Genetic-Algorithmic Optimisation for School-Allocation Mechanisms

A Study of Amsterdam's Student to High-School Allocation Problem

MSc Thesis *(Afstudeerscriptie)*

written by

Philip W.B. Michgelsen
(born May 30th, 1990 in Enschede, The Netherlands)

under the supervision of **Prof. Dr. Jan van Eijck**, and submitted to the Board of Examiners in partial fulfillment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam.*

Date of the public defense:	**Members of the Thesis Committee:**
July 5th, 2016	Dr. A. Baltag
	Prof. Dr. J. van Eijck
	Prof. Dr. B. van der Klaauw
	Dr. C. Schaffner
	Dr J. Szymanik

Institute for Logic, Language and Computation

Foreword

Abstract

In the municipality of Amsterdam, students who graduated from primary school, are allocated to high schools through a school-allocation mechanism. In recent years, different school-allocation mechanism have been used and all have been criticised by the public. This thesis investigates the school-allocation mechanisms that have been used in Amsterdam, from a mathematical point of view, and aims to make it mathematically clear and formal what properties a school-allocation mechanism must have, according to the municipality of Amsterdam's policy makers, and what properties the used allocations mechanisms actually have. Furthermore, this thesis aims to explore if a genetic-algorithmic optimisation method can help in the design of a new school-allocation mechanism, which can outperform the school-allocation mechanisms used in Amsterdam. The two most important formal properties desired of school-allocation mechanisms by policy makers, are Pareto consistency and strategy proofness. A school-allocation mechanism is Pareto consistent if it only produces Pareto-optimal allocations. In addition, other conditions are posed by Amsterdam's policy makers, which determine, given two Pareto-optimal allocations, which is considered to be better. With these additional conditions, an objective fitness function can be given, which determines which Pareto-optimal allocation is fittest. The existence of a particular school-allocation mechanism, the Permutation Allocation Mechanism, which is able to produce all Pareto-optimal allocations for some given school market, makes it then possible to find the fittest Pareto-optimal school allocation by using a genetic-algorithmic optimisation method. This thesis shows that the so obtained genetic-algorithmic optimised school-allocation mechanism is expected to statistically outperform its not-optimised counterpart.

Acknowledgments

First of all, I would like to thank my supervisor Jan van Eijck: thank you for introducing me to the fascinating topics of functional programming, genetic algorithms and matching theory. Without the classes and research projects taught by you, it would not have been possible for me to write this thesis. Also, I would like to thank you very much for helping me gain the computational skills I always aspired. I would also like to thank Christian Schaffner, whose courses I enjoyed so very much and who helped me gain the by me much desired mathematical maturity. Moreover, I am very grateful to Göran Sundholm, my professor from Leiden, for introducing me to field of Logic, which became my great passion. For this particular project, I would like to thank Werner Toonk, from the Amsterdam city council, and Pieter Reimer, from the municipality of Amsterdam, who provided me with the background knowledge of the Amsterdam school-allocation problem and with the data needed to run the simulations. In addition, I would like to thank all the other members of the thesis committee: thank you very much that you are willing to make the time to read and assess my 'afstudeerscriptie'. To all other MoL students, I would like to say a huge thanks for always helping me out and making the time to help me acquire the background knowledge I was lacking upon starting this master's program. In particular, I would like to thank Tim Coopmans, Mathijs Mul, Anna Oosting and Olim Tuyt. Finally, I would like to thank my parents for always supporting me and my greatest love Marit, for always standing by me.

Contents

1 Introduction **1**

2 Background **2**
2.1 Dutch School System . . . 2
2.2 Amsterdam High Schools . . . 4
2.3 School-Allocation Mechanisms in Amsterdam . . . 6
2.4 Deferred Acceptance in Amsterdam . . . 8
2.5 Looking Ahead . . . 10

3 School Allocations **11**
3.1 School Allocations . . . 11
3.2 Student Preferences . . . 12
3.3 School Markets . . . 14
3.4 Priorities . . . 14
3.5 Properties of Allocations . . . 15

4 School-Allocation Mechanisms **19**
4.1 Mechanisms . . . 19
4.1.1 Permutation Allocation Mechanism . . . 20
4.1.2 Boston Allocation Mechanism . . . 23
4.1.3 Deferred Acceptance School-Allocation Mechanisms . . . 27
4.2 Strategic Questions . . . 32
4.2.1 Strategy Proofness . . . 32
4.2.2 Strategic Investigations of School-Allocation Mechanisms . . . 34
4.3 Simulations with Mechanisms . . . 36
4.4 Mechanisms for Amsterdam . . . 38

5 Genetic Optimisation **40**
5.1 Optimisation . . . 40
5.2 Genetic Algorithms . . . 40
5.3 Reproduction . . . 43
5.4 Genetic Optimisation of School Allocations . . . 45
5.4.1 Organisms and Chromosomes . . . 45
5.4.2 Fitness Function and Reproduction . . . 46
5.4.3 Chromosome Manipulation: Crossover and Mutation . . . 47
5.4.4 GOPAM . . . 49
5.5 Simulations with the GOPAM . . . 51
5.6 Genetic-Algorithmic Optimisation for Allocation Mechanisms . . . 52

6 Bibliography **54**

7 Appendix: Computational Implementations **57**
7.1 Haskell . . . 57
7.1.1 Imported Modules and Introduced Types . . . 57
7.1.2 General Help Functions . . . 57
7.2 School Allocation Mechanisms . . . 59
7.2.1 Permutation Allocation Mechanism . . . 59
7.2.2 Boston Allocation Mechanism . . . 60
7.2.3 Deferred Acceptance . . . 61
7.3 GOPAM . . . 62
7.3.1 Help Functions . . . 62
7.3.2 The GOPAM . . . 63
7.3.3 Fitness Functions . . . 67

Chapter 1

Introduction

In recent years, in the municipality of Amsterdam, students who graduated from primary schools, were allocated to high schools through a school-allocation mechanism. The mechanism used for this allocated students to high schools based on preference lists handed in by students using a deferred acceptance school-allocation mechanism. The first deferred acceptance mechanism used was severely criticized, because it resulted in Pareto-inefficient school allocations. An allocation A is Pareto inefficient, if there exists some allocation A' in which some students are assigned a school which they prefer over the school assigned to them in A, and no student is assigned a school in A', which the student likes less than the school assigned in A. In particular, it meant that the allocation, produced by the school-allocation mechanism in Amsterdam, gave students rational trade options: in the final allocation some students could, by switching places, both be better off, based on their handed-in preference list. Such school swaps were not allowed, since they would render the allocation mechanism *not* strategy proof. In a strategy-proof allocation mechanism, the dominant strategy for students is to reveal their true school preferences. Strategy proofness was an important motivation of the representative body of Amsterdam's high schools and the municipality of Amsterdam, for choosing the Deferred Acceptance school-allocation algorithm.

In this theses different known school-allocation mechanisms will be investigated from a mathematical point of view. The Amsterdam school-allocation problem of allocating primary school students to high schools according to handed-in preference lists, will be taken as the point of departure. All school-allocation mechanisms investigated in this thesis are mechanisms that closely resemble mechanisms that have been used in Amsterdam to allocate students to high schools. The goal of this thesis is twofold. First, this thesis aims to make it mathematically clear and formal what properties the Amsterdam policy makers wish a school-allocation mechanism to have and what specific properties allocations produced by the different known mechanism actually have. Second, this thesis aims to explore if a genetic-algorithmic optimisation method can help in the design of a new school-allocation mechanism, which can outperform the known school-allocation mechanisms. It is the hope of the author that (i) this thesis enables policy makers to choose a school-allocation mechanism best suited for the specific problems of Amsterdam, and that (ii) this thesis shines light on a new innovative approach to school-allocation mechanisms by using genetic algorithms to optimise known school-allocation mechanisms.

The structure of this thesis will be as follows: first, the Dutch school system and specifics of the Amsterdam school-allocation problem will be introduced; second, the formal understanding of school-allocation problems will be set out; third, the school-allocation mechanisms that have been used in Amsterdam will be studied and tested; and fourth, the new genetic approach to school-allocation mechanisms will be explored and tested. All code that is used for testing the school-allocation mechanisms discussed in this thesis and in the design of the new genetic optimisation method, is attached to the end of this thesis.

Chapter 2

Background

2.1 Dutch School System

The Netherlands educational system is divided into three 'horizontal' levels: primary, secondary and tertiary education. The three horizontal education levels consist of different 'tiers', which range from a more applied to a more scientific orientation. The primary school level consists of only one tier, which is called 'basisschool'. The secondary school level consists of three tiers VMBO, HAVO and VWO, where both the VMBO and VWO tiers are further subdivided into different programs (again ranging from a more applied to a more scientific focus). Furthermore, VMBO students have the possibility to follow their education with extra aid called LWOO (NL: leerwegondersteuenend onderwijs).[1] The tertiary education level consists of vocational education (MBO) and higher education (HBO and University). All schools in the secondary education level will be called 'high schools' in the following. Figure 2.1 below gives a schematic overview of the Dutch school system.

In the Netherlands there is compulsory education. The *Compulsory Education Act of 1969*, requires students to be enrolled in a registered school from the age of 5 for at least 12 *school* years or until the end of the school year in which the student turns 16. After this period, students who do not have obtained a diploma for HAVO, VWO or MBO two and higher, will have to follow education until they have done so, or until they turn 18. This means that in The Netherlands there is *de facto* compulsory education until the age of 18.[2] Since primary school normally takes eight years and students usually are five years old in the first year, all students who attend 'basisschool' will subsequently attend secondary education.

Every student, upon leaving primary school, receives an official school recommendation (NL: schooladvies), composed by the teaching staff, which dictates for which secondary education the student may apply.[3] Students then apply themselves for some secondary school and educational program, that is in line with the school recommendation. Students are allowed to apply for more *applied* programs than advised, but may not apply to more more scientific programs than recommended in the official recommendation. Secondary schools are obliged, barring some rare exceptions, to accept the applying students, if the application is in accordance with the school recommendation.[4] Since some schools are very popular and school capacities are limited, schools have to fairly decide which students to enrol and which not. To help this decision process *school-allocation mechanisms* have been introduced, either by schools individually (e.g. lotteries) or by schools collectively (e.g. matching mechanisms).

[1] For students with special educational needs for which the before mentioned secondary education tiers (also with extra aid) are too demanding, there exists a special form of secondary education: *praktijkonderwijs*. In the following, due to the very limited size and special nature of this form of education, we will not regard *praktijkonderwijs* as secondary education.

[2] Article 3 and 4a of the Compulsory Education Act of 1969 (NL: Leerplichtwet 1969).

[3] There is no legal bound on the number of tiers that may be advised by the teaching staff to some student, although the PO-raad and VO-raad (the representative bodies of respectively primary and secondary schools) claim that no more than two tiers or programs can be advised.

[4] Article 2 sub 1, in conjunction with, article 3 sub 2, of the Organisational Decision on the Law on Secondary Education (NL: Inrichtingsbesluit Wet op het Voortgezet Onderwijs).

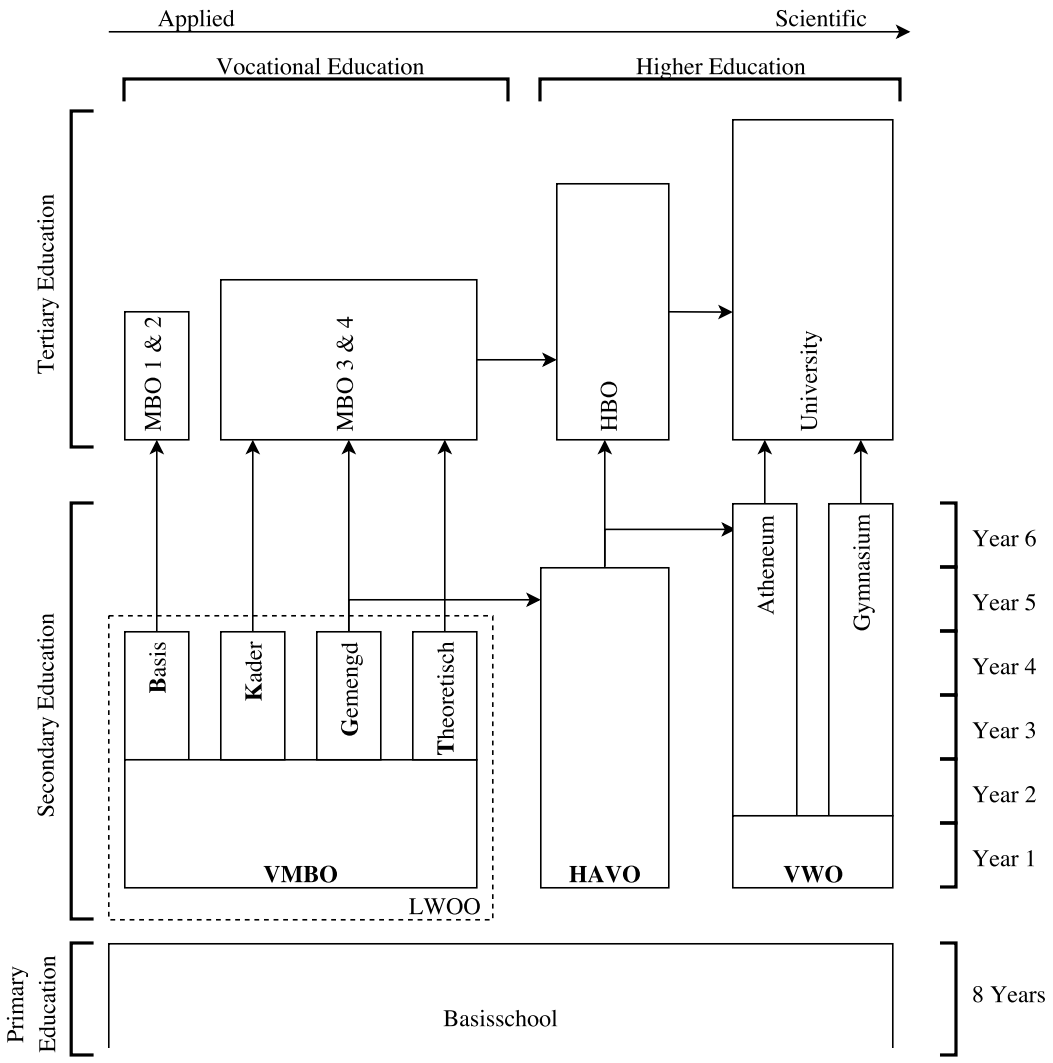

Figure 2.1: Schematic Representation of the Dutch School System

In the city of Amsterdam, the difference in popularity between different high schools was very large. This resulted in a particular need for individual lotteries or a central school matching mechanism. Such decision processes, although of great impact on students, have no influence on the future study opportunities of the student, since the differences in quality of education between different high schools, for a particular tier, are very small, and graduation in a specific tier determines which path of studies a student can follow, not at which school that tier was followed. The arrows from secondary education programs to tertiary education programs, in the Dutch school-system diagram (Figure 2.1) indicate what is the most scientific tertiary education program that students are allowed to attend to, after finishing secondary education. Degrees of some secondary and tertiary education programs allow a continuation of studies in some other secondary or tertiary education program (where such a shift of tier is allowed is again indicated with arrows). The Dutch school system diagram illustrates that it is very important for Dutch high-school students to attend the 'right' secondary education *tier* in light of the possible tertiary education tiers that can be attended. Attending the 'right' secondary education tier is however completely up to the future high-school students themselves, since there is no shortage of seats in Amsterdam, nor in the Netherlands, to place all high-school students in some school, within acceptable range from their home, offering the secondary education tier the student wants to attend. There are, however, some high schools to which more students apply then it can offer seats.

This problem is known (in Amsterdam) as *the capacity problem*. It is important to stress again, that this capacity problem by no means implies that some students will not be able to follow secondary education in the educational tier of their liking within an acceptable distance from their home. If in the following there is any reference to a *capacity problem* it is meant that more students prefer to enrol at some particular high school than it can offer seats.

2.2 Amsterdam High Schools

This thesis investigates school-allocation mechanisms that have been used or may be used in the municipality of Amsterdam. To better understand the need for a good school-allocation mechanism in Amsterdam, it is important to first clarify the present situation of secondary education in Amsterdam.

In the City of Amsterdam, the local government, the municipality of Amsterdam, and the the representative body of all high schools and vocational education institutions in Amsterdam, the OSVO, agree on further legislation on the transfer from primary education to secondary education in the City of Amsterdam, every year, in a document called the 'Core Procedure Transfer Primary Education Secondary Education' (NL: kernprocedure Overstap PO-VO).[5] The regulations found in the Core Procedure include regulations on, the official school recommendations, application to high schools, the school-allocation mechanism, providence of information on high schools, and, priorities for students for specific high schools.

