome.

Finally, it remains to analyze IR-Cycles' running time complexity. The dominant subroutine in a single IR-Cycles call is its cycle-finding subroutine. In our case, we rely on the Szwarcfiter-Lauer algorithm by Szwarcfiter and Lauer [18] to find cycles in our improvement graph. This algorithm has time complexity $\mathcal{O}(V + E \cdot C)$, where V is the amount of vertices in the improvement graph, E is the amount of edges it contains, and C is the amount of cycles. However, since we only ever require a single cycle to be found at any time, it is possible to modify the Szwarcfiter-Lauer algorithm to return the first positive-sum cycle it finds, rather than all cycles that are present in the graph; this improves the algorithm's running time and reduces its time complexity to *approximately* that of finding whether the graph has cycles, which is $\mathcal{O}(V + E)$. However, in the worst case, all but the last cycle that is found will be negative-sum, in which case we would have to go through all cycles in the improvement graph after all. Thus the final time complexity of this step remains $\mathcal{O}(V + E \cdot C)$, although the average complexity is much less.

Sadly, the improvement graph in IR-Cycles' case can contain prohibitively many edges, due to the large amount of choices that are available to the algorithm at any point. Furthermore, the Szwarcfiter-Lauer is called after every cycle execution, meaning it may be called approximately $\bar{C}$ times; here $\bar{C}$ denotes the highest amount of cycles that is present at any point in the improvement graph during IR-Cycles' execution. Thus the total time complexity of IR-Cycles is $\mathcal{O}(\bar{C} \cdot (V + E \cdot \bar{C}))$. Given that the amount of cycles in a graph can —and in the case of as well-connected a graph as the improvement graph of a typical IR-Cycles run, often *will*—be exponential, it becomes clear that IR-Cycles' runtime complexity is exponential in its input.

3.5. Contextualizing IR-Cycles and WOSMA-like Algorithms

In Section 2.5, we briefly mentioned the Generalized Absorbing Top Trading Cycles (GATTC) family of algorithms, which is designed for housing markets with indifference [4, 15]. It has the weakness of requiring $|F| = |H|$, but perhaps this may be dealt with in the same manner we deal with it in MCPMA.

In this section, we will prove that not all WOSMA-like mechanisms, nor IR-Cycles, belong to this family of algorithms.

GATTC is a relatively straightforward generalization of the well-known TTC algorithm for matching markets to the Matching Markets with Indifference problem. Suppose that, given a matching, we have a digraph $G = (F \cup H, E)$ where for each $f \in F$ and each $h \in H$, $(f, h) \in E$ if h is one of f's maximally preferred houses, and $(h, f) \in E$ if f is currently matched with h. An *absorbing set* of nodes in this digraph is a strongly connected component from which there are no outgoing edges; and two nodes i, j form a *symmetric pair* if $(i, j) \in E$ and $(j, i) \in E$. An absorbing set is paired-symmetric if each node in the set belongs to some symmetric pair. Finally, a *good cycle* is any cycle in G that contains at least one node that does not belong to some symmetric pair. (Correspondingly, a non-good cycle contains only nodes that belong to some symmetric pair.)

Then GATTC mechanisms take the form described in Algorithm 4.

Algorithm 4: GATTC mechanisms as described by Aziz and De Keijzer [4]. As long as the output of some mechanism can alwys be obtained by a procedure of this form, said mechanism is part of the GATTC family.

```
GATTC $(G)$
    Input: $G = (F \cup U, E)$ : Digraph
    while $G$ is not empty do
        Repeat the following a finite number of times on $G$:
            Either implement a non-good cycle (if $G$ is not empty), or do nothing.
            Either remove a paired-symmetric absorbing set (if one exists) and adjust $G$
              correspondingly, or do nothing.
        Repeatedly remove paired-symmetric absorbing sets and adjust $G$ correspondingly,
          until there are none left.
        If $G$ is not empty, implement a good cycle.
```

Theorem 3.5.1. *The WOSMA-like family of mechanisms is not wholly part of the GATTC family of mechanisms, and IR-Cycles is not included in GATTC either.*

Proof. In Section 3.2, we mention that there exists, for any problem instance, some WOSMA-like mechanism that achieves the best achievable (constrained) Pareto optimal outcome; and we saw in Theorem 3.4.1 that IR-Cycles achieves a best achievable Pareto optimal outcome as well.

In the following example, GATTC will inexorably provide one outcome, which is not the best achievable Pareto optimal outcome; this latter outcome, however, is achievable by IR-Cycles and at least one WOSMA-like mechanism.

Suppose that f_i is matched with h_i, and that their fit-scores are as follows:

	h_1	h_2	h_3
f_1	0.0	0.9	1.0
f_2	0.0	0.0	1.0
f_3	1.0	0.0	0.0

Then the corresponding digraph G contains no non-good cycles, nor any paired-symmetric absorbing sets. It only contains the good cycle $f_1 \rightarrow h_3 \rightarrow f_3 \rightarrow h_1 \rightarrow f_1$. So any GATTC mechanism would first implement this cycle, leaving f_2 matched with h_2, for a total score of 2.0. Meanwhile, the best achievable outcome is one with matched pairs $(f_1, h_2), (f_2, h_3), (f_3, h_1)$, which gives a total score of 2.9. Since the only GATTC mechanisms are those that would give the GATTC outcome, and given that IR-Cycles and at least some WOSMA-like mechanism achieve a different, better outcome, our theorem follows. □

The above result highlight an important difference between the Social Housing Market problem and the traditional Matching Markets with Indifference problem: the former uses numerical preferences, whereas the latter uses ordinal ones. Since numerical preferences contain strictly more information than ordinal preferences, algorithms for the latter problem can be trivially translated to our problem (as we have done with WOSMA), but algorithms for our problem cannot be easily translated back to the Matching Markets with Indifference problem. Our example exploits this fact.

We now have a wide variety of algorithms: MCPMA gives us a weakly super-optimal upper bound for the unconstrained problem, which IR-Cycles solves to optimality, and which WOSMA solves efficiently to Pareto-efficiency; finally, for the constrained problem variant, we have modified MCPMA to give us an efficient and locally optimal algorithm in Improvement-MCPMA. With the theoretical background now established, our next task is to research how these algorithms' respective performances compare in practice.

4 Experimental Approach

In Chapter 2 we highlighted two algorithms to run on our problem, and in Chapter 3 we came up with two more algorithms of our own; next comes the step of running these algorithms on sample dynamic matchings to see how well they manage to solve the social housing market problem. In this chapter we describe our various approaches to running these experiments; we moreover provide their results and discuss them. Our main purpose here is to gain an understanding of how well those algorithms perform relative to each other. In particular, we look into: (i) the difference between MCPMA and IR-Cycles AfterSteps, to gain an understanding of how reliably MCPMA may be used as an optimality-bound on the performance of the other algorithms; (ii) whether our locally optimal algorithms reliably perform better, globally, than the locally sub-optimal ones; (iii) what the performance difference is between algorithms designed for the constrained problem variant and algorithms designed for the unconstrained problem variant; (iv) how these algorithms compare in different types of housing markets; and more. Through answering these questions, we hope to come to a better, more nuanced understanding of, firstly, our algorithms' relative performance, and, secondly, how high-quality our solutions to the social housing market problem may be.

In Section 4.1, we discuss how we create data and how we parse said data in order to create a dynamic matching. In Section 4.2 we briefly explain how we modified the house-to-household ratio of these dynamic matchings in order to investigate the influence of house rarity on the algorithms' performance. Then, in Section 4.2.1, we introduce 10 'fitness environments', which vary the fit-values across the matching so as to simulate different types of housing markets (e.g. markets where many houses fit most households vs. markets where good houses are rare for almost all families). Next, in Section 4.3, we introduce six algorithms which we run on these dynamic matchings. These include the algorithms we have seen before, plus one new, basic heuristic called 'Naive.' With all the necessary information in place, we describe our main questions and hypotheses in Section 4.4, and next describe how our experiments were run in Section 4.5. Finally, we provide our experiments' results in Section 4.6 and discuss them in Section 4.7.

4.1. Data Generation

The basis for our data generation method is a database from Woningnet, obtained through Ymere. The database details moves that have taken place within a period of three months, listing families and their new houses.

We have performed manual data pre-processing in the form of removing irrelevant columns, so that all information currently contained in the data is, if not relevant to the model in its current form, at least integrated into the mathematical model of the situation in some way.

We use a simplistic method to generate data. Starting at the first column, we select a value according to a probability distribution such that, if some value x occurs under this column in p% of the entries, it has a p% chance of being selected here. Then the inductive step: For some column c, given some value x_c, the next value x_{c+1} is selected according to a procedure similar to the one above, except we look not at the entire column $c + 1$, but rather only at those entries in $c + 1$ whose values in column c equal x_c.

Column name	Translation	Explanation (when necessary)
gemeente woning	municipality of living space	
label woning	house label	e.g. ”youth housing”, ”regular housing”, etc.
subsidiabele huur	subsidy-eligible rental price	
huurklassen	rent class	e.g. ”cheap”, ”payable”, ”expensive”, etc.
aantal kamers	amount of rooms	
verdieping	floor	the floor on which the living space is located.
lift	elevator	whether the living space is reachable by elevator.
gemeente wz	(former) municipality of household	
postcode wz	(former) postal code of household	
label wz	label of household	e.g. ”newcomer to the market”, ”leaving an independent residence”, etc.
inkomen totaal	total income	annual income.
inkomensklassen	income class	e.g. ”low”, ”mid”, ”high”, etc.
leeftijd verhuring	age of tenant	
leeftijdsklassen verhuring	age class of tenant	e.g. ”26-34”, ”55-65”, etc.
huishoudentype	household type	e.g. ”2 people”, ”1 person with child”, etc.
aantal personen	amount of people	
voorrang wz	priority status of tenant	e.g. empty, ”starter”, ”social-medical urgency”, ”divorcee”, etc.

Table 4.1: The columns included in a dataset output by our data generation process.

In this way, every pair (c_i, c_{i+1}) of columns is realistically correlated, but no full new entry was necessarily present in the original data.

Our sample output dataset’s columns may be viewed in Table 4.1. Of these, the following attributes are taken into account:

- For houses:
  1. The living space’s monthly (subsidy-eligible) rental price.
  2. The living space’s amount of rooms.
  3. The floor on which the living space is located.
  4. Whether the living space is reachable by elevator.
- For families:
  1. The family’s annual income.
  2. The age of the person who is renting.
  3. The amount of people that the family contains.

This data is then turned into a set of houses and households containing these attributes; for a given (house, family)-pair, their attributes are turned into a fit score through the calculation described in Section 2.4. Since the original dataset delivers houses and families in (house,family)-pairs describing that said family that has moved to its paired house, these matches are retained in our dynamic matching’s initial state for those houses and families that are present in said state.