The Core Procedure only allows one or two *successive* secondary education tiers from the enumeration below, as a school recommendation (baring some rare exceptions)[6]

1. VMBO-B (possibly with LWOO);
2. VMBO-K (possibly with LWOO);
3. VMBO-T (possibly with LWOO);
4. HAVO;
5. VWO;

However, the Core Procedure does not allow VMBO-T to be recommended in combination with VMBO-K.[7] This means that in the City of Amsterdam students can have a school recommendation, 'VMBO-T or HAVO,' but not 'VMBO-T or VWO,' or, 'VMBO-B or VMBO-T.' At the end of the seventh class of primary school,[8] students will be given a preliminary school recommendation; in January of the eighth class the definite official school recommendation will be given, by at least two staff members of the primary school in question, where at least one is the teacher of the eighth class.[9]

Each prospective high-school student, after having received information on all available schools (and their capacity), officially applies to one high school (approximately mid March of the final year of primary school). The student does this by setting up a *strict* preference list of high schools, where the student puts the school (s)he prefers the most 'highest' (first) on the preference list. The school that is most preferred is the school where the student legally and de facto applies. This school checks if the preference list is in accordance with the high-school recommendation and if so inputs the preference list in the school-allocation mechanism.[10] If some student cannot be placed, by the school-allocation mechanism at the most preferred school, then the mechanism will try to allocate the student at some other school mentioned on the preference list, in

[5] OSVO & Gemeente Amsterdam (2015).

[6] Also the recommendations 'praktijkonderwijs', 'extra year of language education,' 'in between special provision,' 'special education,' 'high-school preparatory class for international students,' are allowed (cf. OSVO & Gemeente Amsterdam (2015): p. 9.).

[7] OSVO & Gemeente Amsterdam (2015): p. 9.

[8] The seventh class of primary school is internationally sometimes called the 'fifth grade'; it is normally the seventh year of primary school, where students are 10 to 11 years old.

[9] OSVO & Gemeente Amsterdam (2015): p. 10.

[10] OSVO & Gemeente Amsterdam (2015): §4.1, p. 22.

accordance with the order on the preference list. Prospective high-school students are urged to make the preference list as long as possible, such as to prevent that the student cannot be allocated at all to some school on his preference list. If a student cannot be placed by the mechanism, the student will be contacted by the proper authorities informing the student of this fact. All students who have not been allocated or that do not agree with the school that they are allocated to, will again go through a school-allocation mechanism, which will allocate them to some school that still has seats left, based on a new preference list over the schools that still have seats left.[11] Limited by very strict legal boundaries it is allowed for schools to make use of some priority rules for students. Examples of such priority rules are: 'every student who applied and who has siblings already enrolled, will be granted priority,' 'students applying, who have attended primary schools operating under a similar educational philosophy will be granted priority,' 'students applying whose parents are employees will be granted priority.' Students can only make use of some priority if they place the school offering the priority highest on their preference list;[12] if they do so, then they will be guaranteed a seat at that school if and only if there are enough seats to place all priority students.[13] Furthermore, compelling social or medical reasons can justify the use of the so called 'hardship clause' (NL: hardheidsclausule), which is a legal exception that can guarantee students a seat at some high school.

In recent years, 2013 to 2016, 7000 to 8000 students needed to be allocated to some high school.[14] In 2015, the year of the test data used in this thesis, there where 53 high schools, which together offer 135 application possibilities. A high school can offer multiple application possibilities, because it can offer multiple education tiers (e.g. VMBO-T, HAVO, VWO, etc.), special tracks (e.g. music, art, dance, etc.), and combinations of tiers (e.g. HAVO/VWO class, VMBO-T/HAVO class). The application possibilities for combinations of tiers are allowed for the first years of the secondary education such that students which cannot decide between two education tiers, have the opportunity to postpone their tier decision.

Nota Bene: In the following each application possibility will be regarded as a separate high school.

In 2015 of the approximately 8000 students, one quarter had a VWO school recommendation, another quarter a HAVO or HAVO/VWO school recommendation, and the other half a school recommendation for at least one type of VMBO.[15] Based on 2015 students' preference lists, there were 25 schools (19%) for which there were more students who placed the school highest on their preference list than the school had seats available. The latter means that the ratio of available seats of some school x divided by the number of students who place school x highest on their preference list, the capacity indicator, was greater than one for 25 schools. There were 7 HAVO, 8 VMBO and 10 VWO high schools that had a capacity indicator greater than one in 2015. The most extreme cases were: IJburg College 1 VMBO-K, with a capacity indicator of 4.7 (47 highest preferences for 10 seats), Spinoza Lyceum HAVO, with a capacity indicator of 3.4 (199 highest preferences for 58 seats), and Spinoza Lyceum VMBO-T Art Track (NL: kunstklas), with a capacity indicator of 2.8 (156 highest preferences for 56 seats).[16] Note that the capacity indicator tells something about the *actual* popularity of some school, since the preference lists that were handed in by students in 2015 were made under knowledge of the fact that handing in preference lists reflecting the true preferences was a dominant strategy for the school-allocation mechanism in use then; that is, no student could be hurt by handing in his or her actual true preferences.

[11] OSVO & Gemeente Amsterdam (2015): §4.1 in conjunction with §6.2.

[12] In 2015 students could also make use of their priority for schools not placed highest on their preference list, but students were only allowed to use one priority (OSVO & Gemeente Amsterdam (2014): p. 31 (§4.1), p. 35 (§5.1), p. 36 (§5.6), p. 28 (§5.8)).

[13] OSVO & Gemeente Amsterdam (2015): §4.3 in conjunction with p. 32.; Gautier et al. (2016): p. 14.

[14] Numbers of 2013 to 2015 from Gautier et al. (2015), p.4; numbers of 2016 published by the municipality of Amsterdam on their website on secondary education, URL: `https://www.amsterdam.nl/onderwijs-jeugd/voortgezet-onderwijs/`, retrieved 14 June 2016.

[15] Gautier et al. (2015): p. 3.

[16] The numbers for obtaining the capacity indicator come from Table 3 in Gautier et al. (2015), pp. 7-10.

2.3 School-Allocation Mechanisms in Amsterdam

The history of school-allocations mechanisms in Amsterdam can be described by identifying four periods (in chronological order): (i) the pre-lotteries period, (ii) the desynchronized lotteries period, (iii) the synchronized lotteries period, and (iv) the deferred-acceptance matching period. In the pre-lotteries period, no lotteries to fairly decide which student applications to accept were organised by high schools yet, because there was no *capacity problem*. In the first years that high schools faced more applications than they had seats available, two strategies were used by high schools to fairly decide which students to accept: they either organised a lottery, or accepted students based on time of application. The introduction of these decision mechanisms marked the beginning of the desynchronized lotteries period. Both decision mechanisms made high schools push the date of the decision on which students to accept forward each year. To overcome this, the municipality of Amsterdam and the representative body of high schools in Amsterdam, the OSVO, introduced, in 2005, strict rules for this decision process. These rules can be briefly summarised as follows:

1. each student was only allowed to apply to one school between set dates (by the OSVO);
2. high schools had to accept students based on a fair lottery, organised at a time set by the OSVO.

All students who were unfortunate in the first round of lotteries could in a next similar round apply for the schools that at that time still had seats left. In this second round, students could again only apply for one school, but high schools were now allowed to either organise a fair lottery (at a set time) or to base the decision on the time of application. Students who were unfortunate twice would then be allocated by the representative body, at some high school that still had seats available, while considering the specific wishes and needs of the student.[17] The introduction of these strict rules marked the beginning of the synchronized lottery phase, and the system introduced by the municipality of Amsterdam and the OSVO can be seen as a variant of the *Boston school-allocation mechanism* (BAM). The BAM can be briefly described as follows:[18]

0. Students hand in a preference list over (all) secondary schools;
1. Each school decides by a fair lottery which students, who listed the school as their first choice, to accept.
k. Each school, with seats still available, decides by a fair lottery who of the students, who listed it as their k-th choice, to accept.

The k-th round is repeated, until each student is assigned a seat. An important difference between the BAM and the system used in Amsterdam in the synchronized lotteries phase, is that students are not obliged to hand in a complete preference list over all high schools in advance, but are allowed to apply for one school (for which they want to participate in the lottery) at each stage. This means that students in the Amsterdam case can hand in their preferences *adaptively*, based on the outcomes of earlier rounds. The introduction of strict regulations for the high-school admissions was the introduction of the first real *school-allocation mechanism* in Amsterdam. From that point on, there was a uniform system in place to allocate students to high schools, that in principle could have been executed by one single interactive computer program. Hence, if in the following there is a reference to school-allocation mechanisms in Amsterdam, the Amsterdam Boston variant, discussed above, will be understood as a member of this collection.

In 2012 four researchers from the *Department for Research, Information and Statistics for the municipality of Amsterdam*, conducted a study on what parents saw as their 'ideal school', which schools parents preferred for their children, their opinions on the city's plan for new schools and on the school-allocation mechanism (i.e. the Boston variant).[19] It was found that many parents found the synchronized lotteries stressful and that most parents preferred a school-allocation mechanism where a preference list over high schools can be handed in before the allocation mechanism started. Furthermore, 6 out of 10 parents indicated that they would prefer to hand in a school preference list with the three schools most preferred by their child, ordered in order of preference; almost no parent supported the city's plan to request students to hand in preference

[17] OSVO & Gemeente Amsterdam (2013): §§ 4.1-2, 6.3, pp. 41-44, 64.
[18] Abdulkadiroğlu et al. (2005): p. 369; De Haan et al. 2015: pp. 5-6.
[19] Cohen et al. (2012): pp. 11-14.

lists of a minimal length of 15.[20] The study also showed that 8 out of 10 parents found the process of the synchronized lotteries clear.[21] In the same study, the researchers also found that in the year 2011, a school year in the synchronized lotteries phase, only 6% of students where unfortunate in the first round. Furthermore, they found that there were no significant differences in school performances between the first-round fortunate and unfortunate students. Moreover, of the unfortunate students almost all indicated that they were equally pleased with their life as the fortunate students.[22]

The municipality of Amsterdam and the representative body of high schools, OSVO, decided, based on the 2012 study, to introduce a new school-allocation mechanism in 2015: a deferred acceptance school-allocation mechanism. The choice for the Deferred acceptance school-allocation mechanism was mainly based on the following two reasons:

1. it allowed students to hand in one preference list of high schools in advance;
2. handing in a preference list according to the true preferences over high schools, is a (weakly) dominant strategy in deferred acceptance school-allocations mechanisms.

Deferred acceptance school-allocations mechanism match students to high schools according to preferences of the students over high schools *and* preferences of the high schools over students. Since in the Dutch school system high schools are obliged to accept all applying students (barring some rare exceptions) preferences of high schools over students are not allowed in a similar way as students have preferences over high schools. However, in the Amsterdam system, some students have priorities for certain schools, and these priorities can be seen as preferences of high schools over students (e.g. suppose two students s and s' both want to go to school a, and student s has a priority for school a and s' not, then this can be modelled by saying that school a prefers s over s'). Nevertheless, there are not enough priorities to create lengthy preference lists for the high schools, which deferred acceptance mechanisms need to function well. Therefore, preference lists of high schools over students are artificially augmented. The augmenting of school preferences can then be carried out by *single* or *multiple* tie-break. The distinction between *multiple tie-breaking* and *single tie-breaking* deferred acceptance school-allocation mechanisms lies in the following:

- *multiple tie-breaking*: every high school's preference list is randomly augmented by an individual lottery;
- *single tie-breaking*: one lottery determines the augmentation of each high school's preference list

If school priorities are incorporated in high schools' preference lists, then both under multiple and single tie-breaking not all high-school preference lists are identical, but under single tie-breaking all high-school preference lists are very similar. If school priorities are not incorporated in the high schools' preference lists and the preference lists are thus randomly generated, then under single tie-breaking all high schools' preference lists are identical, whereas under multiple tie-breaking this is not necessarily the case. The assignment of artificial preference lists over students to high schools, in the deferred acceptance school-allocation mechanisms, is where a 'lottery element' is incorporated in the allocation mechanisms. Based on the preference lists of the two parties (students and high schools) a deferred acceptance mechanism matches students to schools to create a high-school allocation. In 2015 the multiple tie-breaking deferred acceptance school allocation mechanism (DA-MTB) was used,[23] whereas in 2016 the single tie-breaking deferred acceptance school-allocation mechanism (DA-STB) was used; in both cases priorities were incorporated in the high schools' preferences.[24] In Chapter 3 (Theorem 4.19) it will be shown that deferred acceptance with single tie breaking, with no priorities incorporated in high school's preferences, is equivalent to the permutation school-allocation mechanism (PAM) which is an implementation of the random serial dictatorship school-allocation mechanism (RSD).

[20] Cohen et al. (2012): pp. 46-48.
[21] Cohen et al. (2012): p. 44.
[22] Cohen et al. (2012): p. 45.
[23] OSVO & Gemeente Amsterdam (2014): p. 44.
[24] OSVO & Gemeente Amsterdam (2015): p. 5.; Gautier et al. (2016): p. 2.

2.4 Deferred Acceptance in Amsterdam

The first use of a deferred acceptance mechanism marked the beginning of the contemporary deferred accepting matching period. In 2015 the DA-MTB was used to allocate 7510 students.[25] At that time, there were 9846 seats available at 53 schools, where schools, in this case, can contain multiple application possibilities (i.e. including all educational tiers offered by some the high school). There was a shortage of 998 seats to place all students at the school they preferred most. In the final allocation produced by the DA-MTB, 74% of the students were allocated to the school they preferred most[26] and 95% of the students were allocated to a school students placed in the top three of their preference list.[27]

The outcome of the 2015 allocation was severely criticized, because parents claimed to have been promised that 99.9% of the students would be allocated to a school students placed number one, two of three in their preference list.[28] Furthermore, parents complained that the outcome of the allocation was *Pareto inefficient*, which means that there were pairs of students, who would, by swapping allocated schools, both be allocated to a school they preferred more. The two criticisms led to a judicial process initiated by the parents against the municipality of Amsterdam and the representative body of high schools (OSVO) before the court of Amsterdam. The aim of the lawsuit was to force the OSVO to allocate students who had been allocated to a school sixth or lower on their preference list, to a school higher on their preference list and to allow students to swap schools.[29] Both requests were rejected by the court, because (i) the court did not agree that the OSVO had led parents to believe that 99.9% of the students would be allocated to a school in their top three,[30] and (ii) swapping of assigned schools could not be allowed, because it may create a market for school seats,[31] it would cease to give all students equal opportunities,[32] and it would render the mechanism *not strategy proof* in future use.[33]

Interestingly, the court ruling gives insight in why the representative body of schools (OSVO) and the municipality of Amsterdam explicitly opted for a *strategy-proof* school-allocation mechanism (which all deferred acceptance variants are from the perspective of the students). In strategy-proof school-allocations mechanisms, no student can be hurt by handing in a preference list, which reflects his or her actual true high-school preferences. The arguments to opt for a strategy-proof school-allocation mechanism, noted in the court ruling, are (i) it reduces the number of people who hand in preference lists that do not reflect true preferences,[34] (ii) it gives policy makers insight into the true preferences of students, which allows them to adjust school policy to actual needs,[35] and (iii) it facilitates communication, because parents and students can be genuinely instructed to hand in their true preferences, since this can never be in their own disadvantage.[36]

In the same month as the court proceeding, the OSVO requested a research company to conduct a satisfaction survey on the DA-MTB under parents, students, high-school staff and primary school staff. The study showed that 77% of the participating parents was pleased or very pleased about the DA-MTB-produced allocation. Furthermore, it became clear that 76% of the participating parents found it acceptable if their child would be allocated to a school in its 'top three'.[37] However, the study also showed that 53% of the 1143 participating parents and about half of the participating students, found it hard to identify 10 schools in order of preference. Moreover, even though 86% of the participating students were aware that it was beneficial to list as many schools as possible in their preference list, only 13% succeeded in enlisting at least

[25] Gautier et al. (2015): p. 4.
[26] Gautier et al. (2015): p. 19.
[27] Panel Inzicht (2015): p. 7.
[28] Rechtbank Amsterdam (2015): §3.3.
[29] Rechtbank Amsterdam (2015): §3.1.
[30] Rechtbank Amsterdam (2015): §4.6.
[31] Rechtbank Amsterdam (2015): §4.8.
[32] Rechtbank Amsterdam (2015): §4.8.
[33] Rechtbank Amsterdam (2015): §4.9.
[34] Rechtbank Amsterdam (2015): §4.5.
[35] Rechtbank Amsterdam (2015): §4.5.
[36] Rechtbank Amsterdam (2015): §2.7; Gautier et al. (2015b): pp. 4-5.
[37] Panel Inzicht (2015): p. 33.

10 schools.[38] In the first weeks of the school year, when students had enrolled in their assigned schools, the research company also assessed student satisfaction. Of students allocated to a school they ranked first or second on their preference list, 95% were pleased with the final allocation; the same was true for 50% of the students allocated to their third choice or lower. Nonetheless, 75% of the participating students allocated to a third choice or lower likes going to school at their allocated school, versus 91% for students allocated at their first or second choice.[39] Another important finding of the study was that 51% of the participating parents found the school allocation process as a whole (school choice, application and allocation) not transparent.

In 2015, in a written reply to the criticisms and the survey on the school-allocation mechanism, the municipality official for education Simone Kukenheim,[40] expressed her disappointment in the decision for a *Pareto inefficient* school-allocation mechanism.[41] Furthermore, she indicated five 'starting points', which should be taken into account by the decision for any school-allocation mechanism:[42]

1. students and parents should be, to the highest possible degree, able to control to which schools the student will be allocated;
2. a school-allocation mechanism in which more students are assigned their highest preferred school, should be favoured over other allocation mechanisms; students should only *not* be allocated to their highest preferred school, if it is not possible to allocate all students who prefer that school highest to that school;
3. the final allocation should not give students any reason to want to switch seats among each other;
4. the workings of the allocation mechanism should be clear to parents, students and schools and must be transparent;
5. the allocation mechanism should provide the final allocation within a couple of weeks.

In short, this means that the municipality of Amsterdam finds any school-allocation mechanism sufficient (1) if it respects preferences of students, (2) if it tries to place as much students as possible on their highest preferred school, (3) if it is Pareto consistent,[43] and (4) if it is comprehensible and (5) fast. Starting points 2, 3 and 5 can be safely translated into a mathematical setting. In fact, it can be demonstrated mathematically that the DA-STB, under certain conditions, satisfies starting points 3 and 5, and, to a certain degree, also 2. Since the DA-STB was adopted in some form, after this statement of the municipality official, it can be concluded that the authorities deem the DA-STB used in Amsterdam, to satisfy starting points 1 and 4. This conclusion gives a point of reference to determine if other school-allocation mechanisms satisfy these starting points 1 and 4.

In 2016 the DA-STB was used to allocate 7453 students. There were 10196 seats available, at 62 schools, where schools, in this case, can contain multiple application possibilities (i.e. including all educational tiers offered by some the high school). There was a shortage of almost a thousand seats to place all students at the school they preferred most. In the final allocation 83.66% of the students was placed at the school of their highest preference (for 2015 DA-MTB only 74%) and 95.23% was allocated a seat in a school in their top three.[44] Because it is expected under DA-STB that some students will be placed at a school very low on their preference list, students and parents were advised to hand in longer preference lists than previously, when the DA-MTB was used. The length of the preference lists was 7.72 on average (this was 5.25 in 2015 for DA-MTB), for VWO students this was 11.19 (this was 6.7 for DA-MTB).[45] The DA-STB was unable to allocate 37 students (0.005%) of which 11 students (30% of 37) had only one school on their preference list, 18 students (49% of 37) had less than four schools, and 9 students (19% of 37) ten or more schools. Students

[38] Panel Inzicht (2015): p. 29.
[39] Panel Inzicht (2015): p. 34.
[40] A political appointment in the executive branch of the municipal government (NL: Wethouder Onderwijs).
[41] Kukenheim (2015): pp. 2-3.
[42] Kukenheim (2015): pp. 3-4.
[43] N.B. a school-allocation mechanism that produces Pareto-optimal allocations is called Pareto consistent.
[44] OSVO (2016); Gautier et al. (2016): p. 15.
[45] Gautier et al. (2016): p. 10.

with a VWO school recommendation were overrepresented among the not-allocated students (49% of 37).[46]

The representative body of high schools organised another 'run' of DA-STB for all students who were not allocated or unhappy with their assignment. These students could hand in a new preference list over schools that (by now) still had seats available. 21 of the 37 not-allocated students and 46 previously allocated students participated in this second round. All students participating in this second round were allocated to the school of their first choice (in their *new* preference list over still available schools).[47]

In 2016, 1020 students (14%) had placed a school for which they were offered priority, highest on their preference list (in comparison, in 2015, when the DA-MTB was used, 811 students (11%) placed a school for which they were offered priority highest). In the specific DA-STB adopted in Amsterdam, students could only use their priority if they placed the school for which they had a priority highest on their preference list; in the DA-MTB, used the year before, students were allowed to use their priority (only one) for a school on any place in their preference list. In 2016, there were some schools, for which more students were offered priority than the school had seats available; which students were accepted by that school was then decided by the single tie break.[48]

2.5 Looking Ahead

At the time of writing this thesis, no satisfaction survey has yet been conducted on the DA-STB, used in 2016. There is a satisfaction survey planned and its results will be presented to the city council of Amsterdam in September 2016.[49] Before the publication of these results, it is hard to say how pleased parents, students, high-school staff and primary school staff are with the mechanism and the allocation produced by the mechanism. The mechanism still led to some very unfortunate students' high-school assignments and as it required longer preference lists to be handed in, it will thus be very interesting to see how parents and students have experienced this year's allocation process.

In the following the different school-allocation mechanisms used in Amsterdam will be discussed from a mathematical point of view. Each mechanism will be explained and investigated for the mathematical properties desired by policy makers (and society). Some of these mechanisms will be treated in the form in which they have been used in Amsterdam and others will be treated more generally. For the mathematical investigation of school-allocation mechanisms, 'school allocation' needs to be mathematically translated. In the next chapter, Chapter 3, all formal concepts needed to formally discuss the different school-allocation mechanisms will be introduced. In Chapter 4, it will be shown what kind of allocations the different mechanisms produce and what the game-theoretic consequences of different mechanisms are.

In light of the 'starting points' given by the municipality official for education, a new approach to school-allocation mechanisms will be presented in Chapter 5. It will be explored if using *genetic-algorithmic optimisation* could help in providing a school-allocation mechanism which satisfies more of the desired properties than the other discussed school-allocation mechanisms. The benefit of using a genetic optimisation method is that a *fitness function* needs to be introduced, which determines which allocations will 'survive', and will thus be produced by a genetic-algorithmic optimised school-allocation mechanism. The political wishes for school allocations can be incorporated into and formally captured by this fitness function, which makes a genetic-algorithmic optimised school-allocation mechanism a very legitimate school-allocation mechanism.

[46] The DA-STB is unable to place some student if all schools on the preference list of the students have no seats left at the stage of mechanism where it is allocating the student (cf. Kukenheim (2016): p. 2.; Theorem 4.19).
[47] OSVO (2016); OSVO (2016); Gautier et al. (2016): p. 11.
[48] Gautier et al. (2016): p. 14.
[49] Kukenheim (2016): p. 3.

Chapter 3

School Allocations

3.1 School Allocations

The Amsterdam school choice case can be formally described as an allocation problem between two disjoint sets of agents: students applying for Amsterdam's high schools and the high schools of Amsterdam. The particularities of the Amsterdam case will only be mentioned in this formal treatment of the problem when strictly necessary. Consequently, in the following no longer will be referred to the actual students applying for high schools in Amsterdam, but to a finite set of n students $\mathcal{S} = \{s_1, s_2, ..., s_n\}$, where each s_i denotes a student applying to go to high school h_i from a finite set of m high schools $\mathcal{H} = \{h_1, h_2, \ldots, h_m\}$, where we associate a capacity (or quota)[1] c_h to each high school; the set containing all such capacities composes a function from high schools to positive integers greater than zero, s.t. $\mathcal{C} : \mathcal{H} \to \mathbb{Z}_{>0}$ and $\mathcal{C} = \langle (h_1, c_{h_1}), (h_2, c_{h_2}), ..., (h_m, c_{h_m}) \rangle$. These capacities indicate the maximum number of students who can be allocated to some high school from $\mathcal{H}$. Since all students rightfully applying for high schools in Amsterdam have a right to receive secondary education, the objective is to produce an allocation such that each student is matched to some high school from which it will receive its secondary education in which no high school is assigned more students than for which it has seats. The following definition captures the concept of an allocation mathematically.

Definition 3.1 (Allocation) *Let $\mathcal{S}$ and $\mathcal{H}$ be two finite disjoints sets and let $\mathcal{C} : \mathcal{H} \to \mathbb{Z}_{>0}$ be a function from $\mathcal{H}$ to positive integers greater than zero, s.t. for each $h \in \mathcal{H}$ there is some positive integer greater than zero $\mathcal{C}(h) = c_h$ denoting the capacity of h. An allocation A of elements of $\mathcal{S}$ to elements of $\mathcal{H}$ is a function $A : \mathcal{S} \to \mathcal{H}$, such that for all $h \in \mathcal{H}$: $|\{s \in \mathcal{S} | A(s) = h\}| \leq \mathcal{C}(h)$.*

The disjoint sets of agents underlying an allocation are also called *parties*. The following example clarifies the mathematical understanding of a school allocation:

Example 3.2 *Say that there are two high schools a and b, s.t. a has capacity 1 and b has capacity 2. Furthermore, assume that there are three students, student 1, 2 and 3. Formally this gives $\mathcal{S} = \{1, 2, 3\}$, $\mathcal{H} = \{a, b\}$, and $\mathcal{C} = \{(a, 1), (b, 2)\}$. A possible allocation, given these students and high schools could be $A = \{(1, a), (2, b), (3, b)\}$, which captures the allocation in which student 1 is assigned to school a and students 2 and 3 are assigned to school b. Since no school is assigned more students than its capacity, the sets $\mathcal{S}$ and $\mathcal{H}$ are disjoint and finite, A is an allocation according to Definition 3.1.*

[1] cf. Roth & Sotomayor (1990): p. 126.

3.2 Student Preferences

In the Amsterdam case, the goal is to produce allocations that respect the preferences of students over high schools. Consequently, to each student $s \in \mathcal{S}$ a *list* is associated capturing these preferences. Before the formal treatment of these preferences via *lists* can be introduced, it is important to make some remarks about modelling school preferences. Preference lists aim to represent how a student would choose among different alternatives when faced with a choice. If some student s prefers school h over h', this is meant to represent the fact that, if student s were faced with a choice from a set of high schools which to attend, s would never choose h', if h was available as a choice as well. If, however, student s would choose either one in the foregoing situation, it is said that s is *indifferent* between h and h'.[2] With this understanding of preferences in mind, three further assumptions on preferences will be made, namely that preference lists are (i) *total*, (ii) *transitive*, and (iii) *strict*. The first two assumptions are customary in economics and attempt to capture *rationality*. A preference list is *total*, when all schools mentioned in the preference lists can be compared, that is, for every two schools h and h' mentioned in the preference list, the student is able to say which school he likes more or that he is indifferent between the two. A preference list is *transitive*, if for all triples of schools h, h', h'', it holds that, if school h is preferred over h' and h' is preferred over h'', than school h is preferred over school h''.[3] A preference list is *strict* if for every two schools mentioned in the preference list, the student is not indifferent. Another example of a total, transitive, strict ordering is the list of positive integers from 0, 1, 2, onwards. The modelling of preferences adopted here is called an ordinal modelling of preferences, which means students have a first choice, second choice, third choice, etc. Furthermore, it is not necessary for every high school h from $\mathcal{H}$ to occur in some preference list. [4] The adoption of such a model of preferences to capture the Amsterdam case is justified on the following three grounds:

1. the 2012 study of the research department of the municipality of Amsterdam suggested that parents wanted students to be able to hand in preferences as ordinal preference lists;[5]

2. ordinal preference list are able to capture 'incomplete preference lists', that is, preference lists in which students do not have to indicate in which order they prefer *all* high schools, but only which high schools they prefer most (this incomplete preference list was the preference list format in use in Amsterdam in 2015 and 2016);

3. there is no empirical data of preference lists with indifferences nor of cardinal preference lists available, hence allowing such preference list would make it impossible to critically test new suggested allocation mechanisms.

In the foregoing it was announced that the preference lists of students will be formally modelled as mathematical *lists*. To introduce the formal understanding of preference lists, the mathematical notion of a *permutation* needs to be introduced first.

Definition 3.3 (Permutation) *Let A be some finite set with cardinality n. Let $\Pi_A = \{\pi \mid \pi : n \to A\}$, that is, Π_A is the set of functions from n to A (N.B. n is an ordinal, i.e. $n = \{0, 1, 2, ..., n-1\}$). Let Π_A be the set of permutations of A s.t. every $\pi \in \Pi_A$ is some permutation of A. A permutation π is called a* random permutation *of A if some π is chosen uniform from Π_A, this is denoted by $\pi \xleftarrow{r} \Pi_A$.*

Permutations are intuitively best understood as orderings of sets, e.g. $\langle 1, 3, 2\rangle$ is a permutation of the set $\{1, 2, 3\}$. Permutations are understood as functions, viz. bijections, such that it is possible to refer to the i-th element of some permutation and vice versa. Let π be the permutation $\langle 1, 3, 2\rangle$ of the set $\{1, 2, 3\}$; formally π is then a specific bijection from $\{0, 1, 2\}$ to $\{1, 2, 3\}$, namely the function (i.e. set of pairs) $\{(0, 1), (1, 3), (2, 2)\}$; with this representation it is possible to denote the first element of π by $\pi(0)$ and to denote the position of 3 in π as $\pi^{-1}(3)$, which is 1. With Definition 3.3 a definition of *lists* can be given:

[2] Roth & Sotomayor (1990): pp. 17-18.
[3] Roth & Sotomayor (1990): pp. 18-19.
[4] Manlove (2013): p. 1.
[5] Cohen et al. (2012): pp. 46-48.

Definition 3.4 (List) *The surjection $\vec{A} : m \to A$ is a* list*, if there exists some finite set A with cardinality n and finite ordinal m s.t. $m \geq n$. Some operations on lists: Let $\vec{A}$ and $\vec{B}$ be lists with cardinality resp. n and m,*

- *Translation: $T_\delta(A) = \{(a + \delta, b) \mid (a, b) \in A\}$;*
- *Concatenation: $\vec{A} + +\vec{B} = \vec{A} \cup T_n(\vec{B})$;*
- *Add (an element): $a : \overrightarrow{A} = \overrightarrow{\{a\}} \cup T_1(\overrightarrow{A})$;*
- *Tail of a list: $\tau(\overrightarrow{A}) = T_{-1}(\{(i, a) \in \overrightarrow{A} \mid i > 0\})$;*
- *k Initial Segment: $\iota(k, \overrightarrow{A}) = \{(j, a) \in \overrightarrow{A} \mid j < k\}$*

Nota Bene: (i) $T_\delta(\vec{A})$ is a function, but not a list if $\delta > 0$; (ii) $\vec{A} + +\vec{B}$ is a list; (iii) $a : \overrightarrow{A}$ is a list. Notation: to stress that some object is a list, an arrow will be written over it; furthermore, angle brackets (or 'chevrons') are used to give a list as an ordering of elements, e.g. $\langle a_1, a_2, a_3\rangle$ is a list (representation) of $\{a_1, a_2, a_3\}$ and formally is the set $\{(0, a_1), (1, a_2), (2, a_3)\}$.

From the above definition it follows that all permutations are lists, but not that all lists are permutations.