4.2. Environments

It is important that we test our algorithms on various different housing markets. Thus the bipartite graphs that we generate differ on two dimensions: The base size, representing the number of houses and households that are initially added, and a modifier $envRatio$, representing the ratio of houses to households. If a graph's $envRatio$ indicates a surplus on either side, this surplus is then removed from the graph. For example: A graph with size 10 and $envRatio = 0.5$ would have five of its houses removed, resulting in a graph where $|H| = 5$ and $|F| = 10$.

We use the following $envRatio$-values to represent different types of markets, from ones where there are far too few houses to ones where there are more than enough for everyone: $\{0.50, 0.75, 1.00, 1.25, 1.50\}$. The situation where $envRatio = 0.5$, where there are twice as many families as there are houses, is of particular interest to housing corporations such as Ymere, who often deal with a great surplus of households who want to move into a city.

4.2.1. Fitness Environments

To further diversify our housing markets on top of the different sizes and $envRatio$-values, we incorporate 10 'fitness environments' into our experiments. Here, different fitness environments express different ways to create a distribution of fit-scores in a housing market, so that different types of housing markets may be compared. In Section 2.4, we already brought up two different fitness environments: fit_{avg} and fit_{min}, which based themselves on realistic data. To these we add eight other fitness environments, which are represented by different ways to mathematically generate fit-values. In all cases, the fitness environment is only used to calculate scores between individual houses and families, and may thus be viewed as being fit-functions in their own right; the overall matching score is still calculated using the $eval_{avg}$-function we saw in Section 2.4, which simply averages over the calculated fit-score of all households in the given matching.

8 of these fitness environments draw fit-values from statistical distributions. For these, we include both a regular version and a 'constrained' version. The constrained versions use the same samples as the regular versions, but round these sampled values to those values attainable by fit_{avg}, namely $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$. This is done because in reality, family's valuations of houses may be very multifarious, and are not necessarily well-represented by a function that can only return one of four values. The difference between these regular fitness environments and their constrained counterparts, which similarly reduce the complexity of the regular environments to these same four values, may give some indication of how our model choice in fit_{avg} influences the results we get.

1. MatchingAVG: this is fit_{avg}. This fitness environment operates on the semi-realistic data described in Section 4.1, and thus represents a somewhat realistic housing market in which families would be willing, for instance, to move from a house that fits them badly to a house that fits them imperfectly, but better nonetheless.

2. MatchingMIN: this is fit_{min}. Like MatchingAVG, this fitness environment operates on our semi-realistic data, but since the fit here is binary—either the house is a complete fit, or it is unacceptable—it represents a more difficult market where families are only willing to move to houses that fit them in every way.

3. NormalDistLowVar & NormalDistLowVarConstrained: these fitness environments represent a market in which most houses are a moderate fit for most families. Here, the fit between any house and any household is a precomputed value that is sampled from a normal distribution where $\mu = 0.5$ and $\sigma^2 = \frac{1}{18}$. Those few values that are not contained within the range $[0, 1]$ are set equal to whichever limit they exceeded.

4. NormalDistHighVar & NormalDistHighVarConstrained: these fitness environments represent a market in which the houses are, on average, a moderately good fit for families, but there exist larger differences in the distribution of these scores. These environments proceed in the same manner as their LowVar-counterparts, except here we use a normal distribution with $\mu = 0.5$ and $\sigma^2 = \frac{1}{6}$ for a wider spread of values.

5. ExpDistLowLambda & ExpDistLowLambdaConstrained: these fitness environments represent a market in which the fit-values are sampled from an exponential distribution with $\lambda = 1$, meaning that about a third of the fit-values are 1, and the rest are about evenly divided within $[0, 1)$. In other words, a lot of houses are a great fit for a lot of families, and the rest is distributed approximately randomly. As in the normal-distribution fitness environments, we constrain all values to the range

$[0, 1]$. To give the reader an idea of what an exponential distribution with $\lambda = 1$ looks like, we have provided sample distributions in Fig. C.1.

6. ExpDistHighLambda & ExpDistHighLambdaConstrained: these fitness environments represent a market in which the fit-value follows a 'proper' downwards exponential curve where $\lambda = 5$, with the vast majority of values being very low. In other words, most houses here are a very poor fit for most families, and it's quite rare for a household to find a house that fits them well. Aside from the λ-value, these environments are defined analogously to their $\lambda = 1$-counterparts. The reader may find sample distributions in Fig. C.1.

4.3. Algorithms

We have implemented six different algorithms so that we may compare their performances on the social housing market problem: a new but very basic heuristic which we call 'Naive'; a slightly modified version of the Minimum-Cost Perfect Matching Algorithm [14]; two more heavily modified versions of the Worker-Optimal Stable Matching Algorithm [6]; the Improvement-MCPMA algorithm mentioned in Section 3.3.4; and IR-Cycles from Section 3.4.

We noted in Section 2.4 that there are two approaches to 'running' a dynamic matching, namely **PerStep**, which runs the algorithm at every timestep, and **AfterSteps**, which runs the algorithm once after the dynamic matching has 'run its course'.

An overview of all six algorithms, including whether we compare their performance on both the PerStep and the AfterStep settings, is available in Table 4.2.

4.3.1. Naive

Naive is a very basic heuristic suitable for the constrained problem variant. At each timestep in a given dynamic matching, Naive simply goes through all families in some random order and assigns each of them them some house they most prefer, from those houses that were empty at the start of the current timestep—so long as the family prefers said house to their current house. It is thus similar to the well-known Random Serial Dictatorship mechanism, except families may already own houses here. It is meant to represent the ad-hoc decision-making heuristic that might be used by a social housing corporation that does not use algorithms, and instead at any timestep tries to 'naively' find, for all families, a house that best fits them, amongst those that are empty at the start of that timestep and that have not yet been allocated to another family. As such, we only run it in the PerStep setting. Alternatively, it also serves as a rudimentary model of a market in which houses are allocated based on a waiting-queue, in which the random order in which families are chosen, is analogous to an ordering of these families on a waiting list: the earlier a family is chosen to pick their favourite house, the higher up they are on the waiting list.

4.3.2. Minimum-Cost Perfect Matching Algorithm

We base our algorithm on the Minimum-Cost Perfect Matching Algorithm given by Kleinberg and Tardos [14], which we described in Section 2.5.1. We apply two modifications, both of which have been described in more detail in Section 3.3.4: namely, we add dummy houses or households to enable MCPMA to work with matchings where $|H| \neq |F|$, and we redefine the edges' weights so as to make MCPMA minimize $\sum_{(h,f)\in M}(1 - fit(h, f))$. As noted in Section 2.5.1, MCPMA is run only in the AfterSteps scenario, since it provides a non-IR solution that we use merely as a global benchmark.

4.3.3. Worker-Optimal Stable Matching Algorithm

We base our algorithm on the Worker-Optimal Stable Matching Algorithm (WOSMA) given by Erdil and Ergin [6], which we described in Section 2.5.2. Our version has some substantial differences to make the algorithm more realistic, however. For example: in the original paper, the algorithm's input matching is necessarily stable, being the output of the well-known Deferred Acceptance algorithm, which produces stable matchings. In our model, however, the DA-algorithm does not apply, since it relies on both sides having preferences, rather than just one. Thus we skip the step of running DA before WOSMA. As a result of this, despite WOSMA's name, we cannot ourselves guarantee stability. However, the original notion of stability does not apply either way: WOSMA was designed for two-sided markets with indifference, but we reduce the mechanism to work on one-sided markets with indifference simply by making all houses indifferent over all families. Only the original guarantee of Pareto optimality remains, and this we trivially retain.

The original WOSMA mechanism selects cycles at random, and allows for families to move to houses they do not strictly prefer; of course, this would never happen in practice. We are moreover curious if a different cycle-selection strategy might yield improved results. Thus we present two custom versions of WOSMA: **Regular**, which selects some cycle in an effectively random manner while guaranteeing that all families that are moved, are getting a house they strictly prefer to their original house; and **FindMax**, a WOSMA-like mechanism which tries to find and carry out, at every step, some 'optimal' cycle. Both these versions are still accurately described by Algorithm 3 (besides the aforementioned Deferred-Acceptance step). However, they differ in what their two-labeled graphs look like, what kinds of cycles they accept, and how they find these cycles.

Here we give an overview of the modifications we have made:

1. **Regular.** The creation of a Two-Labeled Graph goes as described in Algorithm 3. Finding a cycle, meanwhile, happens as follows. First we run Gabow's path-based algorithm [8] for identifying strongly-connected components of TLG_M. Then, going through components of size > 1 until we have found some cycle, we run a simple Depth-First-Search algorithm[1] on the component currently being analyzed, that distinguishes between three node states: *undiscovered*, *being explored*, *explored*. However, we only traverse edges that are either fully strict, or whose source node is either $\emptyset$, or some household w that has been part of some cycle that has been previously executed within this WOSMA-call. As soon as we find a cycle, we execute it in our matching M_G.

   In this manner, it is guaranteed that all households that are moved, are moved to some house they **strongly** prefer to their initial house; but moreover, since any real-world changes would only occur after the end of the algorithm's run, it is the case that after we've moved them to a better house in the matching, we are free to relocate them (within the same timestep) to any other house that is at least as good, thereby potentially allowing for more cycles and chains to occur.

2. **FindMax.** Our creation of the two-labeled graph TLG_M begins similarly as in Regular, but we add different edges in a different manner, and with different non-binary weights. (In the case of FindMax, 'Two-labeled graph' is a misnomer; we keep the name for consistency's sake.)

   (a) $w \to v$. These we include whenever some family w *strictly* prefers some other family v's house over their own. This edge gets a weight of $fit(w, M_G(v)) - fit(w, M_G(w))$, that is, equal to the 'strength' of their preference for the other house.

   (b) $w \to \emptyset$. These we include whenever some family w prefers some empty house over their own. If h is the empty house they most strongly prefer, then this edge is assigned a weight of $fit(w, h) - fit(w, M_G(w))$.

   (c) $\emptyset \to w$. Same as in Regular.

   The edges' weights are chosen such that, as long as there exists some Pareto-improving action that may be taken, it corresponds to a positive-sum cycle in this graph.

   FindMax then runs the Szwarcfiter-Lauer algorithm [18] to identify all cycles in TLG_M. Of these, it picks some cycle among those with the highest sum of weights along their edges (which represents these cycles would cause the highest 'improvement' for the matching), is at least as short as any other cycle amongst these, and executes it in our matching M_G.

WOSMA Regular and WOSMA FindMax do not differ meaningfully in their practical running times. We shall use WOSMA Regular as our baseline strategy, representing the strategy of a social housing corporation in the unconstrained problem variant.

[1]This DFS-algorithm has been modified slightly, since unlike in standard DFS, in our Regular WOSMA, it is possible to come across a node that has been marked 'explored' while exploring some other node. For example: suppose we have explored some node w_1 along all of its *valid* edges. Then it might still have some invalid edge $w1 \to w2$, which was not traversed because the edge does not fulfill our condition (namely that it be either strict, sourced at $\emptyset$, or sourced at some household that has previously moved). However, there might be a sequence of valid edges $w_2 \to w_3 \to w_1$. Thus we might find w_1 while exploring w_2. This is nothing to worry about. However, is is the case that some cycle $w_2 \to w_3 \to w_1$ does not constitute a valid cycle, because $w_1 \to w_2$ is invalid. (After all, if it were not, we would have explored w_2 already.) Moreover, since w_1 has been fully explored recursively along all valid outgoing edges, we will not find any valid cycle here. (After all, any cycle $w_2 \to w_1 \to \cdots \to w_2$ must include at least some invalid edge; otherwise we would have explored w_2 already.)
Thus when we encounter an explored node during exploration, we simply ignore it.

Name	Problem Variant	Advantages	Disadvantages	Dynamic?
Naive	Constrained	Simple, represents ad-hoc decision-making	No guarantees	No
Improvement-MCPMA	Constrained	Efficient & locally optimal	N/A	No
MCPMA [14]	Unconstrained	Weakly super-optimal & efficient	Not IR	No
WOSMA [6] Regular	Unconstrained	Efficient, IR	Not locally optimal	Yes
WOSMA [6] FindMax	Unconstrained	IR, tries to make good decisions	Not locally optimal	Yes
IR-Cycles	Unconstrained	IR, locally optimal	Inefficient	Yes

Table 4.2: An overview of the algorithms we compare in our experiments.

4.3.4. Improvement-MCPMA

The Improvement-MCPMA mechanism proceeds exactly as described in Section 3.3.4. Like Naive, Improvement-MCPMA is designed for the constrained problem variant, in which chains and cycles are presumed to be difficult to organize. We exclude Improvement-MCPMA from our AfterSteps experiments, since, as noted in Section 3.3.4, it does not translate well to the dynamic setting.

4.3.5. IR-Cycles

Our implementation of IR-Cycles proceeds as described in Section 3.4, using a modified version of the Szwarcfiter-Lauer Szwarcfiter and Lauer [18] for finding simple cycles in a graph, to search the graph for cycles which are individually rational and which, if carried out, would increase the matching's average fit. Our adaptation of their algorithm does not find all cycles; instead, as soon as it finds one cycle that satisfies these requirements, it carries it out in the matching.

4.4. Questions & Hypotheses

Having explained all necessary information about our algorithms and the various configurations that we run them in, we here describe our questions and hypotheses.

4.4.1. Questions

In Chapter 1, we brought up two central research questions: how might the housing market be improved by different strategies, and how much can it be improved to begin with? There are moreover a number of auxiliary questions which we have introduced along the way, namely: What is the difference in potential outcome matching quality between the constrained and the unconstrained problem variants? How large are the benefits our algorithms may reap from the dynamic setting using perfect future knowledge (i.e. AfterSteps) vs. when they have no such knowledge (i.e. PerStep)? How are the algorithms' respective performances influenced by different values of $envRatio$, different graph sizes, and different fitness environments? Here we make these questions more exact, so that we may experimentally answer them:

1. How much can the housing market be improved by different strategies?

   Q.1a To answer the question of how much a housing market may be improved, we need a reliable upper bound. We have one in IR-Cycles in the AfterSteps scenario, but it is too slow to run on large problem instances, whereas MCPMA is very efficient but gives a weakly super-optimal solution. To assess how reliable MCPMA is as an upper bound, we need to know the following: how close does IR-Cycles AfterSteps' score come to the MCPMA's?

2. What is the difference in potential outcome matching quality between the constrained and the unconstrained problem variants?

   Q.2a WOSMA Regular, WOSMA FindMax, and IR-Cycles all relate to the unconstrained problem variant, whereas Naive and Improvement-MCPMA specifically apply to the constrained problem variant. How much worse are our matchings when we may not take chains and cycles than when we may, and how is this affected by the environments and fitness environments we use?

3. How well do our different strategies perform?

   Q.3a Given that both Naive and Improvement-MCPMA are strategies for the constrained problem variant, under what circumstances does which perform better?

   Q.3b Given that IR-Cycles is locally optimal, but is not guaranteed to be globally optimal in the PerStep scenario, how does it fare in PerStep against WOSMA Regular and WOSMA FindMax?

   Q.3c Which performs better under which circumstances: WOSMA Regular or WOSMA FindMax?

4. How large are the benefits our algorithms may reap from the dynamic setting using perfect future knowledge (i.e. AfterSteps) vs. when they have no such knowledge (i.e. PerStep)?

   Q.4a How do WOSMA Regular, WOSMA FindMax, and IR-Cycles differ in their performance in PerStep versus AfterSteps?

5. How are our algorithms' performances influenced by different values of $envRatio$, different graph sizes, and different fitness environments?

   Q.5a How are the scores impacted when, in any of the normal or exponential distribution-based fitness environments, we move from their regular variants to their constrained counterparts?

   Q.5b Similarly to question Q.5a: How do the algorithms' performances change as we move from MatchingAVG (which utilizes four possible fit-scores) to MatchingMIN (which utilizes a binary fit-score)?

   Q.5c How are the algorithms' scores affected as we move from normal distributions with a low variance, to ones with a high variance?

   Q.5d Similarly to question Q.5c: How are the algorithms' scores affected as we move from exponential distributions with a low λ-score, to ones with a high λ-value?

   Q.5e How do the algorithms fare as $envRatio$ decreases from 1.50 to 0.50?

   Q.5f How are the algorithms' scores affected as the matchings increase in size?

4.4.2. Hypotheses

Prior to running our experiments, there were a number of things we expected to see:

H.1 MCPMA will score quite a bit better than IR-Cycles AfterSteps in every configuration, since the latter suffers from having to ensure Individual Rationality. (This hypothesis relates to question Q.1a.)

H.2 As $envRatio$ decreases, the requirement that algorithms guarantee Individual Rationality will constrain their action spaces more strongly; thus we should see their scores decrease relative to MCPMA. (This hypothesis relates to question Q.5b.)

H.3 As a result of the above, and taking IR-Cycles' optimality into account, we moreover expect that the gap between IR-Cycles in AfterSteps and the other algorithms' performances will grow as $envRatio$ decreases. (This hypothesis relates to question Q.5b as well.)

H.4 Algorithms' scores should increase as the graph size grows (within a single $envRatio$-value), since more possibilities in the market mean a higher chance of families finding a house that fits them well. MCPMA benefits from this as well, but less so, since it has the largest action space to begin with. Thus we expect the algorithms scores to increase relative to MCPMA as the size grows. (This hypothesis relates to question Q.5f.)

H.5 WOSMA FindMax should, on average, perform quite a bit better than FindMax Regular, since the former tries to find and effectuate 'particularly good' cycles at every iteration, whereas the latter simply picks random cycles each time. (This hypothesis relates to question Q.3c.)

H.6 Improvement-MCPMA gets a locally optimal solution within its limited action space; thus it should perform better than Naive, which tries to greedily optimize the matching ad-hoc. (This hypothesis relates to question Q.2a.)

H.7 WOSMA FindMax should consistently perform better in AfterSteps than in PerStep[2], since it has access to better cycles in the former configuration. (This hypothesis relates to question Q.4a.)

H.8 For the fitness environments that are based on a normal or an exponential distribution, we expect that the average score of the algorithms will increase as we go from the constrained versions to the regular versions. We expect this due to the fact that the higher fidelity of fit-values in the regular versions gives the algorithms more information to make decisions with than the constrained variants do, as a result of which the algorithms should be able to make slightly better decisions. (This hypothesis relates to question Q.5a.)

H.9 We expect algorithms to perform better in the high-variance normal-distribution fitness environments than in the low-variance ones, since a high variance will mean the average fit-value for families with houses that they prefer to their current house, increases, which the algorithms should be able to make good use of. However, we expect that MCPMA can make even better use of this, and thus we expect the algorithms scores relative to MCPMA to drop slightly in going from the low-variance normal-distribution fitness environments to the high-variance ones. (This hypothesis relates to question Q.5d.)

H.10 We expect most algorithms to perform quite a bit worse in the exponential fitness environments with a high λ-value than in those with a low λ-value. This is because, as can be seen in Fig. C.1, good fits are very rare in exponential distributions with a high λ-value; thus we expect that only MCPMA and IR-Cycles AfterSteps can maneauver themselves towards those rare outcomes where the average fit-score is highest, whereas for the other algorithms, these global maxima should be much harder to find. (This hypothesis relates to question Q.5e.)

4.5. Experimental Setup

With our questions and hypotheses in place, we now describe the experiments we ran to shed light on them.

As noted in Section 4.3, we compare six algorithms (see Table 4.2), which differ in their performance, their speed, and their corresponding problem variants (with WOSMA Regular, WOSMA FindMax, and IR-Cycles sharing one problem variant, and Naive and Improvement-MCPMA sharing another).

We compare the performance of WOSMA Regular, WOSMA FindMax, and IR-Cycles on both PerStep and AfterSteps. The others, as discussed in their respective sections, are only run in the PerStep setting.

The reasoning behind our comparing PerStep with AfterSteps, as noted in Section 2.4, is that we want to see how well each algorithm makes use of the dynamic nature of the market, as well as how well an algorithm would perform if we had perfect foresight. A large difference between AfterSteps and PerStep suggests that the algorithm in question may not be well-suited to a dynamic environment, though if its AfterSteps score is particularly high, then provided the social housing corporation in question has some knowledge of upcoming changes in the market's supply of houses and families, it might be a good strategy to pursue after all.

As noted in Section 4.4, this difference between PerStep and AfterSteps is especially relevant in the case of IR-Cycles, which performs optimally; thus, whatever advantage may be gained by having perfect foresight, we should be able to find it in the difference between the results IR-Cycles achieves in PerStep and the results it achieves in AfterSteps.

For each $envRatio \in \{0.50, 0.75, 1.00, 1.25, 1.50\}$ and each of the 11 fitness environments we named in Section 4.2.1, we run all six algorithms in the PerStep-configuration, as well as the three specified above in the AfterSteps-configurations. Each algorithm we run on 50 different matchings created using a base size (i.e. yet unmodified by $envRatio$) in $\{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25,$

[2]The weak version of this statement is a mathematical fact for IR-Cycles. Furthermore, we have no hypotheses on this topic regarding FindMax Regular's behaviour, since it picks random cycles at all times; and though AfterSteps increases the action space, we do not expect it to improve the average action within that space, relative to PerStep.

$30, 35, 40, 45, 50, 75, 100\}$, after which they are modified by the corresponding $envRatio$-value as described in Section 4.2. These matchings are then made dynamic in the manner described in Section 2.4 through the addition of a $tCount$-variable, which is equal to $\frac{2}{3} \cdot \min(|H|, |F|)$—a value chosen so as to allow for a large difference between the initial matching and its final, 'full' state, while still providing a sufficiently large 'ground-level' matching that new houses and households are integrated into. Thus, for example, each dynamic matching where $envRatio = 1$ starts at one-third the size of what it will eventually become.