Example 3.5 *Let $A = \{1, 2, 3\}$, then $\pi = \langle 1, 2, 3\rangle$ is a list (or 'list representation' of A) and a permutation of A, but $\overrightarrow{A'} = \langle 1, 1, 2\rangle$ is a list based on $A' = \{1, 2\} \subseteq A$, but not a permutation of A' (nor of A). Furthermore note that we have:*

$$\overrightarrow{A'} = \langle 1, 1, 2\rangle = \{(0, 1), (1, 1), (1, 2)\}.$$

Some important remarks on lists. First, note that $\vec{A}(i)$ denotes the $(i + 1)$-th element of the list $\vec{A}$ and that $\overrightarrow{A}^{-1}(a)$ gives the position (or 'index') of a in $\overrightarrow{A}$. Second, note that the *translation* operation from Definition 3.3 is an operation on sets of pairs (functions) and is only used to define the operations on lists and is not used anywhere else. Third, note that operation of *concatenation* can be intuitively understood as simply concatenating two lists to make one longer lists, e.g. $\langle 1, 2\rangle + +\langle 2, 1\rangle = \langle 1, 2, 2, 1\rangle$. Fourth, note that *adding an element* always adds an element at the beginning of the list, e.g. $a : \langle b\rangle = \langle a, b\rangle$, if a needs to be added to the end of $\langle b\rangle$, $\langle b\rangle + +\langle a\rangle$ will do. Fifth, the tail of some list $\langle a_1, a_2, ..., a_n\rangle$ is just $\langle a_2, ..., a_n\rangle$ s.t. the latter is still a list. Sixth, a k *initial segment* of some list $\overrightarrow{A}$ is the list of the first k items of $\overrightarrow{A}$; any initial segment $\overrightarrow{A'}$ of some list $\overrightarrow{A}$ is a k *initial segment* for some $k \leq |\overrightarrow{A}|$. Seventh, note that the empty set $\varnothing$, is also a list; it is is usually called *the empty list* and is sometimes denoted by $\langle\rangle$. Seventh, observe that, $\overrightarrow{A} + +\langle\rangle = \overrightarrow{A} = \langle\rangle + +\overrightarrow{A}$ and $a : \langle\rangle = \langle a\rangle = \overrightarrow{\{a\}} = \{(0, a)\}$. With lists explained, the formal understanding of preference lists can be introduced:

Definition 3.6 (Preference List) *Let A and B be two finite disjoints sets. Let Π_B be the set of all permutations of B. A* preference list *P_a of some $a \in A$ over B is an initial segment of some $\pi \in \Pi_B$.*

In the following a preference list will be called *complete* if it is a permutation of the other party. Furthermore, given some preference list P_s of some student s that contains elements h and h' from the other party, the relation 's prefers h over h'' will be denoted by $h >_s h'$. Note, that if $h >_s h'$, then $P_s^{-1}(h) < P_s^{-1}(h')$. That is, if a student prefers h over h', then the index of h is lower than the index of h' in the respective preference list. Moreover, the k-th *preference position* of student s is $P_s(k - 1)$ and the preference position of high school h for student s is $P_s^{-1}(h) + 1$. Hence, the *first choice* of some student s, $P_s(0)$, is preference position 1. Furthermore, with the 'top n' of student s the initial segment with length (i.e. cardinality) n of P_s is referred to. Moreover, note that the set of all preference lists $\mathcal{P} = \{P_{s_1}, P_{s_2}, ..., P_{s_n}\}$ is called a *preference profile*. The set of preference lists of students over high schools is therefore called the preference profile of the students.

Example 3.7 *Let* $\mathcal{H} = \{a, b, c\}$ *and* $\mathcal{S} = \{1, 2, 3\}$*, with* $\mathcal{C} = \{(a, 1), (b, 2), (c, 1)\}$*. Furthermore, say*

- $P_1 = \langle a, b, c \rangle$*;*
- $P_2 = \langle b, a, c \rangle$*;*
- $P_3 = \langle b, c \rangle$*.*

Then the preference lists of students 1 and 2 (i.e. resp. P_1 *and* P_2*) are* complete *and the preference list of student 3 is* incomplete*. Furthermore, student 1 has school a on preference position 1, school b on preference position 2 and school c on preference position 3, whereas student 3 has no preference position 3. Moreover, student 2 his first choice is b, his second choice is a and so on; the 'top 2' of student 2 is* $\langle b, a \rangle$*.*

3.3 School Markets

The study of allocations between two parties that respects preferences of one of the parties or of both parties, originally belongs to economics. In particular, finding suitable allocations is studied in a sub field of microeconomics called *market design*, where the theory on 'suitable allocations' is denoted by 'matching theory'. Matching theorists are mainly concerned with applying game-theoretic techniques, to formalize economic properties of allocations, such that 'market failures' can be prevented. The origin of 'matching theory' in economics, explains why the single concept to capture all concepts introduced in this section before, is called a 'school *market*':[6]

Definition 3.8 (School Market) *Given a finite set of n students* $\mathcal{S}$*, a finite set of m high schools* $\mathcal{H}$*, a capacity assignment* $\mathcal{C} : \mathcal{H} \to \mathbb{Z}_{>0}$ *and a preference profile of the students* $\mathcal{P} = \{P_s \mid P_s$ *is the preference list of student* $s \in \mathcal{S}\}$*, the quadruple (4-tuple),* $\mathcal{M} = (\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$ *is called a* school market *iff the sum of all capacities is at least the cardinality of* $\mathcal{S}$*.*

Observe that a school-allocation problem, as the Amsterdam case, is only referred to as a market, if and only if all student are able to receive secondary education somewhere. The problem is therefore not, *if* or *how* the students can get secondary education, but *where*.

3.4 Priorities

In Amsterdam there have been different rules regarding priorities. Priorities are rights that some students can obtain to be admitted to some high schools. Priorities are only allowed within very strict legal boundaries and can only be used in some exceptional cases. Examples of such priority rules are: 'every student who applied and who has siblings already enrolled will be granted priority,' 'students applying who have attended primary schools operating under a similar educational philosophy will be granted priority,' 'students applying whose parents are employees will be granted priority.' Before 2016 it was possible for students to apply to some high school via a priority, even if this school was not placed highest on their preference list; nowadays this is no longer permitted. Students can only use their priority right if they place the school for which they have such a right highest on their preference list. The priority right then guarantees students a seat on the respective high school if and only if the number of students with priorities for that high school does not exceed the capacity of that high school. If the latter is the case, then a lottery decides which students will be guaranteed seats.[7]

The priorities in Amsterdam form no *system of priorities*, that is, there is no order over priorities such that some priorities weigh heavier than others.[8] About 14% of the students used a priority right in 2016, by

[6] Manlove (2013): p. 8.
[7] OSVO & Gemeente Amsterdam (2015): §4.3 in conjunction with p. 32.; Gautier et al. (2016): p. 14.
[8] In Boston there is such a *system of priorities*, cf. Abdulkadiroğlu et al. (2005): p. 368.

placing a school for which they had a priority, highest on their preference list. Priorities can be formalised by regarding them as preferences of high schools over students. That is, a high school h will be said to prefer student s over s' if student s is offered priority for h and s' is not; if both are offered priority or both not, then school h is said to be indifferent on s and s'. The number of priorities in Amsterdam is so low, and the system of priorities so simple, that the preference lists for high schools over students, obtained by incorporating priorities, are very 'indifferent'. The latter will lead to new problems, namely to decide upon which tie-breaking rules to use. Tie-breaking rules decide which student to favour if only some can be assigned to some school h to which all students want to be assigned and on which h is indifferent. In the contemporary rules for priorities in Amsterdam, this issue can be circumvented, since priorities can be dealt with, before the allocation mechanisms come in to play. That is, first all students with priority rights will be allocated according to their priorities. Then, a school-allocation mechanism is used on the remaining school market, that is the market that only contains students not yet assigned (i.e. students who did not use priorities and students who were eliminated by lottery for their priority school) with properly adjusted capacities. The school-allocation problem regarding the so obtained school market, is the school-allocation problem on which this thesis focusses. In the following priorities are therefore neglected. There is thus no need for a discussion on which tie-breaking rule to use, nor do school-allocation mechanisms that cannot cope with priorities need to be dismissed, since the Amsterdam school-allocation problem can be regarded as a school-allocation problem in which there are no priorities.

3.5 Properties of Allocations

As indicated earlier, finding allocations is not just to find some matching between two disjoint parties, but to find matchings that satisfy certain conditions. In the Amsterdam case, some of the conditions which a school allocation must satisfy can be taken from the written reply to the criticisms on the DA-MTB produced allocation of 2015. In her reply, the municipality official for eduction Simone Kukenheim, indicated five 'starting points' which should be taken into account in the design of a (new) school-allocation mechanism for the Amsterdam school case. These 'starting points' suggest which mathematical properties the Amsterdam school allocation should have. The 'starting points' can be summarized as follows, the school-allocation mechanism (1) should respect preferences of the students, (2) should try to place as much students as possible on their first choice, (3) should not give rise to reasonable trading options between students, and (4) it should be comprehensible and (5) fast.[9] As indicated in the previous chapter (cf. Section 2.4) starting points 2, 3 and 5 allow a mathematical translation, however only 3 has a univocal translation, viz. *Pareto optimality*. An allocation A is Pareto optimal, if there cannot be some allocation A', such that in A' some students are better off than in A, whereas no students in A' are worse off than in A. It is however non trivial that the concept of *Pareto optimality* translates the third starting point of *no reasonable trade options*. In this section it will be argued, why this concept, and not the more direct translation of *pair-wise Pareto efficiency* is a formal translation of the third starting point of the municipality official.

An allocation is *pair-wise Pareto efficient* if there are no pairs of students who want to switch schools. This can be expressed by the condition that there cannot be two students s and s' s.t. s prefers the school to which s' is assigned over the school to which he is assigned and vice versa.

Definition 3.9 (Pair-wise Pareto-Efficient Allocations) *Let $A : \mathcal{S} \to \mathcal{H}$ be some allocation between two finite disjoint sets $\mathcal{S}$ and $\mathcal{H}$. The allocation A is* pair-wise Pareto efficient *iff:*

$$\forall s, s' \in \mathcal{S} : A(s) <_s A(s') \Rightarrow A(s') >_{s'} A(s)$$

Intuitively, some allocation is pair-wise Pareto efficient, if for every student, for every school on his or her preference list, there is no student which is assigned a school that the student prefers over its present assigned school, is willing to switch with the student in question. Although pair-wise Pareto efficiency is a desirable property for school allocations, the property by itself does not capture everything the municipality of Amsterdam wants for its school allocations. Consider the following example:

[9] Kukenheim (2015): pp. 3-4.

Example 3.10 *Let $\mathcal{S} = \{1, 2\}$ and let $\mathcal{H} = \{a, b, c\}$, such that all high schools have capacity 1. Say we have the following preference lists:*

- $P_1 = \langle a, b, c \rangle$;
- $P_2 = \langle a, c, b \rangle$.

Consider the allocation $A = \{(1, b), (2, c)\}$. Now student 1 and 2 do not want to switch, that is, $A(1) >_1 A(2)$ and $A(2) >_2 A(1)$, hence A is pair-wise Pareto efficient. Note, however, that allocation A could be 'improved' to some allocation A' by assigning school a to either 1 or 2, consider the allocation $A' = \{(1, a), (2, c)\}$. In allocation A', student 1 has preference position 1 and student 2 has preference position 2, whereas under A both students had preference position 2.

From the above example it is clear that it is possible to 'improve' pair-wise Pareto-efficient allocations, such that no student is assigned a lower preference position, whereas some student is assigned a higher preference position. If the latter is possible, then the allocation is *not* Pareto optimal. A closer look at Example 3.10 suggests that it might be wise to not only desire pair-wise Pareto-efficient allocations, but also *saturated* allocations. An allocation is *saturated* if all schools which students prefer over the school they are assigned to have reached their full capacity.

Definition 3.11 (Saturated Allocations) *Let $A : \mathcal{S} \to \mathcal{H}$ be some allocation between two finite disjoint sets. Let $\mathcal{C} : \mathcal{H} \to \mathbb{Z}_{>0}$ be the capacities of all $h \in \mathcal{H}$. The allocation A is* saturated *iff:*

$$\forall s \in \mathcal{S}\ \forall h \in \mathcal{H} : h >_s A(s) \Rightarrow |@(h)| = c_h,$$

with c_h the capacity of h and $@(h) = \{s \in \mathcal{S} \mid A(s) = h\}$.

Intuitively *saturated* school allocations are such that it is not the case that some (popular) high school has seats left, but students who want to go there, are not allocated to that school, but to some other school. The following example will demonstrate, that although pair-wise Pareto-efficient saturated allocations fulfil important conditions that the municipality of Amsterdam desires, they still allow objective 'improvement':

Example 3.12 *Let $\mathcal{S} = \{1, 2, 3\}$ and let $\mathcal{H} = \{a, b, c\}$, such that all high schools have capacity 1. Say we have the following preference lists:*

- $P_1 = \langle b, a, c \rangle$;
- $P_2 = \langle c, b, a \rangle$;
- $P_3 = \langle a, c, b \rangle$;

Consider the allocation $A = \{(1, a), (2, b), (3, c)\}$. Note that A is pair-wise Pareto efficient, since (i) student 1 and 2 do not want to switch places, because 2 doesn't want to, (ii) student 1 and 3 do not want to switch, because 1 does not want to, and (iii) student 2 and 3 do not want to switch, because 3 does not want to. Furthermore, A is saturated, because all schools have reached their full capacity. Now consider allocation $A' = \{(1, b), (2, c), (3, a)\}$, in this allocation all students are assigned their first choice, whereas under A they all were assigned their second choice. This means that there exist an allocation, namely A', in which some students are better off (viz. all students) and no students are worse off. Hence, A is not *Pareto optimal. Allocation A' cannot be reached from A by a pairwise switching of places, s.t. each switch is an improvement for all participants, but it can be reached by a* trade cycle*. Consider the following trade cycle, $1 \mapsto 2 \mapsto 3 \mapsto 1$, which captures the idea that if 1 goes to the school of student 2, student 2 goes to the school of student 3, and student 3 goes to the school of student 1, then all students are at a school they prefer over the school assigned under A. Clearly this trade cycle benefits all participants, because before all students were assigned their second choice and afterwards all are assigned their first choice, hence pair-wise Pareto efficient saturated allocations are not optimal allocations yet.*

The example above illustrates that pair-wise Pareto efficiency only tells something about pair-wise trades of school seats from which both students benefit, and that the impossibility of such pair-wise trades does not imply the impossibility of trade cycles that benefit all students in the cycle. If a trade cycle that benefits all participants is possible in some allocation, then it cannot be that the allocation is Pareto optimal. In other words, if some allocation is Pareto optimal then such trade cycles are impossible (the contrapositive of the former statement). With these insights, a formal definition of Pareto optimality can be given:

Definition 3.13 (Pareto-Optimal Allocations) *Let $A : X \to Y$ be some allocation between two finite disjoint sets. The allocation A is Pareto optimal iff there is no allocation $A' : X \to Y$ s.t.*

(i) $\exists x \in X : A(x) < A'(x)$;

(ii) $\forall x \in X : A(x) \leq A'(x)$.

Note that the allocation A' from Example 3.12 is a Pareto-optimal allocation. Furthermore, with the concept of Pareto optimality now formalised, the previous examples (Examples 3.10 and 3.12) establish the following two propositions:

Proposition 3.14 *If some allocation is pair-wise Pareto efficient, then it is not necessarily Pareto optimal.*

Proposition 3.15 *If some allocation is saturated and pair-wise Pareto efficient, then it is not necessarily Pareto optimal.*

The above propositions suggest that Pareto optimality is a stronger property than the combined property of pair-wise Pareto efficiency and saturatedness. This suggestion is correct, since if some allocation is Pareto optimal, then it is impossible that there can be two students who, if they switch, they are both better off (otherwise the allocation was *not* Pareto optimal). Furthermore, if some allocation is Pareto optimal, then it is impossible that it is not saturated, since a trivial 'improvement' of the allocation would just be to assign the students for which the school that they prefer over their present assigned high school, to that more preferred school. The foregoing implies that every Pareto-optimal allocation is pair-wise Pareto efficient and saturated.

Proposition 3.16 *Every Pareto-optimal allocation is pair-wise Pareto efficient and saturated.*

The municipality official suggested that a suitable allocation for the Amsterdam school choice case did not allow any reasonable trading options for students. From the above it is clear, that it does not suffices to only allow no pairwise trading options, nor to only allow saturated pair-wise Pareto-efficient allocations, to capture this idea. The stronger notion of Pareto-optimality does capture this idea, and, furthermore, implies that there are no finite trading cycles possible either. The notion of Pareto optimality is also found in the literature as an important 'optimality criterion' for one-sided one-to-many matchings (as the Amsterdam school case) and it is regarded by economists as a "minimal requirement for any 'reasonable' solution" to these kind of matching problems.[10] In the following we will denote the set of Pareto-optimal allocations for some school market $\mathcal{M}$ by $PAROPT_{\mathcal{M}}$.

The other two 'starting points' of the municipality official that allow mathematical translation were the mechanism *should respect preferences of the students* and *should try to place as much students as possible on their first choice*. The first of these two is not univocal: different mechanisms 'respect' the preferences of students in different ways. To decide whether a system 'respects' student preferences can therefore be explained in different ways. For example, one understanding could be 'handing true preferences is a dominant strategy', another 'the full top n of a student is accounted for by the mechanism'. This condition will be

[10] Manlove (2013): pp. 5-6, 304; cf. Hylland & Zeckhauser (1979): p. 293.

treated when different known mechanisms are discussed and when strategic questions are investigated. The other 'starting point' mentioned in the opening paragraph, of *the more students are assigned their first choice, the better*, is also not univocal; the words of the municipality official "I advocate a system in which more students are assigned their first choice", can be explained in multiple ways. Three possible understandings are:

1. I advocate a system in which the more students are assigned their first choice the better;
2. I advocate a system in which students should only *not* be allocated to their highest preferred school, if it is not possible to allocate all students who prefer that school highest to that school;
3. I advocate a system in which more students are assigned their first choice, *than under DA-MTB allocations.*

The second understanding is based on an example that the official gives prior to the cited standpoint, which is the following: "on 28 schools students are eliminated by lottery, whereas on 14 of these schools the number of students who indicated it as their first choice was higher than the capacity".