Going through these base sizes from low to high, we let each algorithm run (both PerStep and, where applicable, AfterSteps) on the same corresponding fifty dynamic matchings until it took longer than 60 seconds[3] to complete; thus slower algorithms did not reach the highest size. Finally, for each size, we average the final score of the matching as a whole. Using this average, we calculate for each algorithm except MCPMA the ratio between its final average score and the average score achieved by the weakly super-optimal MCPMA, so that the final result indicates how close the algorithm came to optimality. Of course, this measure is imperfect, since MCPMA might at times be super-optimal. Fortunately, the much slower IR-Cycles' AfterSteps scores shows, at least for lower sizes, the extent to which MCPMA's scores exceeds optimality, and thus may suggest how accurate this optimality ratio measurement is.

4.6. Experimental Results

Here we provide the reader with the results of the experiments described in Section 4.5. The results for MatchingAVG and MatchingMIN may be found in Fig. 4.1. The results for NormalDistLowVar, NormalDistLowVarConstrained, NormalDistHighVar, and NormalDistHighVarConstrained may be found in Fig. 4.2. Finally, the results for ExpDistLowLambda, ExpDistLowLambdaConstrained, ExpDistHighLambda, and ExpDistHighLambdaConstrained are given in Fig. 4.3.

It now remains to discuss whether our hypotheses from Section 4.4 still hold in practice, and what answers to the questions that we raised there, are implied by these results.

4.7. Discussion

Here we answer the questions we raised in Section 4.4 using the results we provided in Section 4.6, verifying our hypotheses in the process. Finally, these insights are condensed into a general answer to the main research question we raised in Chapter 1: what strategies could a housing corporation use to improve the housing market, and how much may a housing market be improved in the first place?

In answering these questions, when we talk about an algorithm's performance, we are referring to how well it scores on the objective function defined in Section 2.4, relative to MCPMA. In other words: the more closely the average fit for all families in the matching approximates the (super-)optimum defined by MCPMA, the better the algorithm performed.

A.1 Question Q.1a: The performance of IR-Cycles Aftersteps is consistently quite close to MCPMA's performance, being often 0.95 times as high, and never less than 0.90 times as high. However, the low amount of sizes that we could test IR-Cycles AfterSteps on means it is hard to tell how its performance would fare in larger graphs. Thus we cannot give a final answer to this question, but the data we have suggests that MCPMA provides us with a quite trustworthy upper bound on this problem. This weakly disproves our hypothesis H.1, we we thought the difference would be larger.

A.2 Question Q.2a: There is a clear pattern in the data of either Improvement-MCPMA or Naive—and often both—underperforming relative to the other algorithms. This gap can grow quite large, especially in exponential fit-distributions. If we constrain ourselves to environments where $envRatio = 0.5$, however, then Improvement-MCPMA often performs quite close and, at times, even better than WOSMA Regular, and performs about 0.89 times as well as WOSMA FindMax (both PerStep and AfterSteps). Thus we estimate that in a market like this, being unable to carry out chains (larger than 1) and cycles has a limited effect on the market's quality.

[3]This limit was chosen so as to get enough results for all our different configurations in a short enough timespan. In practice, the algorithm would likely get several days to run, so that it may inform decision-making on a weekly basis.

A.3 Question Q.3a: There is a clear and surprising pattern in the data, showing a large correlation between the relative performance of Improvement-MCPMA and Naive, and the value of $envRatio$. Specifically, in markets with too few houses, Improvement-MCPMA tends to perform much better than Naive, but in markets with many more houses than families, Improvement-MCPMA consistently underperforms greatly compared to every other algorithm, including Naive. In other words, the pattern we predicted in hypothesis H.7 is verified for low $envRatio$-values, but it is completely wrong for high $envRatio$-values. One possible explanation for this is that locally optimal strategies such as Improvement-MCPMA are more valuable when local optima are rarer and thus harder to find, which is the case when $envRatio$ is low; whereas in situations where there are plenty of options available for families, a greedy approach like Naive works well.

A.4 Question Q.3b: IR-Cycles PerStep consistently performs at least as well, and often better, than WOSMA FindMax PerStep, and it performs vastly better than WOSMA Regular in almost all cases.

A.5 Question Q.3c: In the majority of situations, and particularly when looking at PerStep, WOSMA FindMax outperforms WOSMA Regular. Thus hypothesis H.5 is validated.

A.6 Question Q.4a: IR-Cycles PerStep's performance is consistently close, but rarely equal, to its AfterSteps' performance. The biggest differences seem to take place when ExpDistHighLambda is the fitness environment being used. The same broadly holds for WOSMA Regular and WOSMA FindMax, with FindMax exhibiting the largest gap on average.

A.7 Question Q.5a: For most distribution-based fitness environments, moving from the regular to the constrained variant increases the score gap between algorithms' PerStep and AfterSteps performances. However, in the case of the exponential distribution with $\lambda = 5$, we see the opposite effect. On the whole, this change has slight effects, but no consistent patterns are discernable. As a whole, the effects do not seem strong enough to validate hypothesis H.8, but the results *are* weakly consistent with it.

A.8 Question Q.5b: MatchingMIN has an only moderate negative effect on most algorithms' scores relative to MCPMA, except for Improvement-MCPMA, whose relative performance decreases strongly. However, when $envRatio = 0.5$, this effect is similar to that suffered by WOSMA FindMax.

A.9 Question Q.5c: A switch from normal distributions with a low variance to ones with a high variance, has lightly negative effects on most algorithms' performance. This is consistent with hypothesis H.9.

A.10 Question Q.5d: In most unconstrained configurations where we move from an exponential distribution with $\lambda = 1$ to one where $\lambda = 5$, we see that the gap between algorithms' PerStep scores and their AfterSteps scores is increased quite greatly. However, the opposite is true for the constrained distributions. Surprisingly, the effect we predicted in hypothesis H.10 is only weakly present.

A.11 Question Q.5e: We saw one example effect of a change of $envRatio$ in our answer to question Q.2a. Apart from this, a higher $envRatio$ makes WOSMA Regular, WOSMA FindMax, and IR-Cycles all perform much better. WOSMA Regular sees the biggest benefit here, though, as discussed in our answer to question Q.3c, it also has the most room to improve. We see the effect we expected to see in hypothesis H.2: as $envRatio$ decreases, the gap between most algorithms' performance and that of MCPMA increases. The effect we hypothesized in hypothesis H.3 is also present. However, Improvement-MCPMA is mostly exempt from the effects hypothesized here; as discussed in our answer to question Q.3a, its performance grows closer to that of the other algorithms as $envRatio$ decreases.

A.12 Question Q.5f: There is no clear general pattern as to what effect the graphs' size has on the algorithms' scores, except within fitness environments. In the two semi-realistic fitness environments as well as in the exponential distribution-based ones, there is a very weak upwards trend. The normal distribution-based fitness environments, meanwhile, mostly show a constant trend, except for NormalDistHighVarConstrained, in which the algorithms' performance shows a light downwards trend. Hypothesis H.4 is mostly refuted: most algorithms' performance does indeed increase between sizes 5 and 20 in most situations, but further size increases have basically no effect.

Our two main research questions, which we raised in Chapter 1, were: **what strategies could a housing corporation use to improve the housing market, and how much may a housing market be improved in the first place?**

As noted in Section 4.3.1, we use Naive's results as our baseline representing the usual housing corporation strategy in the constrained problem variant, where it is assumed to be impractical for housing corporations to carry out chains and cycles; and as mentioned in Section 4.3.3, we use WOSMA Regular's results as our baseline for the unconstrained problem variant in which housing corporations do have the freedom to carry out chains and cycles. Moreover, as motivated in Section 4.2, we primarily focus on the $envRatio = 0.5$-scenario for practical advice.

Since we lack a globally optimal method that adheres to the step-wise constraints of Improvement-MCPMA and Naive, we cannot give a resolute answer to the question of how much a housing market may be improved when chains and cycles are not allowed. However, taking IR-Cycles as our upper bound, then our results, as noted in answers A.1 and A.2, suggest with some certainty that Improvement-MCPMA (the dominant constrained strategy when $envRatio = 0.5$) performs 0.87 times as well as IR-Cycles AfterSteps does; thus the actual difference in practice is likely no larger than this. Moreover, as described in answer A.3, a performance increase of 29% may be gained when $envRatio = 0.5$ by switching from Naive to Improvement-MCPMA.

If cycles and chains *are* allowed, then our results, as described in answers A.1, A.4 and A.5, suggest that WOSMA FindMax (PerStep) can gain a performance increase of 15% over WOSMA Regular (PerStep), and gives us, in practice, a very nearly optimal strategy.

In all cases, however, as discussed in answer A.8, MatchingMIN, where houses are quick to be completely unacceptable for many families, stands out as showing a fairly high difference between optimality (through IR-Cycles) on one side, and the performance of all other algorithms on the other. Here the optimality ratio of WOSMA FindMax PerStep is reduced to 86%, and WOSMA Regular PerStep's becomes 82%; moreover, Improvement-MCPMA here does not perform meaningfully better here than Naive, as they reach optimality ratios of 80% and 81%, respectively, in this scenario.

Thus our final answer to our research question is as follows: Given a realistic setting, we may improve a constrained housing market with 29% by switching from the Naive to the Improvement-MCPMA strategy, and switching from WOSMA Regular to WOSMA FindMax in an unconstrained housing market can yield a 15% performance increase, resulting in markets that are close to optimal.

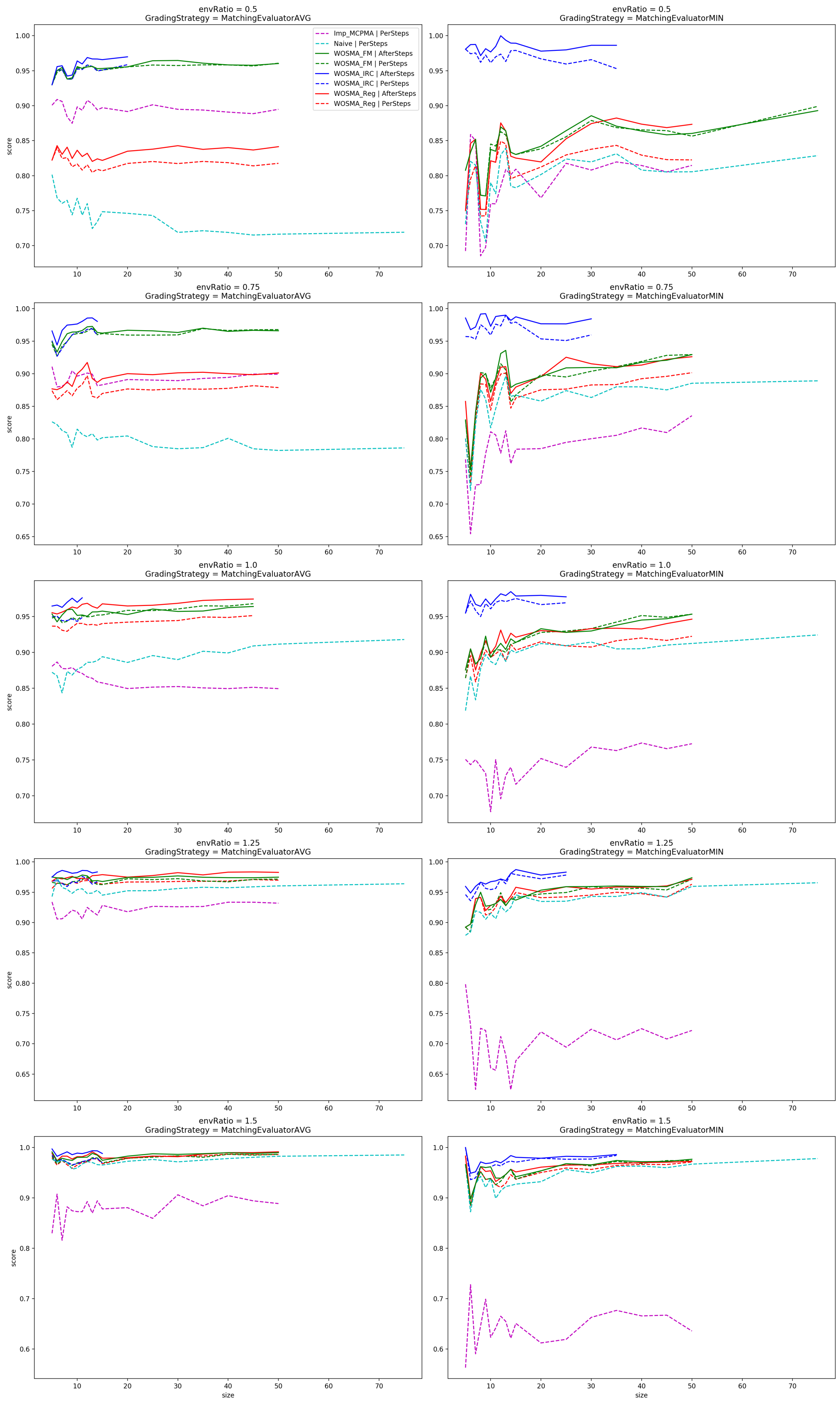


Figure 4.1: Experimental results for the fitness environments MatchingAVG and MatchingMIN. 'Score' denotes not the algorithms' absolute scores, but rather their scores relative to MCPMA. For clarity's sake, sizes above 75 have been excluded, since often Naive was the only algorithm to make it that far within the time available.

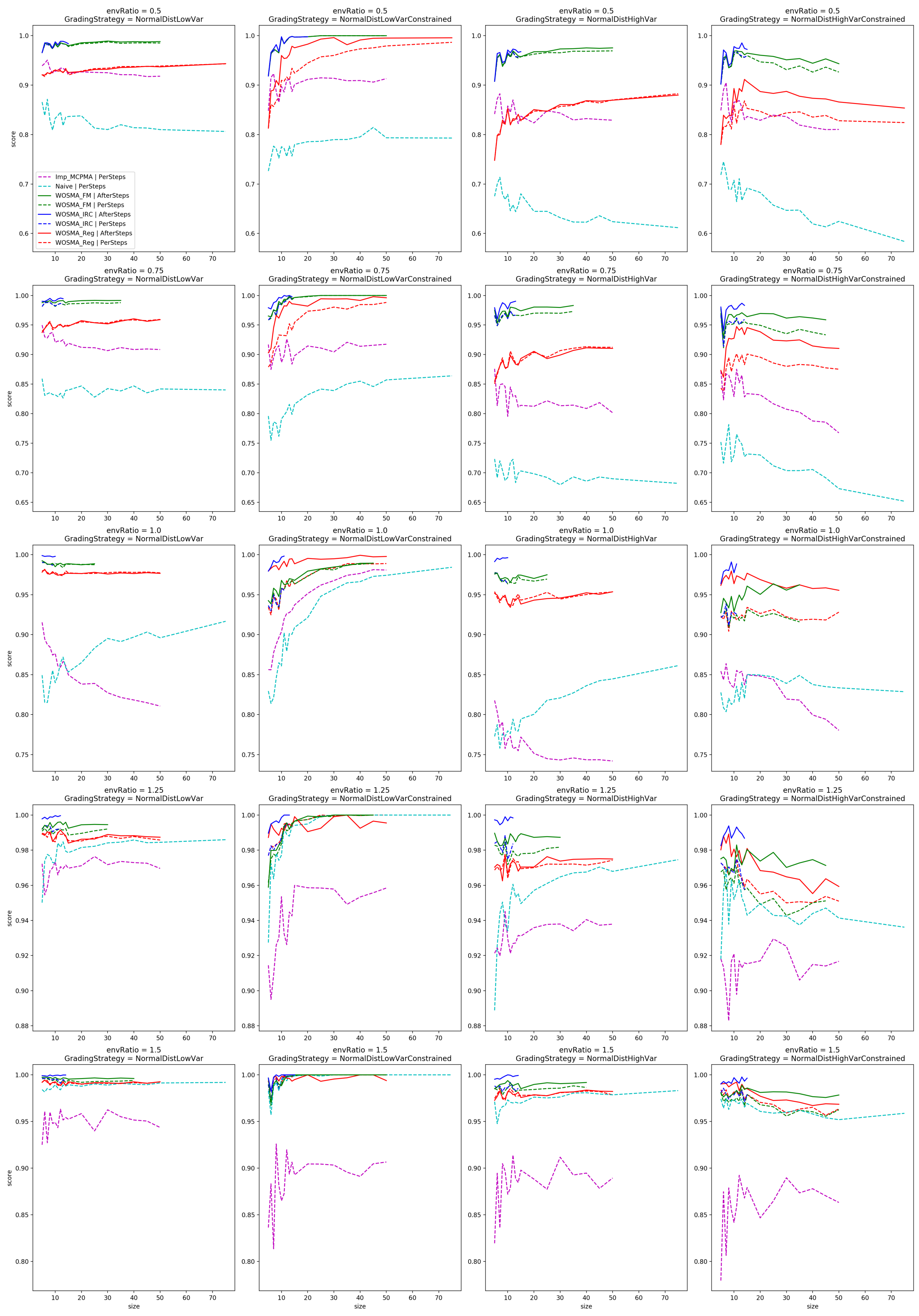


Figure 4.2: Experimental results for the fitness environments based on normal distributions. ‘Score’ denotes not the algorithms’ absolute scores, but rather their scores relative to MCPMA. For clarity’s sake, sizes above 75 have been excluded, since often Naive was the only algorithm to make it that far within the time available.

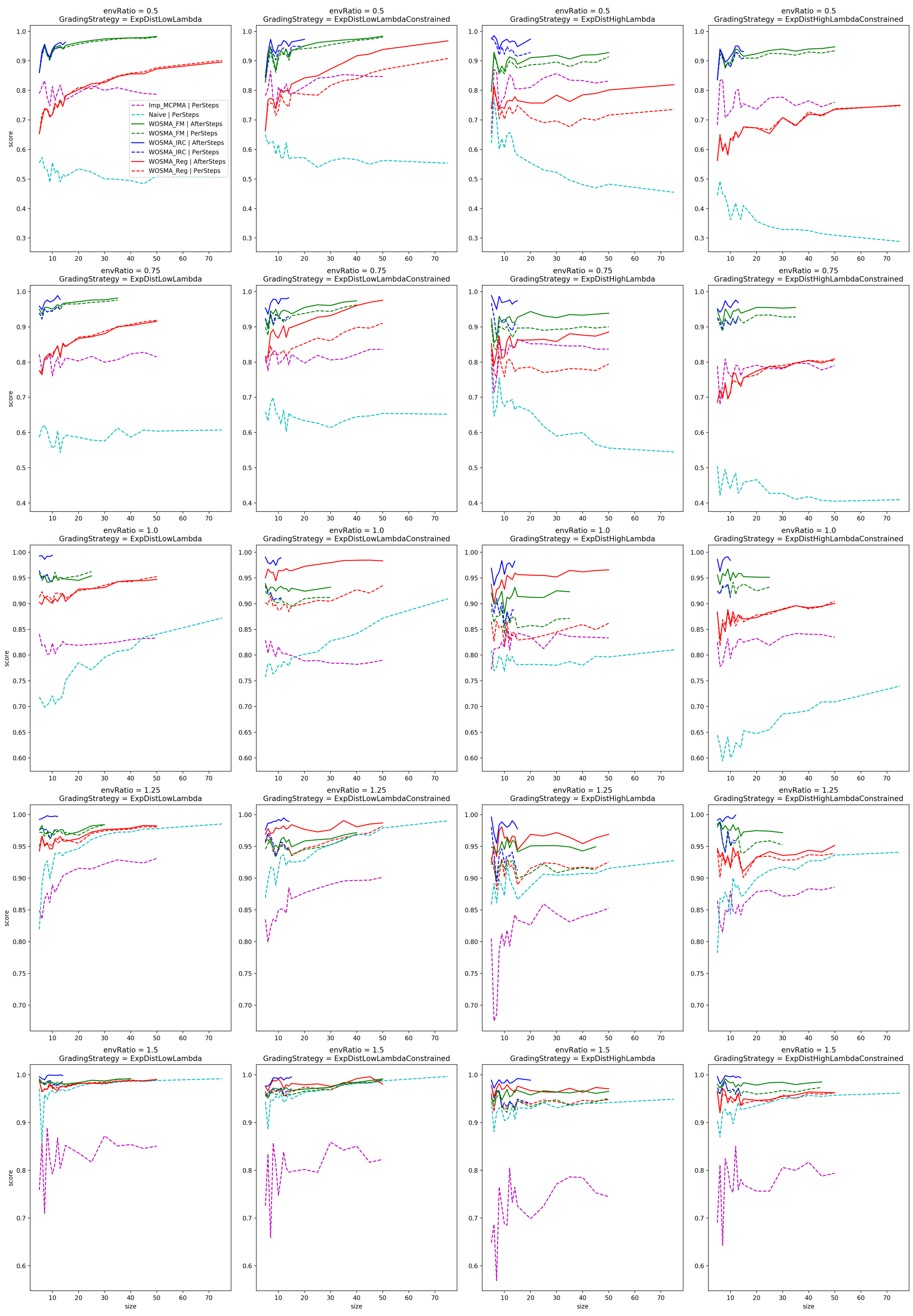


Figure 4.3: Experimental results for the fitness environments based on exponential distributions. ‘Score’ denotes not the algorithms’ absolute scores, but rather their scores relative to MCPMA. For clarity’s sake, sizes above 75 have been excluded, since often Naive was the only algorithm to make it that far within the time available.