Example 3.12 clearly demonstrated, that it is not sufficient to demand pair-wise Pareto-efficient allocation (even combined with saturatedness) for all of the three above understandings, since in Example 3.12, all three students could be assigned their first choice, but assigning all of them their second choice still yields a pair-wise Pareto-efficient and saturated allocation. In spite of the fact that Pareto-optimality captures many of the desired properties of allocations (pair-wise Pareto efficiency, saturatedness, no trade cycles) allocations which are Pareto optimal do not necessarily assign as much students to their highest preferred school as possible. Consider the following example:

Example 3.17 *Let $\mathcal{S} = \{1,2,3,4\}$ and let $\mathcal{H} = \{a,b,c\}$, s.t. school a has capacity 2 and all other schools have capacity 1. Say we have the following preference lists:*

- $P_1 = \langle a,b,c\rangle$;
- $P_2 = \langle a,b,c\rangle$;
- $P_3 = \langle b,a,c\rangle$;
- $P_4 = \langle b,a,c\rangle$.

Let $A = \{(1,c),(2,a),(3,a),(4,b)\}$ be some allocation that is produced by assigning students the highest preferred still available school in the following order, first student 4, than 3, than 2, than 1. Then allocation A is Pareto optimal (result established in next section; Theorem 4.5). Observe that also the allocation $A' = \{(1,a),(2,a),(3,c),(4,b)\}$ is possible. In A' three students are assigned their first choice (viz. students $1,2$ and 4), whereas in A only two students are assigned their first choice (viz. students 2 and 4). Futhermore, under A' all students who preferred school a the most are now assigned to school a, which was not the case under A.

The above example demonstrates that Pareto-optimal allocations do not necessarily satisfy the second starting point (in the first and second understanding) of the municipality official. Note, however, that both A and A' are Pareto-optimal allocations.[11] The example shows that Pareto optimality can satisfy two of the 'starting points', but does not necessarily do so. To produce allocations which always satisfy the wishes of the policy makers, more than Pareto-optimality should thus be demanded. Note that Pareto optimality should nevertheless always be demanded to let the allocation satisfy the municipality official's third 'starting point'. Furthermore, observe that it can never be the case that among the allocations that assign the most students to their first choice, there is no Pareto-optimal allocation. Example 3.17 thus shows that Pareto-optimality is a necessary property for allocations to have, but it is not a sufficient property. In the following section the allocations mechanisms occurring in Amsterdam's school-allocation history will be treated (i.a. DA-STB, DA-MTB, Boston Allocation Mechanism). For all the treated mechanisms it will be shown which properties, discussed in this section, the allocations, produced by the treated mechanisms, have, and furthermore, if the allocations satisfy the municipality official's recommendations.

[11] Assign in the order $1,2,4,3$ the students their highest still available school, cf. Theorem 4.5.

Chapter 4

School-Allocation Mechanisms

4.1 Mechanisms

Bipartite matching problems, as the Amsterdam school-allocation problem, come in different forms. Bipartite matching problems can be *one-sided* or *two-sided* and can be *one-to-one* or *one-to-many*. The distinction between one-sided and two-sided matching problems is based on how many parties have preferences. The Amsterdam school-allocation problem, in its present formal format, is thus a one-sided Bipartite matching problem. The distinction between *one-to-one* or *one-to-many*, refers to nature of the matching, if individuals are matched with individuals, the problem is *one-to-one*; if individuals are matched to sets, the problem is *one-to-many*. In the Amsterdam school-allocation problem high schools (individuals) are matched to sets of students. The Amsterdam problem is therefore naturally modelled as a *one-to-many one-sided Bipartite matching problem*.

The introduced distinctions between *one-sided* and *two-sided* are important, because these classes indicate which solution concepts are desirable. In general, the central solution concept for two-sided matching problems is *stability*, and for one-sided matching problems is *Pareto optimality*.[1] The distinction between *one-to-one* and *one-to-many* is important for choosing which mechanism can be used to produce an appropriate matching. This latter distinction is sometimes purely semantic, since some problems can be framed in either 'class'. First of all, note that any one-to-many matching problem (as the Amsterdam school case) can be translated into a one-to-one matching problem. If one focusses on *school seats*, in stead of *schools*, the Amsterdam school-allocation problem becomes a one-to-one matching problem: students (individuals) must be assigned to school seats (individuals). A formal understanding of this different framing would be to translate the set of high schools $\mathcal{H}$ to a set of school seats $\mathcal{H}' = \{p_1^{h_1}, p_2^{h_1}, \ldots, p_{c_{h_1}}^{h_1}, p_1^{h_2}, \ldots, p_{c_{h_2}}^{h_2}, \ldots, p_{c_{h_m}}^{h_m}\}$, s.t. for each high school $h \in \mathcal{H}$ there are c_h distinct school seats p_j^h, with $1 \leq j \leq c_h$ in $\mathcal{H}'$. The many-to-one matching problems that can be framed as one-to-one matching problems, have the advantage that all formal results on one-to-one matching mechanisms carry over to the problem space of those many-to-one matching problems.

Just as many-to-one problems can sometimes be framed as one-to-one problems, so can one-sided problems sometimes be framed as two-sided problems. By assigning random artificial preferences to the side which originally had no preferences, both sides have preferences and two-sided matching mechanisms can be used. The consequence of such translations are not always as innocent as the translations from may-to-one to one-to-one. For instance, it is unclear which solution concept should be chosen, stability or Pareto optimality for one of the two parties. Furthermore, these artificial preferences can have severe real life implications and it is arguable if this is always admissible. In the Amsterdam school-allocation problem, there is however also a more natural two-sided framing of the problem possible, by understanding priorities as school preferences (cf. Section 3.4). Since in the following priorities will not be understood as school preferences, this translation is not further discussed here.

[1] Manlove (2013): pp. 5-6; Abdulkadiroğlu et al. (2009): p. 1955.

In the above the term *mechanism* was used to describe a systematic procedure that selects an allocation for a school market.[2] If a mechanisms selects the allocations based on a revealed preference profile of the students, then mechanisms are called a *direct mechanisms*. [3] Since all the mechanisms that will be discussed in the following require students to reveal their preferences over schools, no distinction will be made between direct and indirect mechanisms.

Definition 4.1 (Mechanism) *Let $\mathcal{M}$ be some school market. Let $\mathcal{A}$ be the set of allocations possible for $\mathcal{M}$. If $\mathfrak{M}$ is a function mapping school market $\mathcal{M}$ to some allocation $A \in \mathcal{A}$, then $\mathfrak{M}$ is a mechanism.*

In the following section some widely used matching mechanisms will be discussed. First two one-sided school-allocation mechanisms will be discussed, the *Permutation Allocation Mechanism* and the *Boston Allocation Mechanism*. Thereafter, two different versions of the two-sided allocation mechanism with *deferred acceptance* will be discussed. The two-sided allocation mechanisms treated in the following still satisfy the definition above and only need a *school market* as input, since these mechanisms will artificially create preferences for the high schools.

4.1.1 Permutation Allocation Mechanism

The first school-allocation mechanism that will be discussed is the *Permutation Allocation Mechanism (PAM)*. This mechanism is a queue-order mechanism, in which individuals are assigned schools from a set of remaining still available schools, when students with higher queue rankings are already assigned there most preferred still available school.[4] Queue-order mechanisms, as the PAM are also called *(random) serial dictatorships* and are sometimes named *lottery mechanisms* if the order of the queue is determined randomly.[5] Serial dictatorships are widely used, for example to allocate dormitory rooms (or on-campus housing facilities) to students[6] and to assign Navel academy graduates to ships.[7] The following set of instructions in pseudo code outline the *Permutation Allocation Mechanism*.

Definition 4.2 (Permutation Allocation Mechanism (PAM)) *All (probabilistic) algorithms that are extensionally equivalent to the following algorithm are called the* Permutation Allocation Mechanism *(PAM):*

input : *School Market* $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{P}, \mathcal{C})$.
output: *Allocation* $A : \mathcal{S} \to \mathcal{H}$.

Generate a random permutation $\overrightarrow{S} \underset{r}{\leftarrow} \Pi_{\mathcal{S}}$;

for $0 \leq i \leq |\overrightarrow{S}| - 1$ **do**

  Assign $\overrightarrow{S}(i)$ *to the most preferred school of* $\overrightarrow{S}(i)$*, according to* $P_{\overrightarrow{S}(i)}$ *that still has seats available;*

return *the set of all* $(s, A(s))$ *with* $s \in \mathcal{S}$.

Notation: both the allocation A produced by PAM, and the process of producing A by PAM, on the school market $(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$ by permutation $\overrightarrow{S}$ will be denoted by $PAM_{\overrightarrow{S}}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$. The set of allocations that can be produced by the PAM on some school market $\mathcal{M}$ will be denoted by $PAM_{\mathcal{M}}$.

The PAM simply generates a random order of students and then consequently assigns each student in that order to the most preferred school that is still available. If 'assign' is mentioned in the stipulation of a school-allocation mechanism, then this is where the allocation, viewed as a function, is generated; that is, for every 'assignment' a pair *(student, high school)* is added to the allocation. Each iteration of the 'for loop' in

[2] A mechanism is sometimes also called a *social choice function*, cf. Svensson (1999) p. 559.
[3] Abdulkadiroğlu & Sönmez (2003): p. 733.
[4] Svensson (1999) p. 558.
[5] Abdulkadiroğlu & Sönmez (1998): p. 689.
[6] Abdulkadiroğlu & Sönmez (2003): p. 731.
[7] Roth & Sotomayor (1990): p. 90.

the above description of the PAM is where a student is assigned the best available school. Each such iteration will be called a *stage* of the PAM. The *stage* in which some student s is assigned to some school by PAM can be denoted by $\overrightarrow{S}^{-1}(s)$, that is, the index of s in the permutation used to allocate students. The following example briefly demonstrates the workings of the mechanism:

Example 4.3 *Let $\mathcal{S} = \{1,2,3\}$ and let $\mathcal{H} = \{a,b,c\}$, such that all high schools have capacity 1. Say we have the following preference lists:*

- $P_1 = \langle c, a, b \rangle$;
- $P_2 = \langle a, b, c \rangle$;
- $P_3 = \langle a, b, c \rangle$;

Let the above described school market be the input to the PAM. The first thing the PAM does is the generate a random permutation $\overrightarrow{S}$ of the set of students $\mathcal{S}$. Say $\overrightarrow{S} = \langle 2,3,1 \rangle$. Then in stage 0, the PAM assigns student $\overrightarrow{S}(0) = 2$ to its most preferred school that still has seats available. Since all schools still have seats available, this is a, hence $(2,a)$ is added to the allocation. In the next stage, the PAM assigns student $\overrightarrow{S}(1) = 3$ to 3's most preferred school that still has seats available, which is b, hence $(3,b)$ is added to the allocation. In the final stage, student $\overrightarrow{S}(2) = 1$ is assigned to school c, since c still has seats available. Hence, the PAM generate the allocation $A = \{(1,c),(2,a),(3,b)\}$.

The allocations produced by mechanisms can have certain formal properties (cf. section 3.5). If all the allocations produced by some mechanism, given some school market, have certain properties, the mechanism itself can also be said to have certain corresponding formal properties. From the introductory discussion on school-allocation mechanisms it was claimed that matching mechanisms for one-sided matching problems that produce Pareto-optimal allocation are of special interest. If some matching mechanism only creates Pareto-optimal allocations, then it said that the mechanism is *Pareto consistent*:

Definition 4.4 (Pareto-consistent Mechanism) *Let $\mathcal{M}$ be some school market and let $\mathfrak{M}$ be some mechanism. Say $\mathfrak{M}_{\mathcal{M}}$ is the set of allocations that $\mathfrak{M}$ can produce for the school market $\mathcal{M}$. Then $\mathfrak{M}$ is Pareto consistent if and only if $\mathfrak{M}_{\mathcal{M}} \subseteq PAROPT_{\mathcal{M}}$.*

The allocations produced by the *Permutation Allocation Mechanism* (or *Random Serial Dictatorship*) are of special interest, since all of them are Pareto optimal. That is, the PAM is a Pareto-consistent mechanism. Furthermore, all Pareto-optimal allocations possible for some given school market $\mathcal{M}$ are exactly the possible outcomes for the PAM on that school market $\mathcal{M}$.

Theorem 4.5 (PAM = PAROPT) *Let $\mathcal{M} = (\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P})$ be some school market, with $|\mathcal{H}| = m$. Let $PAM_{\mathcal{M}}$ be the set of all allocations that can be produced by PAM on $\mathcal{M}$ and let $PAROPT_{\mathcal{M}}$ be the set of all Pareto-optimal allocations for $\mathcal{M}$. Then $PAM_{\mathcal{M}} = PAROPT_{\mathcal{M}}$.*

Proof. First the inclusion $PAM_{\mathcal{M}} \subseteq PAROPT_{\mathcal{M}}$ will be proved, then the inclusion $PAM_{\mathcal{M}} \supseteq PAROPT_{\mathcal{M}}$.

$\subseteq$: Let A be some allocation produced by $PAM_{\overrightarrow{S}}(\mathcal{M})$. Assume for reductio that A is not a Pareto-optimal allocation, then there exists some allocation A' s.t.

- $\exists s \in \mathcal{S} : A(s) <_s A'(s)$, and
- $\forall s \in \mathcal{S} : A(s) \leq_s A'(s)$.

Let s be the first student in $\overrightarrow{S}$ that is strictly better off in A', that is $\forall s' \in \mathcal{S}(S^{-1}(s') < S^{-1}(s) \Rightarrow A(s') \leq_{s'} A'(s'))$. Say $A(s) = h$ and $A'(s) = h'$, with $h \neq h'$. Since $PAM_{\overrightarrow{S}}(\mathcal{M})$ did not assign h' to s, it must have been that h' did not have seats left at stage $S^{-1}(s)$ of $PAM_{\overrightarrow{S}}(\mathcal{M})$. The latter implies that there are $c_{h'}$ students assigned to h' in $PAM_{\overrightarrow{S}}(\mathcal{M})$, before stage $S^{-1}(s)$. Since s is the first student who is better off under A' and no student is worse off in A', it must be that for all

students s' with a lower index than s in $\overrightarrow{S}$: $A(s') = A'(s')$. Hence, if there are $c_{h'}$ students, with a lower index than s, assigned to h' under A, then also $c_{h'}$ students, with a lower index than s are assigned to h' under A'. Since $A'(s) = h'$ as well, there are more than $c_{h'}$ students assigned to h' under A'; by Definition 3.1 we then have that A' is *not* an allocation: contradiction. This means that there cannot be such an allocation A', hence A is a Pareto-optimal allocation.

$\supseteq$: Let A be some Pareto-optimal allocation for the school market $\mathcal{M}$, with $|\mathcal{H}| = m$. Let $U_i = \{s \in \mathcal{S} \mid A(s) = P_s(i)\}$, that is U_i is the set of students who have preference position $i+1$ under A. Then there are sets $U_0, U_1, ..., U_{m-1}$, s.t. $\bigcup_{i\geq 0}^{m-1} U_i = \mathcal{S}$, where some U_i are possibly empty. Say $\overrightarrow{U_i}$ is some list representation of U_i and let

$$\overrightarrow{S} := \overrightarrow{U_0} + +\overrightarrow{U_1} + +... + +\overrightarrow{U_{m-1}}.$$

Say $PAM_{\overrightarrow{S}}(\mathcal{M}) = A'$. It will now be shown by induction on i, that $\forall s \in \overrightarrow{U_i} : A(s) = A'(s)$.