5 Reflection

In this chapter we summarize our theoretical and experimental contributions, reflect on the design choices we have made throughout this work, and address what limitations these choices pose to our research's practical applicability. We describe avenues for future research to surpass these limitations, in Chapter 6.

5.1. Summary of Theoretical Contributions

In this work, we have made a number of theoretical contributions to the field.

We first introduced the Social Housing Market problem, which bears similarities with many known matching problems—most notably the House Allocation with Indifference problem—but differs from them in several ways (as described in Section 2.4). In Section 3.5 we proved that the two problems are meaningfully different, such that algorithms from the House Allocation with Indifference problem may be trivially adapted to the Social Housing Market problem, but not vice versa. Given that in Section 2.3 we found no literature for our exact problem, we conclude that our research provides a new and more realistic perspective on the practical problem of how to model and improve a real-life social housing market.

We have next shown how to adapt WOSMA to the unconstrained problem variant and offered a weak upper bound on algorithms' performance through MCPMA. We then delved into the class of WOSMA-like mechanisms and showed that a number of intuitive WOSMA-like mechanisms fail to perform optimally even on the constrained problem variant. For the sake of experimental research, we nevertheless implemented and tested one WOSMA-like mechanism, namely WOSMA FindMax. Finally, having been unable to find a locally optimal algorithm for our problem in the literature, we designed the IR-Cycles algorithm, which returns the locally optimal individually rational solution for any instance of the Social Housing Market Problem.

Furthermore, for the constrained problem variant, we have adapted MCPMA into Improvement-MCPMA and shown that this algorithm is locally optimal; we moreover introduced for this problem variant the simple Naive strategy, which implements the Random Serial Dictatorship mechanism with existing tenants.

From our experimental analysis of these algorithms, we found that MCPMA is a surprisingly viable algorithm to calculate upper bounds with, since in a wide variety of circumstances, it is very close to the actual optimum set by IR-Cycles AfterSteps. This suggests that it may well serve as an efficient metric to compare future algorithms with.

Finally, we have found that the higher the ratio of households to houses is, the better the locally optimal Improvement-MCPMA performs; as this ratio decreases, Naive takes the lead—suggesting that in this latter category of problem instances, a locally optimal mechanism does not lead to a globally optimal one. This opens up the question of what algorithm might perform globally optimally in these problem instances, and what performance increase it might give compared to Naive and Improvement-MCPMA.

5.2. Practical Limitations

There are a number of practical limitations to our data, model, and algorithms. In this section we describe them in detail.

5.2.1. Housing corporations have limited ability to influence the market

Housing corporations rarely have the ability to completely choose which houses are allocated to which families. Instead there often exists some standard system—in the case of Amsterdam, the WoningNet system—through which house leases are mediated; some of these houses may be owned by some housing corporation, which, under Dutch governance, may be free to offer a small percentage (e.g. $5 - 25\%$) of their houses directly to households. Such a household, meanwhile, is then free to either accept the offered house if they think they cannot do better within reasonable time on the 'regular' system, or reject the house if they would rather try their luck on the market. Thus an important question for housing corporations is: Given a housing market, which houses should we offer to which families? We do not currently investigate this question.

5.2.2. Dynamic markets are more complicated in real life than modelled here

Our version of a dynamic market is quite rudimentary: at every timestep, one house and one household enter the market separately. Of course, in real life, the market's changes over time are much more complex. Furthermore, we made several unrealistic assumptions in our model. For example: (i) when a house is left behind by some family, it might not immediately be available for a new family, since repairs and check-ups might be needed; (ii) keeping families or houses unmatched for a while is implicitly modelled as having zero cost, but is something we would like to minimize in practice; (iii) families that moved very recently might not want to move right away, even to a somewhat better house; and (iv) families are modelled as being happy to move to any house that they prefer even quite weakly (though strictly), whereas in real life, households may desire more than a marginal improvement before they are willing to move.

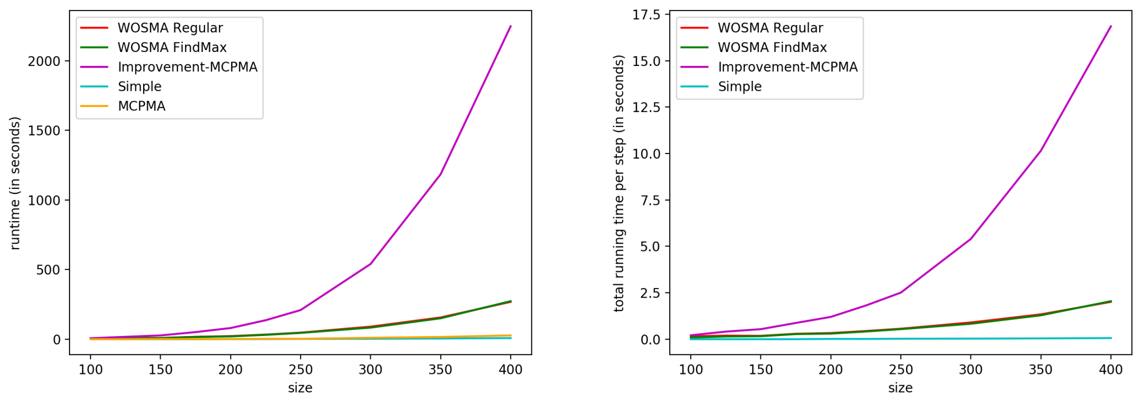
A final limitation of our dynamic model is that the AfterSteps variant presupposes perfect information about the future; our algorithms contain no measures to deal with imperfect information in a graceful manner.

5.2.3. The generated data and matching evaluation methods lack detail

The semi-realistic data which drives the MatchingAVG and MatchingMIN data-environments was generated from a small and limited real data set in a quite crude manner. Furthermore, the methods by which we evaluate individual fits and matchings as a whole, are very simplified, missing many factors that are important in real life. For instance, different families have different preferences over different locations, owing to their personalities and to the neighborhoods' cultural characters, and people that are high up on WoningNet's waiting list are unlikely to accept any house we could offer them. Finally, there are many factors that influence people's decision-making around houses that are hard to describe in precise terms, such as their taste, their precise desires, their personal ideas of what different locations are like, and so forth. These are not present in our simplified model.

5.2.4. Our algorithms' running times are impractical

An algorithm's running time is always an important practical consideration; we chose to let algorithms run for 60 seconds in our experiments, but in reality, an algorithm like these may be given up to several days to run if answers for weekly decision-making are needed (though a faster running time is desirable for experimentation and testing of policy ideas). To shed light on our algorithms' practical running times, We have thus compared the them (besides IR-Cycles, which, having exponential running time, is not usable in practice at any rate) on dynamic matchings of several sizes, using the practical settings of $envRatio = 0.5$ and the fitness environment MatchingMIN. We have graphed the results of this comparison in Fig. 5.1a. In a practical situation without future knowledge, only the PerStep variant is relevant; we thus also supply the running time per timestep as a function of size in Fig. 5.1b. It is clear that Improvement-MCPMA is by far the slowest algorithm here. Improvement-MCPMA's running time is dominated by the MCPMA algorithm, which has a cubic running time [14]. Performing a cubic fit on this runtime data, we get that Improvement-MCPMA's single-step running time as a function of size may be represented by the following function: $f(x) = -1.33045 + 0.03202x - 0.000228206x^2 +$



(a) Total running times.

(b) Running times per timestep. (MCPMA has no PerStep variant, and is thus excluded.)

Figure 5.1: Experimental running times for our five quickest algorithms. Beyond a size of 400, some of these algorithms ran into memory issues on our computer.

$6.55315x^3 \cdot 10^{-7}$. A housing corporation such as Ymere might own around 50,000-100,000 houses, which puts the practical running time of Improvement-MCPMA between 2.58 and 20.7 years. This is, unfortunately, impractical.

5.3. Practical Implications

The limitations discussed above imply that our work's direct value is primarily theoretical, as summarised in Section 5.1. But our results nevertheless suggest **five** practical implications.

Firstly, in a realistic scenario, the performance gaps between the PerStep and AfterSteps versions of all applicable algorithms are in general quite small and often even negligible. Since this is the case across a large number of fitness environments, and there is no pattern of the difference increasing with graph size, it suggests that there might be little benefit to developing algorithms that take future knowledge into account.

Secondly, the small differences in results between constrained and unconstrained fit environments, which we highlighted in answer A.7, suggest that a less granular model of families' and houses' traits may still allow for the model to be useful for drawing practical policy-related conclusions from. Thus our model may provide housing corporations with a simple way with which to test what effects various policy changes might have.

Thirdly, an inability to perform cycles and instantaneous chains may be not too harsh a constraint on the effectiveness of our methods; as noted in answer A.2, Improvement-MCPMA performs 0.89 times as well as WOSMA FindMax in settings where there are twice as many households as houses. This suggests that there may not be much benefit to pursuing cycles in practice. Of course, as noted in Section 5.2.2, we model families that move in one month to be fully willing to immediately move again in the next month, so long as their new house is strictly a better fit than their former house; this unrealistic design decision undoubtedly has some influence on the performance of both Improvement-MCPMA and WOSMA FindMax. However, there is little reason to think this affects either of these algorithms disproportionately compared to the other. Moreover, in constrained fitness environments (and especially in the MatchingMIN scenario, which uses binary fit-scores), households are guaranteed not to move very often, since any household would get to a perfectly-fitting house after 3 (resp. 1) move at most. Here, as well, we see a similarly small improvement gap between Improvement-MCPMA and WOSMA FindMax. Thus we expect our conclusion, namely that there is not much benefit to pursuing cycles and chains in practice, to hold up, even if it were taken into account that families are, in general, not willing to move very frequently.

Fourthly, the improvements which WOSMA FindMax and Improvement-MCPMA were able to get, in a realistic scenario, over WOSMA Regular and Naive, respectively, suggest that perhaps, compared to the current baseline, housing corporations could create healthier social housing markets if they were to use more sophisticated versions of the strategies described in this research. In particular, Improvement-MCPMA's greater performance compared to Naive in practical settings, suggests that in realistic markets, a lot of value can be gained by algorithmically locating global[1] optima as opposed to simply letting the locally best moves take place.

Finally, on top of the above four insights, we note that as our work newly introduces and explores the Social Housing Market Problem, much of its value lies in providing a fruitful starting point for future research to use and develop further. To this end, we describe in our final chapter a large variety of improvements that may be made to our model in order to make it more usable and useful in practice.

[1]'Global' referring here to optima that (as per our first insight) are local in time, but global in the sense that they take the entire matching at its current point in time, into account.