- (Base $i := 0$): Say $\overrightarrow{U_0}(0) = s$, then $\overrightarrow{S}(0) = s$. Since s is the first student in the permutation $\overrightarrow{S}$ it will be assigned its first choice under $PAM_{\overrightarrow{S}}(\mathcal{M})$, hence $A(s) = A'(s)$. Now assume that for all predecessors s'' of some $s' \in \overrightarrow{U_0}$: $A(s'') = A'(s'')$. Now suppose for reductio, $A(s') \neq A'(s')$. Then it must be that at stage $\overrightarrow{S}^{-1}(s')$ of $PAM_{\overrightarrow{S}}(\mathcal{M})$ school $A(s')$ has no seats left. Since for all the predecessors s'' of s', $A(s'') = A'(s'')$, it must then be that school $A(s')$ is assigned more students than its capacity, making A not an allocation: contradiction. Hence, also for s' it must be that $A(s') = A'(s')$.
- (Induction Step $i := k$): Assume as the induction hypothesis (IH) that for all $\overrightarrow{U_j}$ with $j < k$, that $\forall s \in \overrightarrow{U_j}$: $A(s) = A'(s)$. Assume that for all predecessors s' of some $s' \in \overrightarrow{U_k}$: $A(s') = A'(s')$. Now suppose for reductio $A(s) \neq A'(s)$. Then there are two cases to consider, either $A(s) >_s A'(s)$ or $A(s) <_s A'(s)$ (since preferences are strict). In the first case, when s is assigned a better school under A, it must be that school $A(s)$ has no seats left at stage $\overrightarrow{S}^{-1}(s)$ of $PAM_{\overrightarrow{S}}(\mathcal{M})$. But since by IH for all predecessors s' of s in $\overrightarrow{S}$, $A(s') = A'(s')$, it must then be that school $A(s)$ is assigned more students then its capacity, making A not an allocation: contradiction. In the second case, when s is assigned a worse school under A, it must be that at stage $\overrightarrow{S}^{-1}(s)$ of $PAM_{\overrightarrow{S}}(\mathcal{M})$ a better school still has seats left. Since A is a Pareto-optimal allocation, A must be saturated (cf. Proposition 3.16), hence it cannot be that school $A'(s)$ has seats available in allocation A. All predecessors of s are assigned the same schools under A and A', thus it must be that there is some successor s' of s in $\overrightarrow{S}$ s.t. $A(s') = A'(s)$ and $A(s') \neq A'(s')$. This student s' is either better off under A' or worse off; if s' is better off, then both s and s' are better off, without any student necessarily being worse off, making A not Pareto optimal: contradiction; if s' is worse off, then PAM does not necessarily produce Pareto-optimal allocations, since A would be a 'Pareto improvement' of A', but it was shown directly above that $PAM \subseteq PAROPT$: contradiction. Hence, we must have $A(s) = A'(s)$.

From the inductive argument it can be concluded that to every Pareto-optimal allocation A for some school market $\mathcal{M}$ there corresponds some permutation Π on $\mathcal{S}$ s.t. $PAM_{\Pi}(\mathcal{M})$ produces A. Hence every Pareto-optimal allocation for $\mathcal{M}$ is in the set $PAM_{\mathcal{M}}$.

□

Corollary 4.6 *From Theorem 4.5 and Proposition 3.16 it follows that the all allocations for some school market $\mathcal{M}$ produced by the PAM, given $\mathcal{M}$, are pair-wise Pareto efficient and saturated.*

The above theorem demonstrates that all Pareto-optimal allocations for some school market $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{C}, \mathcal{P})$ can be found by performing the PAM with all possible permutations in $\Pi_{\mathcal{S}}$. Moreover, Theorem 4.5 shows that to every permutation there is associated a Pareto-optimal allocation. Note however that there can be more permutations that generate the same allocation, when using the PAM:

Example 4.7 *Consider the following Let* $\mathcal{S} = \{1, 2, 3\}$ *and let* $\mathcal{H} = \{a, b, c\}$*, s.t. all schools have capacity 1. Say that the students have the following preference lists:*

- $P_1 = \langle a, b, c \rangle$*;*
- $P_2 = \langle b, a, c \rangle$*;*
- $P_3 = \langle a, b, c \rangle$*;*

Now consider the following to permutation $\overrightarrow{S} = \langle 1, 2, 3 \rangle$ *and* $\overrightarrow{S}' = \langle 2, 1, 3 \rangle$*. Clearly* $PAM_{\overrightarrow{S}}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P}) = PAM_{\overrightarrow{S}'}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P}) = \{(1, a), (2, b), (3, c)\}$*, since it does not matter if student* 1 *or* 2 *is assigned the best available school first, because the first choice of* 1 *and* 2 *differs.*

Theorem 4.5 establishes that there cannot be more Pareto-optimal allocations for some given school market $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{C}, \mathcal{P})$ than $|\mathcal{S}|!$ (i.e. the factorial of the number of students, which are the number of different permutations of $\mathcal{S}$). Furthermore, Example 4.7 shows that is possible that multiple permutations corresponds to one and the same allocations, hence we must have:

Corollary 4.8 *For a given school market* $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{C}, \mathcal{P})$*, there is a* surjective *function mapping the set of permutations of the set of students* $\Pi_{\mathcal{S}}$ *to the set of Pareto-optimal allocations for that school market* $PAROPT_{\mathcal{M}}$*, namely the function* $PAM_{\overrightarrow{S}}(\mathcal{M})$ *for* $\overrightarrow{S} \in \Pi_{\mathcal{S}}$*.*

Theorem 4.5 will be very important in the design of the genetic school-allocation algorithm introduced in the next chapter, since it shows that if optimisation on the set of Pareto-optimal allocations needs to be performed, this can also be performed on the set of permutation of the students, whereby for each permutation the associated allocation is produced by the PAM.

4.1.2 Boston Allocation Mechanism

The second mechanism that will be discussed is the *Boston Allocation Mechanism (BAM)*. The mechanism resembles the system in use in Amsterdam before the introduction of the deferred acceptance school-allocation mechanism, and can be described as multiple rounds of synchronized lotteries. In short, the BAM performs multiple rounds of lotteries per school to decide which students to assign to that school, such that in round i, lotteries decide if students not allocated yet, will be assigned to their i-th choice. The major difference between the BAM and the system of synchronized lotteries that was in use in Amsterdam, is that in the BAM students hand in preference lists once in advance, whereas in the old Amsterdam system, students handed in their i-th choice at the i-th round. This difference allows students to 'strategise' differently in the old Amsterdam mechanism than in the BAM. In this thesis the BAM will be investigated and not the old Amsterdam system, because the BAM is a more evolved school-allocation mechanism, not peculiar to Amsterdam alone, that was used in more school districts and for other allocation problems.[8] The following set of instructions in pseudo code outline the *Boston Allocation Mechanism*:

(Definition on next page)

[8]The BAM and similar systems were in use in the Hillsborough County School District in Tampa-St. Petersburg, and in Cambridge (MA), Denver, Minneapolis and Seatle; the mechanism was also used for matching medical graduates to internships in several regions of the United Kingdom (cf. Abdulkadiroğlu et al. (2005): p. 369).

Definition 4.9 (Boston Allocation Mechanism (BAM)) *All (probabilistic) algorithms that are extensionally equivalent to the following algorithm are called the* Boston Allocation Mechanism *(BAM):*

input : *School Market* $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{P}, \mathcal{C})$.
output: *Allocation* $A : \mathcal{S} \to \mathcal{H}$.

Let $\rho := -1$;
Let $N_\rho := \mathcal{S}$;
while $N_\rho \neq \varnothing$ **do**
  Let $\rho := \rho + 1$;
  Let $N_\rho := N_{\rho-1}$;
  for $h \in \mathcal{H}$ **do**
    Let $\overrightarrow{N}^h_\rho$ *be some random permutation of* $N^h_\rho = \{s \in N_\rho \mid P_s(\rho) = h\}$;
    Assign the first c_h *students of* $\overrightarrow{N}^h_\rho$ *to* h, *let these students be* W^h_ρ;
    Let $N_\rho := N_\rho \setminus W^h_\rho$;
    Let $c_h := c_h - |W^h_\rho|$;

return *the set of all* $(s, A(s))$ *with* $s \in \mathcal{S}$.

Notation: both the allocation A *produced by* BAM, *and the process of producing* A *by* BAM, *on the school market* $(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$ *by permutation* $\overrightarrow{S}$ *will be denoted by* $PAM_{\overrightarrow{S}}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$. *The set of allocations that can be produced by the PAM on some school market* $\mathcal{M}$ *will be denoted by* $BAM_{\mathcal{M}}$.

Each iteration of the while loop in the BAM is called a *round*. The *round* in which ρ has value i is called the $(i+1)$-th round (or round $i+1$), referring to the position on the preference list of the students to which the round corresponds. The set N_ρ is the set of students who are not assigned in the rounds $\leq \rho$. The set W^h_ρ are the students who are assigned to school h in round $(\rho+1)$, that is, the students who won the lottery for school h, which was their $(\rho+1)$-th choice, in round $(\rho+1)$. Moreover, note that at each round i, the N^h_{i-1} are disjoint (and thus the $W^h_{(i-1)}$ as well), because students can only place one school on the i-th place in their preference list. Furthermore, if a *null school* h_0 is introduced to the set of high schools, such that it has enough seats for all students, i.e. $c_{h_0} = |\mathcal{S}|$, and place this school last on every student's preference list, i.e. $\forall s \in \mathcal{S} : P_s := P_s + +\langle h_0 \rangle$, then the BAM will always terminate. The following example briefly demonstrates the workings of the mechanism:

Example 4.10 *Let* $\mathcal{S} = \{1, 2, 3, 4\}$ *and let* $\mathcal{H} = \{a, b, c\}$, *s.t. school* c *has capacity 2 and all other schools have capacity 1. Say the students have the following preference lists:*

- $P_1 = \langle a, b, c \rangle$;
- $P_2 = \langle b, a, c \rangle$;
- $P_3 = \langle b, a, c \rangle$;
- $P_4 = \langle b, a, c \rangle$.

In the first round of the BAM, the sets N^h_ρ *are then:*

- $N^a_0 = \{s \in N_0 \mid P_s = a\} = \{1\}$;
- $N^a_0 = \{s \in N_0 \mid P_s = b\} = \{2, 3, 4\}$.

Now say the lotteries (permutations) have the following outcomes:

- $\overrightarrow{N^a_0} = \langle 1 \rangle$;
- $\overrightarrow{N^b_0} = \langle 2, 4, 3 \rangle$.

Then student 1 is assigned to school a *and student 2 is assigned to* b. *After these assignments then* $N_0 = \{3, 4\}$. *Now observe that students 3 and 4 both have school* a *as their second choice. Since school* a *does not have any seats left, there will be no students assigned from* $\overrightarrow{N^a_1}$ *in the second round. The latter means that no students are assigned to schools in round 2 and thus* $N_1 = \{3, 4\} = N_2$. *In the third round, students 3 and 4 both have school* c *as their third choice. Since school* c *has 2 seats left and for every*

$\overrightarrow{N_2^c} \underset{r}{\leftarrow} N_2^c$ *students 3 and 4 will be assigned to school c, students 3 and 4 will be assigned to school c in round 3. After these assignments,* N_2 *is set to* $N_2 \setminus W_2^c = \{3,4\} \setminus \{3,4\} = \varnothing$ *and there cannot be a next iteration of the while loop, for* $N_2 = \varnothing$*. This means the BAM returns* $A = \{(1,a),(2,b),(3,c),(4,c)\}$.

From the example it becomes also clear, that an algorithm for performing the BAM, in which one single permutation is used per round, to perform the lotteries is equivalent to the algorithm outlined in Definition 4.9. That is, in stead of generating all the permutations $\overrightarrow{N_\rho^h}$, also one permutation on N_ρ can be generated, such that for each school h the first c_h students (where c_h is the 'updated' capacity), for which $P_s(\rho) = h$, are assigned to h, since all N_ρ^h are disjoint.

The BAM is often used in combination with an expanded system of priorities. For instance, in Boston, when the system was still used, first a younger sibling had priority to attend the same school as an older sibling, next in priority for half of each program's seats were students from the school's 'walk zone', etc.[9] These priorities then influenced the lotteries, that is children with priorities would be ordered in their priority ordering before all non priority students, in the permutation for the specific school in the specific round. Since in the Amsterdam school-allocation problem, how its formalised here, priorities are not taken into account, in the following formal results on the BAM, it will be assumed that there is no such system of priorities. Furthermore, the BAM can also be used without lotteries, if all high schools have elaborate preference lists over students; these preference lists are then used as the lottery outcomes.

The PAM and BAM are different school-allocation mechanisms for some given school market $\mathcal{M}$ and are not extensionally equivalent algorithms.[10] The next examples demonstrates this by giving some allocation on some school market $\mathcal{M}$ which is a possible outcome of the PAM, but can never be produced by the BAM:

Example 4.11 *Let* $\mathcal{S} = \{1,2,3\}$ *and let* $\mathcal{H} = \{a,b,c\}$*, s.t. all schools have capacity 1. Say that the students have the following preference lists:*

- $P_1 = \langle b,a,c\rangle$*;*
- $P_2 = \langle b,a,c\rangle$*;*
- $P_3 = \langle a,c,b\rangle$*;*

Let the above described school market be the input of the PAM with permutation $\overrightarrow{S} = \langle 2,1,3\rangle$*. Then* $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P}) = \{(1,a),(2,b),(3,c)\}$*. Now observe that the BAM, given the same school market* $(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P})$ *would* never *produce this allocation, since the BAM will always assign student 3 to school a in the first round. Furthermore, the BAM will assign either student 1 or 2 to school b in the first round, such that in the third round, then the by lottery eliminated student of the former round will be assigned school c. This means that the only allocations that the BAM can produce for this school market are* $\{(1,b),(2,c),(3,a)\}$ *and* $\{(1,c),(2,b),(3,a)\}$*. The BAM can thus never produce the allocation* $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P}) = \{(1,a),(2,b),(3,c)\}$.

The example above is a counter example for the claim that the BAM and the PAM are extensionally equivalent. The proof of the next theorem demonstrates, however, that every allocation that the BAM produces on some school market $\mathcal{M}$ can also be the result of the PAM on that same market. These two results combined give that $BAM_{\mathcal{M}} \subsetneqq PAM_{\mathcal{M}}$.

Theorem 4.12 $(BAM \subsetneqq PAM)$ *Let* $\mathcal{M} = (\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P})$ *be some school market. Then* $PAM_{\mathcal{M}} \subsetneqq BAM_{\mathcal{M}}$.

Proof. First observe that Example 4.11 demonstrates that $PAM \nsubseteq BAM$, hence it suffices to show that every allocations produced by the BAM for some school market $\mathcal{M}$ can be produced by the PAM, to

[9] Abdulkadiroğlu et al. (2005): p. 368.

[10] Two algorithms φ and ψ are extensionally equivalent, if for the same input φ and ψ always produce the same output. Note, that it is possible for two algorithms φ and ψ to compute the output differently, if so then φ and ψ are intensionally *not* equivalent.

demonstrate that $BAM_{\mathcal{M}} \subsetneq PAM_{\mathcal{M}}$. Let $\mathcal{M} = (\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$ be some school market, s.t. $|\mathcal{S}| = n$ and $|\mathcal{H}| = m$. Let $BAM_{\mathcal{W}}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P}) = A$, where $\mathcal{W} = \{W_0^{h_1}, W_0^{h_2}, ..., W_0^{h_m}, W_1^{h_1}, ..., W_l^{h_m}\}$, that is, $\mathcal{W}$ is the set of all the lottery winning students for each school of each round, where we assume that the BAM needed $l+1$ rounds. Now say:

$$\begin{aligned}\vec{S} = &\left(W_0^{h_1} + + W_0^{h_2} + + ... + + W_0^{h_m}\right) + + \\ &\left(W_1^{h_1} + + W_1^{h_2} + + ... + + W_1^{h_m}\right) + + \\ &... \\ &\left(W_l^{h_1} + + W_l^{h_2} + + ... + + W_l^{h_m}\right)\end{aligned}$$

Note that W_i^h are the students assigned to school h in round $i+1$. Now let $PAM_{\overrightarrow{S}}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P}) = A'$. The following shows by induction on i, that for some arbitrary school h, for all students $s \in W_i^h$: $A(s) = A'(s)$.

- Base ($i := 0$): Let h be some arbitrary school. For all $s \in W_0^h$, school $A(s)$ is their first choice and the BAM allocates them to their first choice. Since $|W_0^h| \leq c_h$, also all $s \in W_0^h$ will be assigned to $A(s)$ by $PAM_{\overrightarrow{S}}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$, because (i) h must have the capacity to place them, (ii) h is their most preferred school, and (iii) the only students placed at h for students s' in $W_0^{h_1} + + ... + + W_0^h + + ... + + W_0^{h_m}$ are the s in W_0^h, because the $W_0^{h_j}$ are disjoint.
- Induction Step ($i = k$). Assume as induction hypothesis (IH) that for all $W_u^{h_j}$ with $u < k$ that $\forall s \in W_u^{h_j} : A(s) = A'(s)$. Let h be some arbitrary school for which W_k^h is non-empty (the empty case is trivial). Let s be some arbitrary student in W_k^h and suppose for reductio that s is not assigned to h by $PAM_{\overrightarrow{S}}(\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$, that is $A(s) \neq A'(s)$. Then there are two cases to consider:

  Case 1: *School h is full.* Note that the only students allocated to h in previous stages are, by the IH, $\bigcup_{u<k} W_u^h$, hence h being full implies $\sum_{u<k} |W_u^h| = c_h$. This means that $W_k^h = \varnothing$ by the structure of the BAM: contradiction.