6 Conclusion

In this work we have introduced, defined, examined, and contextualised the Social Housing Market problem. We have moreover adapted and created several algorithms for this problem, and have compared them on realistic and mathematical data simulating a wide variety of settings.

Our theoretical contributions to the field are as follows: (i) We have introduced the new Social Housing Market problem, which fits the realistic social housing market better than existing problems (such as the Housing Allocation with Indifference problem), and have proven that algorithms which perform well for existing problems are not guaranteed to do as well on ours; (ii) we have adapted WOSMA and MCPMA to the unconstrained version of our problem; (iii) we have designed the WOSMA FindMax variant as well as the IR-Cycles algorithm, which uniquely finds an optimal solution, for the unconstrained version of our problem; (iv) we have adapted MCPMA into Improvement-MCPMA, and the Random Serial Dictatorship mechanism into Naive, for the constrained version of our problem; (v) we have found that the efficient MCPMA provides a fairly realistic upper bound on algorithms' performance; and finally, (vi) we have found that a locally optimal algorithm such as Improvement-MCPMA may, depending on the setting used, underperform relative to the greedy Naive algorithm, opening up the question of whether there exists a constrained algorithm that performs meaningfully better than both.

Our practical contributions, meanwhile, include the following: (i) the slight performance differences between algorithms' PerStep and AfterSteps variants suggest that there may be limited use to taking future information into account when taking decisions; (ii) the negligible differences between constrained and unconstrained fit environments suggest that relatively coarse data may still allow the model to inform policy-making; (iii) the relatively small differences between Improvement-MCPMA and WOSMA FindMax in housing markets where houses are in particularly short supply, imply that there may not be a huge benefit to pursuing cycles and chains; and (iv) the large difference between the 'baseline' Naive strategy and the more sophisticated Improvement-MCPMA in the constrained problem variant, suggests that even disregarding chains and cycles, it may be worth looking into developing and using sophisticated algorithms in practice.

Finally, since our work introduces its central problem, it serves as a starting point for future research to develop further.

6.1. Future Research

As described in Chapter 5, there exist a number of limitations to this research which, if lifted, may allow it to be used as a worthwhile starting point for designing a practically usable model and algorithm. To this end, we here describe steps for future research to solve these limitations.

6.1.1. Housing corporations' limited agency may be taken into account

To address the fact that housing corporations have only a limited ability to influence the market, as described in Section 5.2.1, it might be prudent to incorporate several different methods of selecting viable house-family pairs from a market. On a static basis, these might take the form of simple heuristics such as 'Which move brings the highest benefit to the family in question?', or 'Which cycle has the highest least improvement among its links?'. In a dynamic market, however, where the amount of houses that may be directly mediated represents a kind of ‘budget’ that gets expended over a set period of time, we might require more sophisticated strategies which, at any timestep, take into account the difference between drawing from the budget now versus doing so later. A comparison between these different heuristics may suggest a more exact course of action for social housing corporations than we have been able to provide—and a comparison between the outcomes that may be gained through limited influence vs. when housing corporations have much more agency, may decide an interesting question: namely, whether giving housing corporations more power might lead to better housing markets as a whole. The improvements which WOSMA FindMax and Improvement-MCPMA were able to get, in a realistic scenario, over WOSMA Regular and Naive, respectively, suggest that this might be the case.

Incorporating and comparing static heuristics of which house-household pairs are picked and modelled as taking place in every timestep, would be a fairly simple modification to our current algorithms—a mere filtering step that takes as input the per-step matching changes that these algorithms propose, and returns some subset of them according to the chosen heuristic. The well-considered development of more complex dynamic strategies which trade off uncertain future advantages against certain current benefits, however, would likely be a more involved task.

6.1.2. More complex considerations may be incorporated into our dynamic model

We discussed the simplicity of our dynamic model in Section 5.2.2. A number of changes may be made to make our model more accurate to reality, such as: (i) Multiple houses and households may enter—and leave!—the social housing market at every timestep; (ii) there may be seasonal fluctuations in this process which may be taken into consideration beforehand; (iii) families change over time in size, need for accessibility options, and income; and so forth.

The first of these changes would be trivial to implement; the second and third would require a lot more real-world data analysis.

We moreover brought up a number of unrealistic assumptions that were encoded in our model, such as the notion that families who have moved in one month, might not be willing to move again right in the next month. Some of these assumptions might trivially be encoded within the moves and cycles that our algorithms deem valid, though as the number of such constraints increase, it may become harder for simpler algorithms to remain locally optimal. Pareto optimality should in general be a more easily verifiable quality, however, as discussed in Section 3.2.

Finally, one might create strategies that take uncertainty or imperfect information into account. For example, Abizada and Chen [1] discuss methods for designing ‘robust’ strategies that are capable of dealing with unexpected changes in the availability of houses, and Aziz et al. [5] propose several models of uncertainty in the House Allocation problem which would be interesting to integrate into future research, though doing so might take quite some work.

6.1.3. The data generation and matching evaluation methods might be improved

As noted in Section 5.2.3, the parts of our research which are based on real data, are fairly simplistic in nature. A more detailed analysis of the data that is available might help create a better picture of different real-world markets, and thereby allow for more accurate predictions. Moreover, future research might incorporate the more subtle factors we named in Section 5.2.3, which drive people’s behaviour regarding which houses they would be willing to move to. Fortunately, housing corporations that work together with WoningNet (or similar systems in different cities and countries) may have access to data showing which houses which kinds of people apply to. An approach which incorporates machine-learning to understand whether a house and a family are a good fit for each other, seems like it might be a valuable addition to this research; however, it would be somewhat of an undertaking. Fortunately, there exist many smaller steps to make the data more realistic, and on-hands experience in the field would surely be useful for creating a better idea of which neighbourhoods are desired by which types of people.

6.1.4. Different models may be tested and compared with ours

Throughout this work, we stuck to a model in which the salient factor is a single numerical fit value between each house and each family, but there are a number of different ways we might have modelled the housing market instead.

For instance, we might have incorporated the houses’ values (as calculated in the Netherlands through their WOZ-valuations) to gain a more objective sense of how valuable different houses are, rather than an exclusively subjective one. It might also be valuable, in the future, to incorporate a model of the default WoningNet waiting-list system[1], in which families are dynamically added to (and removed from) some central waiting list, and are modelled as having probabilities for trying for new houses, as well as for how likely they are to be accepted for said houses, that are based on their position in the main queue. Then, at each timestep in a dynamic matching, this process might provide a 'default' behaviour for the market, such that strategies for direct mediation would have to ensure they offer better houses to households than these households are themselves likely to obtain within some period of time. The addition of such a model might require a thorough analysis of WoningNet's data, but would help clarify the added benefits—or lack thereof—that different strategies of direct mediation might have, compared to a default of performing no direct mediation.

6.1.5. Different goals may be pursued

Throughout this work, we have pursued a single goal in our matching optimizations: to maximize the average fit across all families. But there are many other possible (primary and secondary) goals that might merit their own research and strategies.

Examples include: (i) maximizing the matching's average fit for each person rather than for each family; (ii) minimizing inequality in fit (especially between e.g. high-income households and low-income families); (iii) ensuring no neighbourhood has too high a percentage of families that are elderly, have special needs, or fit within some general cultural or personality category[2]; (iv) maximizing the amount of chains we can carry out; (v) maximizing the likelihood that a given family, when offered the house we have picked for them, will accept it; (vi) ensuring that families not merely meet houses' financial demands but are furthermore a good fit for them (i.e. minimizing the allocation of cheap houses to relatively rich families); (vii) and finally, there are several special groups that housing corporations might like to help out specifically, such as elderly people who just barely fail to be given priority as per the government's rules, but who could still use it, people with special needs, or teachers in neighbourhoods which lack them; and so forth.

Our current algorithms may trivially be evaluated and compared on how they perform according to any of these goals, but are not optimized for them and provide no guarantees relative to these objective functions. For each of these goals, new algorithms may be designed, or, where goals are taken as secondary, our existing algorithms may perhaps be adapted to find a better balance between some primary and secondary goals.

6.1.6. The algorithms may be made quicker

As noted in Section 5.2.4, our algorithms—Improvement-MCPMA in particular—are too slow to be run on problem instances of realistic sizes. (For that matter, the WOSMA variants also require too much memory, though Improvement-MCPMA does not.) A 'simple' solution would be to code the algorithms to be more efficient and to improve the model's and algorithms' memory usage, but there are other solutions as well. Taking into account housing corporations' limited agency as noted in Section 5.2.1, if a housing corporation were to run Improvement-MCPMA only on, say, the 10% of families that need their help the most, the algorithm would become tractable—particularly since Improvement-MCPMA, at every timestep, considers not at all houses, but rather only empty houses, of which there are a dramatically smaller amount; thus such a reduction in the amount of families would result in a substantial overall performance increase. With these settings, the algorithm would finish within a day or two, making the algorithm usable in practice. An alternative approach might be to utilize some kind of 'split, solve, and merge'-mechanism that first splits the problem instance into multiple smaller ones, next runs the algorithms on these smaller problem variants, and finally merges the solutions in some way so as to make the final result better than the sum of its parts. It would be an interesting research question to see how various such mechanisms compare, and how close they could come to the optimal solution. Moreover, as noted in answer A.12, our algorithms' performances do not meaningfully increase beyond a certain small size; thus it might be the case that even a simple composite of solutions to smaller sub-problems, which does not modify the smaller solutions at all but merely integrates them as they are given, might perform competitively. This latter approach would be trivial to implement.

[1]Our Naive algorithm, as explained in Section 4.3.1, may be viewed as a very simple model of a waiting list, but it fails to consider many of the effects such a waiting list has on families' behaviour.

[2]See e.g. van Ham and Manley [20] who focus on neighbourhoods' ethnic mix in England.

6.1.7. Miscellaneous improvements

It might be valuable to implement and compare 'Algorithm 1' from Plaxton [15], which, as noted in Section 2.5, is Pareto-optimal, efficient—in fact the authors believe it might have optimal time complexity—and strategy-proof. As proven in Section 3.5, this algorithm is not guaranteed to return the best achievable outcome, but it would nonetheless be interesting to see how it compares to the algorithms tested in this paper.

Finally: IR-Cycles' outcome is guaranteed to be locally optimal, but as we noted in Section 3.2, there must exist some WOSMA-like mechanism that reaches this same outcome through a dramatically smaller action-space; said mechanism would likely be much quicker than IR-Cycles as a result. It yet remains to be found.

Appendix

A
TTC Is Not Pareto-Efficient When Indifference Is Present

Here we provide a brief example, copied from Jaramillo and Manjunath [11], to show that the Top Trading Cycles (TTC) algorithm is no longer guaranteed to be Pareto-efficient when indifference in families' preferences is allowed.

Let f_1, f_2, f_3 be matched with h_1, h_2, h_3, respectively, and let their ordinal preferences be as follows:

f_1	h_2, h_3	h_1	
f_2	h_1	h_3	h_2
f_3	h_1	h_2	h_3

Then breaking f_1's preference tie in favour of h_2 would lead to the outcome $\{(f_1, h_2), (f_2, h_1), (f_3, h_3)\}$, and breaking their tie otherwise would lead to $\{(f_1, h_3), (f_2, h_2), (f_3, h_1)\}$. Neither of these outcomes is Pareto-efficient.

B

Greedy-3.5

We here propose an alternative version of the Greedy-3 mechanism which we defined in Section 3.3.3, namely Greedy-3.5. It modifies Greedy-3 in a manner which the reader might wish to see analyzed; hence we include it here and prove its suboptimality for completeness' sake.

The intuition for this modified mechanism is as follows. For a given house h_x, Greedy-3 first checks which families would most benefit from h_x (i.e. F'_x), and only then searches within this set for those families who would most suffer from not getting it (i.e. F''_x. Greedy-3.5 does the opposite: It first checks, given some house h_x, which families would most suffer from not getting it, and then amongst those chooses some family that would most benefit from getting h_x.

The mathematical formulation of Greedy-3.5 is as follows. We first identify the set of initially empty houses $H^\emptyset$ in the input matching M^0 with houses H and families F. (Here we have that $H^\emptyset \subseteq H$.) Now, so long as there exists some empty house in $H^\emptyset$ (i.e. not in $H \setminus H^\emptyset$) that any family $f \in F$ prefers to their current house, we pick some empty house $h_x \in H^\emptyset$ in the following way. First for each empty house $h_i \in H^\emptyset$ we define F''_i in the following way:

$$\begin{aligned}
F''_x = \{&f_i \in F \\
&\mid \forall h_y \neq h_x \in H^\emptyset \setminus \bar{H}^\emptyset : \\
&(fit(h_x, f_i) - fit(h_y, f_i)) \geq 0 \\
&\wedge (fit(h_x, f_i) - fit(h_y, f_i)) \geq \max_{f_j \in F \setminus \{f_i\}} (fit(h_x, f_j) - fit(h_y, f_j))\}
\end{aligned}$$

Next, we define $F'_i \subseteq F''_i$ to be the subset of families in F''_i that would benefit from h_i at least as much as they would from any other house.

Greedy-3.5 then picks some house h_x from those houses for which F'_i is smallest, and here prioritizes (as Greedy-2 and Greedy-3 did) houses that give the biggest benefit to some family in F'_i. Having picked such a house h_x, Greedy-3.5 assigns it to some random family in F'_x.

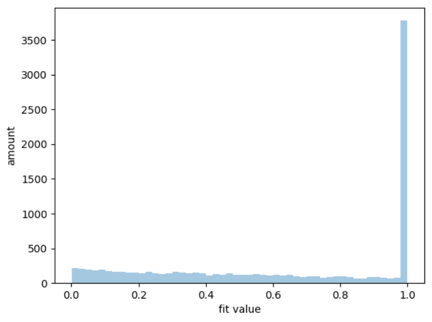
Lemma B.0.1. *Greedy-3.5 is not guaranteed to return the best achievable solution.*

Proof. Our counter-example to Greedy-3 serves us here as well. We get $F''_1 = F'_1 = \{f_1, f_2\}, F''_2 = F'_2 = f_3, F''_3 = \emptyset$. Thus h_2 is allocated first, namely to f_3. Next, we still have $F''_1 = F'_1 = \{f_1, f_2\}, F''_3 = F'_3 = \emptyset$. Greedy-3.5 thus chooses to allocate h_1, and assigns it randomly to either f_1 or f_2. Suppose it chooses f_1. Then we have the same sub-optimal outcome that Greedy-3 was able to pick. Hence Greedy-3.5 is not optimal either. $\square$

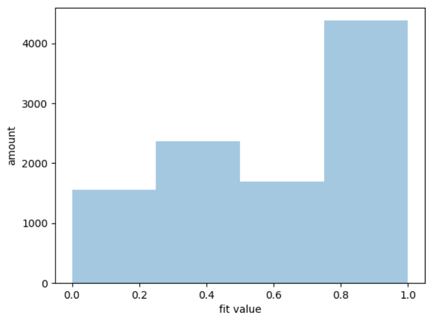
C

Sample Exponential Distributions

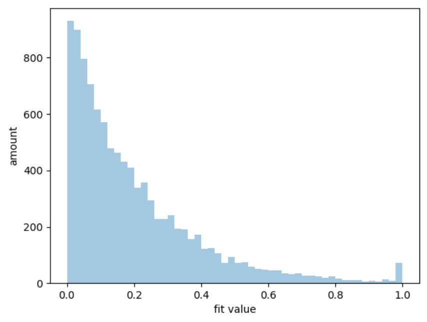
Here we briefly provide Fig. C.1, which shows sample distributions for the four exponential distribution-based grading strategies: ExpDistLowLambda, ExpDistLowLambdaConstrained, ExpDistHighLambda, and ExpDistHighLambdaConstrained.



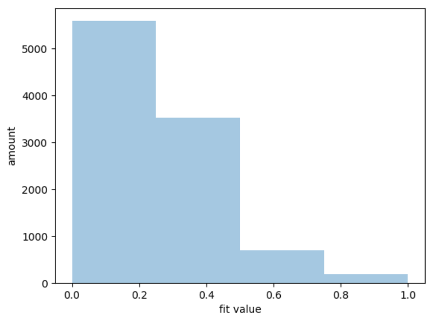
(a) Exponential distribution where $\lambda = 1$ with 50 bins, representing the ExpDistLowLambda grading strategy.



(b) Exponential distribution where $\lambda = 1$ with 4 bins, representing the ExpDistLowLambdaConstrained grading strategy. Possible values are: $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$.



(c) Exponential distribution where $\lambda = 5$ with 50 bins, representing the ExpDistHighLambda grading strategy.



(d) Exponential distribution where $\lambda = 5$ with 4 bins, representing the ExpDistHighLambdaConstrained grading strategy. Possible values are: $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$.

Figure C.1: Exponential distributions for $\lambda \in \{1, 5\}$ with $10,000$ samples. Samples outside of the range $[0, 1]$ were set equal to whichever limit they exceeded. These represent the four exponentially distributed grading strategies we used.

Bibliography

[1] Azar Abizada and Siwei Chen. House allocation when availability of houses may change unexpectedly. *Mathematical Social Sciences*, 81:29 – 37, 2016. ISSN 0165-4896. doi: https://doi.org/10.1016/j.mathsocsci.2016.03.002. URL http://www.sciencedirect.com/science/article/pii/S0165489616000226.

[2] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. *CoRR*, abs/1402.3643, 2014. URL http://arxiv.org/abs/1402.3643.

[3] Andrei Asinowski, Balázs Keszegh, and Tillmann Miltzow. Counting houses of pareto optimal matchings in the house allocation problem. *Discrete Mathematics*, 339(12):2919 – 2932, 2016. ISSN 0012-365X. doi: https://doi.org/10.1016/j.disc.2016.05.027. URL http://www.sciencedirect.com/science/article/pii/S0012365X16301728.

[4] Haris Aziz and Bart De Keijzer. Housing markets with indifferences: A tale of two mechanisms. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, page 1249–1255. AAAI Press, 2012.

[5] Haris Aziz, Péter Biró, Ronald de Haan, and Baharak Rastegari. Pareto optimal allocation under uncertain preferences: uncertainty models, algorithms, and complexity. *Artificial Intelligence*, 276:57–78, Nov 2019. ISSN 0004-3702. URL http://www.sciencedirect.com/science/article/pii/S0004370219301638.

[6] Aytek Erdil and Haluk Ergin. Two-sided matching with indifferences. *Journal of Economic Theory*, 171:268–292, 2017. ISSN 10957235. doi: 10.1016/j.jet.2017.07.002. URL http://dx.doi.org/10.1016/j.jet.2017.07.002.

[7] Zhi Ping Fan, Ming Yang Li, and Xiao Zhang. Satisfied two-sided matching: a method considering elation and disappointment of agents. *Soft Computing*, 22(21):7227–7241, 2018. ISSN 14337479. doi: 10.1007/s00500-017-2725-1.

[8] Harold N. Gabow. Path-based depth-first search for strong and biconnected components. *Information Processing Letters*, 74(3):107 – 114, 2000. ISSN 0020-0190. doi: https://doi.org/10.1016/S0020-0190(00)00051-X. URL http://www.sciencedirect.com/science/article/pii/S002001900000051X.

[9] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2312726.

[10] Jiarui Gan, Warut Suksompong, and Alexandros A. Voudouris. Envy-freeness in house allocation problems. *CoRR*, abs/1905.00468, 2019. URL http://arxiv.org/abs/1905.00468.

[11] Paula Jaramillo and Vikram Manjunath. The difference indifference makes in strategy-proof allocation of objects. *Journal of Economic Theory*, 147(5):1913 – 1946, 2012. ISSN 0022-0531. doi: https://doi.org/10.1016/j.jet.2012.05.017. URL http://www.sciencedirect.com/science/article/pii/S0022053112000713.

[12] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013*, pages 589–600, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-40450-4.

[13] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255 – 267, 1994. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(94)90042-6. URL http://www.sciencedirect.com/science/article/pii/0304397594900426.

[14] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., USA, 2005. ISBN 0321295358.

[15] C. Greg Plaxton. A simple family of top trading cycles mechanisms for housing markets with indifferences. In *Proceedings of the 24th International Conference on Game Theory*, 2013.

[16] Byford M. Karabay B. McNelis S. Sharam, A. and T. Burke. Matching markets in housing and housing assistance. *Australian Housing and Urban Research Institute*, 2018. doi: 10.18408/ahuri-5315301.

[17] Tayfun Sönmez and M. Utku Ünver. House allocation with existing tenants: A characterization. *Games and Economic Behavior*, 69(2):425–445, 2010. ISSN 08998256. doi: 10.1016/j.geb.2009.10.010. URL http://dx.doi.org/10.1016/j.geb.2009.10.010.

[18] Jayme L. Szwarcfiter and Peter E. Lauer. A search strategy for the elementary cycles of a directed graph. *BIT Numerical Mathematics*, 16(2):192–204, Jun 1976. ISSN 1572-9125. doi: 10.1007/BF01931370. URL https://doi.org/10.1007/BF01931370.

[19] Tayfun Sönmez, Utku Unver, Boston College, Jess Benhabib, Alberto Bisin, and Matthew Jackson. Matching, allocation, and exchange of discrete resources. *Handbook of Social Economics, Elseview, forthcoming*, pages 781–852, 09 2009.

[20] Maarten van Ham and David Manley. Social housing allocation, choice and neighbourhood ethnic mix in england. *Journal of Housing and the Built Environment*, 24(4):407, Sep 2009. ISSN 1573-7772. doi: 10.1007/s10901-009-9158-9. URL https://doi.org/10.1007/s10901-009-9158-9.