  Case 2: *Student s prefers $A'(s)$ over h and $A'(s)$ had seats left.* If $A'(s)$ is preferred over h, then there exists $u < k$ s.t. $s \in N_u^{A'(S)} = \{s \in N_u \mid P_s(u) = A'(s)\}$. Since we have $s \in W_u^h$, it must be that for all $u < k$ that $s \notin W_u^{A'(S)}$ (that is, student s must have been eliminated by lottery in all the previous rounds). But if $A'(s)$ is not full and by IH the only students assigned to $A'(s)$ are $\bigcup_{u<k} W_u^{A'(s)}$, then it must be the case that $\sum_{u<k} |W_u^{A'(s)}| < c_{A'(s)}$. But then also $\sum_{u<k} |N_u^{A'(s)}| < c_{A'(s)}$ and thus there exists $u \leq k$ s.t. $s \in W_u^{A'(S)}$ by the structure of the BAM: contradiction.

  Both cases give a contradiction, hence it must be that for arbitrary s in W_k^h: $A(s) = A'(s)$.

Since for arbitrary h, for all i, for all $s \in W_i^h$: $A(s) = A'(s)$, it must be that $A = A'$. Hence, the allocation A could also have been the result of the PAM with permutation $\overrightarrow{S}$. □

The BAM is also a Pareto-consistent mechanisms, since all allocations produced by the BAM, for some given school market $\mathcal{M}$, can also be produced by the PAM and, as established in the previous section, the PAM only produces Pareto-optimal for some given school market $\mathcal{M}$.

Corollary 4.13 *By Theorem 4.12 in conjunction with Theorem 4.5, the Boston Allocation Mechanism is Pareto consistent.*

Corollary 4.14 *From Corollary 4.13 and Proposition 3.16 it follows that the all allocations for some school market $\mathcal{M}$ produced by the BAM, given $\mathcal{M}$, are pair-wise Pareto efficient and saturated.*

From Theorem 4.12, which gives that $BAM_{\mathcal{M}}$ is a *strict* subset of $PAM_{\mathcal{M}}$, it is clear that the BAM cannot produce all Pareto-optimal allocations for some given school market. This also implies that the BAM cannot be exploited to optimise over the set of Pareto-optimal allocations for some school market.

4.1.3 Deferred Acceptance School-Allocation Mechanisms

In this section two variants of the *Deferred Acceptance (DA) School-Allocation Mechanism* will be discussed. DA algorithms are matching mechanisms for two-sided bipartite matching problems. The most famous example of a two-sided matching problem is the *stable marriage problem*, formulated by Gale & Shapley (1962). The problem is to match men and women according to their preferences over the other sex, as to create *stable* marriages. Intuitively, a matching is stable if, in no two couples, both agents prefer one another over their current assignees. If such pairs do exist, then they could undermine the matching produced by the matching mechanism, by forming a private arrangement outside of the matching. A matching mechanism for two-sided preferences should therefore always aim to produce *stable* matchings. There is many empirical and theoretical evidence supporting the solution concept of *stability* for two-sided matching problems.[11]

The stable matching algorithm, described by Gale and Shapley in 1962 can be easily adjusted to be a matching mechanism for specific school markets, where not only the students have preferences over the high schools, but the high schools have preferences over the students as well. These preferences could either be actual preferences (e.g. based on test results by students) or an implementation of priorities as preferences. The *deferred acceptance* school-allocation mechanism so obtained is the following (N.B. $l > k$):

0. Students and schools hand in their preference lists;

k. Until the set of unmatched students is empty, each student is tentatively matched to their first choice. Each school h accepts the c_h highest ranked students, according to h's preferences and are now tentatively matched to h. The not accepted students are set to be unmatched and their preference list is updated to the tail (cf. Definition 3.4) of their preference list.

l. The mechanism produces an allocation according to the final obtained matching.

This procedure is called a *deferred acceptance* matching mechanism, because at each step the mechanism only tentatively assigns students to schools, but does not allocate them directly to that school, that is, the final decision to allocate students to some school is *deferred* to the stage where no more students are unmatched.

Similarly as for the BAM, if a *null school* h_0 is introduced to the set of high schools, such that it has enough seats for all students, i.e. $c_{h_0} = |\mathcal{S}|$, and place this school last on every student's preference list, i.e. $\forall s \in \mathcal{S} : P_s := P_s + +\overrightarrow{\{h_0\}}$, then the algorithm will always terminate. To see this, observe that the highest ranked c_{h_0} students, according to c_{h_0} preferences, are always all students in $\mathcal{S}$. Hence, at some point when some student s is not accepted by any of its real preferred schools, then it will be matched to h_0 by the mechanism, from which it can never be removed. Note that the preferences of the null school are not important, since it has a high enough capacity to accommodate all students. The interpretation of some student being assigned to the null school, is to say that the student cannot be allocated. Another possibility to forcing the algorithm to terminate is by making all the student's preference lists complete, that is augmenting the preference list of each student by placing all schools not occurring in its preference list, after the schools on its preference list in some (strict) order.

From the above description of the deferred acceptance school-allocation mechanism it is clear that the mechanism needs high-school preferences over students to operate. Given these preferences, the mechanism produces *stable* allocations, that is, there will be no two pairs (s, h) and (s', h') in the allocation such that (i) s prefers to be matched with h', (ii) s' prefers to be matched with h, (iii) h prefers to be matched with s', and (iv) h' prefers to be matched with s. It will not be demonstrated formally that the deferred acceptance mechanism is *stable*, because this notion is not very interesting for the Amsterdam school-allocation problem.[12] In the literature on college-allocation problems, stability is often the central solution concept, because both parties (students and colleges) are allowed to have preferences over the other party.[13] A college admission problem is thus often a real two-sided matching problem; the Amsterdam school-allocation problem is however only a

[11] Emperical evidence: Kagel & Roth (2000): pp. 202, 204, in conjunction with Roth (1991) and Roth (1990); theoretical evidence: Roth & Sotomayor (1990).

[12] For more on stability for many-to-one two-sided allocation problems see Roth & Sotomayor (1990).

[13] Abdulkadiroğlu & Sönmez (2003): p. 731.

two-sided matching problem when school priorities are understood as high-school preferences over students. As argued earlier, this understanding of priorities is not chosen in this thesis, since there are not enough priorities nor a complicated enough system of priorities in Amsterdam, to obtain lengthy high-school preferences, needed for well working two-sided matching mechanisms.[14] If only one party has preferences, the property of stability is not important for some allocation to have.[15] In discussing properties of allocations, produced by deferred acceptance school-allocations mechanisms, for the Amsterdam case, it is therefore not necessary to concentrate on high-school welfare, that is, if the allocation is Pareto optimal from the point of view of the high schools.

From the foregoing it is clear that to apply deferred acceptance school-allocation mechanisms in the Amsterdam case, high-school preferences need to be generated. To formulate this idea, the concept of a *tie-breaking rule* is used. High schools in Amsterdam can be formally seen to be indifferent on all students, that is, for all $h \in \mathcal{H}$, for all $s, s' \in \mathcal{S}, s \leq_h s'$ and $s' \leq_h s$. in the adopted formal understanding of the Amsterdam school-allocation problem indifferent preferences are not allowed. To *break* this indifference a *tie-breaking rule* is needed which decides which student the school prefers more, if a school is indifferent between two students. The *tie-breaking rule* by breaking indifferences creates strict preference orderings over students and thus actual *preference lists* for high schools according to Definition 3.6. Two accepted ways for creating preference lists for high schools are, (i) generate a random permutation $\overrightarrow{S}$ of the set of students and set $\overrightarrow{S}$ as the preference list P_h for all high schools $h \in \mathcal{H}$, and (ii) for every school $h \in \mathcal{H}$ generate a random permutation $\overrightarrow{S}_h$ and let $P_h := \overrightarrow{S}_h$. Option (i) is called the *single tie-breaking rule* and option (ii) is called a *multiple tie-breaking rule*.

Definition 4.15 (Tie-Breaking Rule) *Let $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{P}, \mathcal{C})$ be some school market. Let $\theta(\mathcal{M})$ be some (probabilistic) algorithm for generating a* preference list *P_h for all $h \in \mathcal{H}$ over $\mathcal{S}$ (cf. Definition 3.6). Then θ is called a* tie-breaking rule. *If θ is a (probabilistic) algorithm s.t. for every run of θ given some school market $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{P}, \mathcal{C})$, for all $P_h, P_{h'} \in \theta(\mathcal{M}) : P_h = P_{h'}$, then θ is called the* single tie-breaking rule. *Any tie-breaking rule that is not the single tie-breaking rule is called a* multiple tie-breaking rule.

With a clear understanding of how to create high-school preferences and how this can be formally captured, deferred acceptance school-allocation mechanisms can now be formally defined as a school-allocation mechanism, with two inputs: a school market $\mathcal{M}$ and a tie-breaking rule θ for generating the high-school preferences:

(Definition on next page)

[14] If priorities are understood as high-school preferences the notion of stability is understood as: "there should be no unmatched student-school pair (i, h) where student i prefers school h to her assignment and she has higher priority than some other student who is assigned a seat at school h" (Abdulkadiroğlu et al. (2009): p. 1955). If an allocation is stable in this sense, it is sometimes said that there is no 'justified envy'.

[15] Manlove (2013): pp. 5-6.

Definition 4.16 (Deferred Acceptance Allocation Mechanism (DAAM)) *All (probabilistic) algorithms that are extensionally equivalent to the following algorithm are called the* Deferred Acceptance Allocation Mechanism (DAAM)*:*

input : *School Market* $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{P}, \mathcal{C})$*, tie-breaking rule* θ*.*
output: *Allocation* $A : \mathcal{S} \to \mathcal{H}$*.*

Let $H := \mathcal{H} \cup \{h_0\}$*;*
Let $C := \mathcal{C} \cup \{(h_0, |\mathcal{S}|)\}$*;*
Let $N := \mathcal{S}$*;*
Run $\theta(H, \mathcal{S}, \mathcal{P}, \mathcal{C})$ *to generate a preference profile* $\mathcal{P}_H = \{P_{h_0}, P_{h_1}, ...\}$ *for all* $h \in H$*;*
while $N \neq \varnothing$ **do**
  for $h \in H$ **do**
    Let $T_h = \{s \in \mathcal{S} \mid P_s(0)\}$ *;*
    Let $\overrightarrow{W_h} = \iota(c_h, \overrightarrow{M_h})$*, where* $\overrightarrow{M_h} = \{(i, s) \mid s \in T_h\}$ *(c.f. Definition 3.4 for* ι*) s.t.*
     $\forall (i,s)(j,s') \in \overrightarrow{M_h} : P_h^{-1}(s) < P_h^{-1}(s') \Rightarrow i < j$*;*
    Let $N := N \setminus M_h \cup L_h$*, where* $M_h = \{s \in \mathcal{S} \mid (i, s) \in \overrightarrow{M_h}\}$ *and* $L_h = \{s \in T_h \mid (i, s) \notin \overrightarrow{W_h}\}$ *;*
  for $n \in N$ **do**
    $P_n := \tau(P_n)$ *(c.f. Definition 3.4 for* τ*);*
for $h \in H$ **do**
  Assign all s *s.t.* $(i, s) \in \overrightarrow{W_h}$ *to* h*;*
return *the set of all* $(s, A(s))$ *with* $s \in \mathcal{S}$*.*

If θ *is the* single tie-breaking rule*, the obtained algorithm is called the* Deferred Acceptance Allocation Mechanism with Single Tie Break (DA-STB)*; if* θ *is a* multiple tie-breaking rule*, the obtained algorithm is called the* Deferred Acceptance Allocation Mechanism with Multiple Tie Break (DA-MTB)*. Notation: Any allocation produced by the DA-STB for some school market* $\mathcal{M}$ *will be denoted by* $DA(\mathcal{M}, STB)$*; any allocation produced by a DA-MTB by* $DA(\mathcal{M}, MTB)$*. Again both the process and the final allocations will be referred to as* $DA(\mathcal{M}, \theta)$*.*

As in the case of the BAM, each iteration of the while loop in the DAAM is called a *round*. Note that in each round a new list of 'winners' $\overrightarrow{W_h}$ for every school is generated, which are the students who are tentatively assigned to school h in that round. These 'winners' are not, as in the BAM, determined by some random permutation generated each round, but by the preferences of the high schools over the students, generated by a given tie-breaking rule θ. Furthermore, observe that the 'losers' L_h for each school, in some round, are all given a new preference list $\tau(P_l)$, which is just the tail (cf. Definition 3.4) of the old preference list P_l. The following example will briefly demonstrate how the DAAM works, both according to the intuitive description of the algorithm and the formal description:

Example 4.17 *Let* $\mathcal{S} = \{1, 2, 3\}$ *and let* $\mathcal{H} = \{a, b, c\}$*, such that all high schools have capacity 1. Say we have the following preference lists for students and high schools (generated by some tie-breaking rule* θ*):*

- $P_1 = \langle a, b, c\,(, h_0)\rangle$*;*
- $P_2 = \langle a, b, c\,(, h_0)\rangle$*;*
- $P_3 = \langle b, a\,(, h_0)\rangle$*.*
- $P_a = \langle 2, 1, 3\rangle$*;*
- $P_b = \langle 1, 2, 3\rangle$*;*
- $P_c = \langle 3, 2, 1\rangle$*.*

First a new school h_0 *is generated, which has capacity* $|\mathcal{S}| = 3$*. This school is added to each student's preference list. In round 1, both students 1 and 2 will be matched to school a, i.e. their first choice (Formally:* $T_a = \{1, 2\}$*). Only student 2 is accepted by school a, since its capacity is 1, so it can only accept 1 student, and a prefers student 2 over student 1 (Formally:* $T_a = \{1, 2\}$*,* $\overrightarrow{M_a} = \{(0, 2), (1, 1)\}$*,* $\overrightarrow{W_a} = \iota(1, \overrightarrow{M_a}) = \{(0, 2)\}$*, and* $N := \{1, 2, 3\} \setminus \{2\} \cup \{1\} = \{1, 2\}$*). Student 3 is matched to school b and*

accepted by b (Formally: $T_b = \{3\}$, $\overrightarrow{M_b} = \{(0,3)\}$, $\overrightarrow{W_b} = \iota(1, \overrightarrow{M_b}) = \{(0,3)\}$, and $N := \{1,3\} \setminus \{3\} \cup \varnothing = \{1\}$). At the end of round 1, only student 1 is then unmatched, hence his preference list P_1 will be updated to its tail, that is $\langle b, c, h_0 \rangle$ (Formally $P_1 := \tau(P_1)$). In round 2, the only unmatched student, student 1, is matched to his first choice, according to his new preferences, school b (Formally: $T_b = \{1,3\}$). School b has now two students matched to it, student 1 and student 3, which was matched to b in round 1. School b only accepts student 1, since it prefers student 1 over student 3 (Formally: $\overrightarrow{M_b} = \{(0,1),(1,3)\}$, $\overrightarrow{W_b} = \iota(1, \overrightarrow{M_b}) = \{(0,1)\}$, and $N := \{1\} \setminus \{1\} \cup \{3\} = \{3\}$). Now student 3 is the only unmatched student, hence $P_3 := \tau(P_3) = \langle a, h_0 \rangle$. In round 3, student 3 is matched to school a. Since school a prefers student 2 over student 1, student 3 is not accepted by a and thus remains unmatched (Formally: $T_a = \{2,3\}$, $\overrightarrow{M_a} = \{(0,2),(1,3)\}$, $\overrightarrow{W_a} = \iota(1, \overrightarrow{M_a}) = \{(0,2)\}$, and $N := \{3\} \setminus \{2\} \cup \{3\} = \{3\}$). Again student 3's preference list will be updated s.t. $P_3 := \tau(P_3) = \langle h_0 \rangle$. In round 4, student 3 is then matched and accepted by the null school, meaning that student 3 remains unmatched (Formally $T_{h_0} = \{3\}$, $\overrightarrow{M_{h_0}} = \{(0,3)\}$, $\overrightarrow{W_{h_0}} = \iota(1, \overrightarrow{M_{h_0}}) = \{(0,3)\}$, and $N := \{3\} \setminus \{3\} \cup \varnothing = \varnothing$). Finally, the allocation $\{(2,a),(1,b),(3,h_0)\}$ is obtained.

In discussing properties of allocations, produced by deferred acceptance school-allocations mechanisms, for the Amsterdam case, it is not necessary to concentrate on high-school welfare, but only to investigate the welfare of the students, since the high-school preferences are artificially created. Since deferred acceptance algorithms were not designed for one-sided matching problems, not all tie-breaking rules produce Pareto-optimal allocations (from the view of the students). Consider the following example:

Example 4.18 *Let $\mathcal{S} = \{1,2,3\}$ and let $\mathcal{H} = \{a,b,c\}$, such that all high schools have capacity 1. Say we have the following preference lists for students and high schools:*

- $P_1 = \langle a, b\ (, h_0) \rangle$;
- $P_2 = \langle a, b, c\ (, h_0) \rangle$;
- $P_3 = \langle b, a\ (, h_0) \rangle$.
- $P_a = \langle 3, 1, 2 \rangle$;
- $P_b = \langle 1, 2, 3 \rangle$;
- $P_c = \langle 1, 3, 2 \rangle$.

Consider the following 'table description' of the deferred acceptance allocation mechanism, given the above preferences:

	a	b	c	h_0
Round 1	*1,* ~~*2*~~	3	-	-
Round 2	*1*	2, ~~*3*~~	-	-
Round 3	~~*1*~~, 3	2	-	-
Round 4	3	1, ~~*2*~~	-	-
Round 5	*3*	*1*	*2*	-
Allocation	*3*	*1*	*2*	-

In the allocation $A = \{(1,b),(2,c),(3,a)\}$, students 1 and 3 want to switch, since 1 prefers a over b and 4 prefers b over a. This means that from the view of the students A is not pair-wise Pareto efficient and thus not Pareto optimal. From the perspective of the high schools, the switch between student 1 and 3, will not be beneficial, since school a prefers student 3 over 1, and school b prefers 1 over 3. Hence, if both parties have a say on the switch, supposing they base their decision on their preferences, then the switch will not take place. It is easy to see that there cannot be a switch between $(1,b)$ and $(2,c)$, and $(2,c)$ and $(3,a)$, either, hence, allocation A is stable.

The above example not only demonstrates that allocations produced by a deferred acceptance school-allocation mechanism are *not* necessarily Pareto optimal, it also shows that *stability* does not imply Pareto optimality (from the perspective of the students). In Section 3.5 it was argued that Pareto optimality is a property that school allocations for Amsterdam need to fulfil. Hence, from Example 4.18 it can be concluded that not any deferred acceptance allocation mechanism, with any tie-breaking rule θ, will do. Moreover, it is clear that it cannot be that the DA-MTB always produces Pareto-optimal allocations, since Example

4.18, could describe a process of running the DA-MTB. The question remains if there are tie-breaking rules θ for which all allocations produced by DAAM are Pareto optimal. From the next theorem it follows that all allocations produced by the DA-STB are Pareto optimal. Moreover, the theorem states that the DA-STB is extensionally equivalent with the PAM and, by Definition 4.2 it then follows, that the DA-STB is just the PAM.

Theorem 4.19 *(PAM = DASTB) Let $\mathcal{M} = (\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$ be a school market. Let $\overrightarrow{S}$ be some random permutation of $\mathcal{S}$. Let $PAM_{\overrightarrow{S}}(\mathcal{M}) = A$ and let $DA(\mathcal{M}, STB) = A'$, s.t. the identical preferences P_{h_i} of each high school h_i are $\overrightarrow{S}$. Then $A = A'$.*

Proof. It will be shown by induction that for all $i < |\mathcal{S}|$: $A(\overrightarrow{S}(i)) = A'(\overrightarrow{S}(i))$.

- Base ($i := 0$): Say $\overrightarrow{S}(0) = s$ and $A(\overrightarrow{S}(0)) = h$. Then it must be the case that $P_s(0) = h$. If $P_s(0) = h$, then the DA-STB tentatively matches s to h. Now s can only be removed from h by the DA-STB if it allocates c_h students s' to h, for which we have $\overrightarrow{S}^{-1}(s') < \overrightarrow{S}^{-1}(s)$. Since $\overrightarrow{S}(0) = s$ such students s' cannot exist, hence we must have $A'(\overrightarrow{S}(0)) = h$.
- Induction Step ($i := k + 1$): Assume as the induction hypothesis (IH) that for all $j \leq k$: $A(\overrightarrow{S}(j)) = A'(\overrightarrow{S}(j))$. Say $\overrightarrow{S}(k+1) = s'$ and $A(\overrightarrow{S}(k+1)) = h'$. Suppose for reductio that $A'(\overrightarrow{S}(k+1)) = h''$, where $h'' \neq h'$. We then have two cases to consider:
  - Case 1 ($h'' >_s h'$): If $A(\overrightarrow{S}(k+1) \neq h''$, but $A(\overrightarrow{S}(k+1)) = h'$ and $h'' >_s h'$, then it must have been the case that at stage $k+1$ of the PAM school h'' did not have seats left, otherwise s' have been allocated there, since s' prefers h'' over h'. This means that in earlier stages other students than s' have been allocated to h' by the PAM. By the IH we have for these other students that they are also allocated to h' in the allocation A' produced by the DA-STB. This means that we cannot have $A(\overrightarrow{S}(k+1)) = h'$, because then there are more students allocated to h' then it has seats available: contradiction.
  - Case 2 ($h'' <_s h'$): If $A'(\overrightarrow{S}(k+1) \neq h'$, but $A'(\overrightarrow{S}(k+1) = h''$, and $h' >_s h''$, then it must have been the case that there are $c_{h'}$ students s'' allocated to h' by the DA-STB for which we have $\overrightarrow{S}(s'') < \overrightarrow{S}(s')$. Now note that these students s'' are allocated in earlier stages to h' by the PAM, than s' is allocated. Since $A(\overrightarrow{S}(k+1)) = h'$ at stage $k+1$, school h' had seats left at stage $k+1$, which means that in earlier stages of the PAM there are not $c_{h'}$ students allocated to h': contradiction.

If we have that for all $i < |\mathcal{S}|$, $A(\overrightarrow{S}(i)) = A'(\overrightarrow{S}(i))$, then all students are allocated to the same high schools in A and A'. This gives us that if the permutation $\overrightarrow{S}$ of the PAM is the same as the identical preferences of the high schools in the DA-STB, then the two allocations produced by the respective mechanisms are identical. From this we can conclude that the set of allocations the PAM and the DA-STB can produce for the same students, high schools and student preferences, are identical. □

Corollary 4.20 *From Theorems 4.19 and 4.5 it follows that all allocations for some school market $\mathcal{M}$ produced by the DA-STB are Pareto optimal.*

In Theorem 4.19 it is necessary that all high schools really have identical preferences. If a single tie-breaking rule is used to 'break' indifferences on 'preference lists' of high schools, which are only partly strict, then not all preference lists of all high schools are necessarily identical, notwithstanding that one single permutation is used to break all ties. In the literature this scenario is sometimes also called deferred acceptance with single tie break. Note that this scenario cannot occur in the setting of this thesis, since it is assumed that high schools *do not have* preference lists. Hence, it is never necessary to 'break' indifferences of preference lists. In this thesis single tie break refers to the way the artificial high-school preferences are created. If they are created by generating a single random permutation which is then each high school's preference, then a DAAM carried out on this will be equivalent to the PAM carried out on the same school market, if the

permutation of the PAM is the random permutation used for the high-school preferences. Another school-allocation mechanism found in the literature, not discussed in this thesis, but also equivalent to the PAM is the *Top Trading Cycles Mechanism (TTC)* from random endowments. If the initial allocation, from which the trade cycles will be carried out by the mechanism, is a random allocation, then the set of allocations produced by the TTC is equivalent the set of allocations produced by the PAM on that school market.[16]

4.2 Strategic Questions

4.2.1 Strategy Proofness

Policy makers not only set certain conditions for the allocations that some school-allocation mechanism must produce, but also for the game-theoretic properties a school-allocation mechanism needs to satisfy. A school-allocation mechanism can be described as a game, in which students can play certain strategies (the preference lists they hand in) that in part determine the outcome of the game (the school to which they will be assigned). An important game-theoretic property for school-allocation mechanisms found in the literature is *strategy proofness*. A mechanism is strategy proof, if it is a dominant strategy for students to hand in their actual true preferences. This means that it should not be possible for students to hand in some preference list Q, which is different from their actual true preference list P such that the 'strategic' preference list Q allows them to be assigned a better school, than they would have been assigned to had they handed in their actual true preference list P.

The reasons for demanding a strategy-proof mechanism are manifold. In two contemporary school-allocation cases, Boston and Amsterdam, the following reasons were put forward for demanding a strategy-proof mechanism:

- it gives policy makers insight in the private actual true preferences of students, which allows policy makers to adjust their school policy to actual needs;[17]
- it "levels the playing field by diminishing the harm done to parents who do not strategise or do not strategise well";[18]
- it facilitates communication to parents on the school-allocation process, since it enables authorities to truthfully advise parents and children to hand in their actual true preferences;[19]
- it allows its performance to be measured against the true private preferences of the students, since these actual preferences of students are expected to be almost identical to the handed in student preferences.[20]

To demonstrate that a mechanism is strategy proof it suffices to show that, given some allocation $\mathfrak{M}(\mathcal{M})$, produced by some mechanism $\mathfrak{M}$ on some school market $\mathcal{M}$, no student s could have been assigned to some school which he prefers over his assigned school, by handing in 'strategic' (or untruthful) preferences. A mechanism can be shown to be strategy proof directly, or by demonstrating that the mechanisms satisfies the stronger property of *coalition strategy proofness*. The latter notion implies that no group of students (the coalition) can hand in 'strategic' preferences, such that some coalition member is strictly better off and no coalition member is worse off. To capture these ideas formally, the strategic interaction for school-allocation mechanisms is represented in *normal form* (sometimes also called strategic or matrix form). Games written in normal form require a model of a player's utility in every possible outcome of the game, a formal demarcation of possible strategies by the players and a description of all players:[21]

[16] Abdulkadiroğlu & Sönmez (1998): Theorem 2.
[17] Rechtbank Amsterdam (2015): §4.5.; Abdulkadiroğlu et al. (2006): p. 17.
[18] This was the key motivation for supporting a strategy-proof allocation mechanism by the Superintendent of the Boston Public Schools (Abdulkadiroğlu et al. (2006): p. 17).
[19] Rechtbank Amsterdam (2015): §2.7; Gautier et al. (2015b): pp. 4-5; Abdulkadiroğlu et al. (2006): p. 17.
[20] Abdulkadiroğlu & Sönmez (2003): p. 743.
[21] Leyton-Brown & Shoham (2008): p. 3.

Definition 4.21 (Induced Game) *The game* $\mathcal{G}(\mathfrak{M}, \mathcal{M}) = (\mathfrak{S}, \mathfrak{Q}, \mathfrak{U})$ *induced by some mechanism* $\mathfrak{M}$ *for some school market* $\mathcal{M} = (\mathcal{S}, \mathcal{H}, \mathcal{C}, \mathcal{P})$ *is the game where*

(i) the set of players is $\mathcal{S}$*;*

(ii) the set of strategies $\mathfrak{Q}$ *for each player* $s \in \mathcal{S}$ *is the set of possible preference lists over* $\mathcal{H}$*.*

(iii) the set of real-valued utility (or payoff) functions $\mathcal{U}$ *is s.t. for each student* $s \in \mathcal{S}$*, there is a function* $U_s : \mathfrak{Q} \to \mathbb{R}$*, s.t.*

$$u(Q_s) = (|\mathcal{H}| - 1) - P_s^{-1}(A(s))$$

where $A(s)$ *is the school to which* s *is allocated in the allocation* A *produced by* $\mathfrak{M}$ *for school market* $(\mathcal{S}, \mathcal{H}, \mathcal{C}, (Q_{\{s\}}, Q_{-\{s\}}))$*.*

Notation: A strategy profile (i.e. a vector of all the strategies played by the students) is sometimes written as (Q_N, Q_{-N})*, where* Q_N *denotes the strategies for some set of players* N *and* Q_{-N} *denotes the set of strategies of the other players.*

Note that in the game induced by some mechanism and school market, the set of strategies available for each player is the same. Furthermore, the utility of some player for some given strategy is always calculated in the same way; a player's utility only depends on his truthful preferences, the total number of high schools and the school to which he is assigned; if a student is assigned his first choice his utility is maximal. Given a strategy profile, the utility for each player can be calculated. The standard assumption in game theory is that every player is an active strategic player, that can reason about the strategic behaviour of the other players and that always tries to maximize his utility. Furthermore, it is important to stress that game theoretic investigations are not psychological investigations. This means that if the conclusion of some game-theoretic argument is that some player could have done better, this does not mean that in the real world, it is expected that students will always behave in this better way; it just means that if some player plays some other strategy, his utility is higher. In game theory it is standard practice to argue retrospectively, that is *knowing* the strategies of the other players and thus the (probabilistic) outcome of a game, what are the expected utilities for some student for each respective strategy. Also the notions of strategy proofness and coalition strategy proofness need to be understood in this non-psychological understanding. That is, if, given a certain strategy profile and some allocation produced by the mechanism, a player can have a higher utility playing an untruthful strategy instead of a truthful one, everything else being equal, then the mechanism is *not* strategy proof. Similarly, if there exists some coalition $N \subseteq \mathcal{S}$ such that at least one coalition member can profit from 'strategic' preference lists by some coalition members, without any coalition member being hurt, then the mechanism is *not* coalition strategy proof.

Definition 4.22 (Strategy Proofness) *Let* $\mathcal{G}(\mathfrak{M}, \mathcal{M}) = (\mathfrak{S}, \mathfrak{Q}, \mathfrak{U})$ *be the game induced by some mechanism* $\mathfrak{M}$ *for school market* $\mathcal{M}$*. The mechanism* $\mathfrak{M}$ *is called strategy proof, iff:*

$$\forall s \in \mathcal{S}\ \forall Q_S \in \mathfrak{Q} : Q_s \neq P_s \Rightarrow U(Q_s) \leq U(P_s)$$

where Q_s *is some untruthful strategy and* P_s *is a truthful strategy.*

From the above definition it is clear that the notion of strategy proofness only concerns the behaviour of individuals. That is, if a system is strategy proof, no individual can misrepresent its preferences in such a way, that he will be strictly better off. This notion however does not exclude the possibility of a 'strategic' coalition, in which some some members hand in untruthful preference lists, to benefit some coalition members, without hurting any coalition member. The reasons for wanting a strategy-proof allocation mechanism, can also be used as justifications for desiring a coalition strategy-proof mechanism, hence it is important to also introduce this stronger notion.

Definition 4.23 (Coalition Strategy Proofness) *Let $\mathcal{G}(\mathfrak{M},\mathcal{M}) = (\mathfrak{S},\mathfrak{Q},\mathfrak{U})$ be the game induced by some mechanism $\mathfrak{M}$ for school market $\mathcal{M}$. Let $\mathcal{Q}$ and $\mathcal{T}$ be two strategy profiles, s.t. $\mathcal{T}$ is the truthful strategy profile, i.e. $\forall s \in \mathcal{S} : T_s = P_s$, with $T_s \in \mathcal{T}$ and $\mathcal{Q}$ is a coalition manipulated strategy profile, that is, there is some coalition $N \subseteq \mathcal{S}$, s.t. there exists $n \in N$ for which $Q_n \neq P_n$, with $Q_n \in \mathcal{Q}$. The mechanism $\mathfrak{M}$ is coalition strategy proof iff:*

- $\exists n \in N : U(Q_n) < U(T_n)$;
- $\forall n \in N : U(Q_n) \leq U(T_n)$.

Now note that Definition 4.23 allows the coalition member that is better off to hand in its truthful preferences, hence it is possible for some mechanism to be strategy proof, without being coalition strategy proof. In that case, all coalition members that are strictly better off hand in their true preferences, whereas the coalition members which are not strictly better off possibly hand in untruthful preferences. On the other hand, if some mechanism is coalition strategy proof, then it must also be strategy proof. Suppose some mechanism is not strategy proof, then it must be that there is some student who does not hand in its actual preferences and that is strictly better off by doing so. Now take the set just containing this student. This set is a coalition that renders the mechanism not coalition strategy proof, since in this set (i) there is a coalition member which hands in untruthful preferences, (ii) there is a coalition member that is strictly better of, and (iii) no coalition members are worse off. Hence, any mechanism that is coalition strategy proof is also strategy proof. Thus, for showing that some mechanism is strategy proof, it is sufficient to show that it is coalition strategy proof.

The first 'starting point' mentioned by Amsterdam's municipality official for education was that a school-allocation mechanism should respect the preferences of the students. It is not clear what is precisely meant by this condition, but it is possible to interpret this as (coalition) strategy proofness, by saying that a mechanism does *not* respect students' preferences, when it does not take the actual true preferences of students into account in the strongest possible way. Note that this interpretation is not considered *the right* interpretation of this starting point in the following. It is just mentioned here to stress that (coalition) strategy proofness, could be a 'starting point' for the municipality official for education.

4.2.2 Strategic Investigations of School-Allocation Mechanisms

The notions of strategy proofness and coalition strategy proofness are only interesting if there can actually exists (coalition) strategy-proof school-allocation mechanisms. The first school-allocation mechanisms investigated in detail in the foregoing was the simple and elegant *Permutation Allocation Mechanisms*. The following theorem establishes that the mechanism is coalition strategy proof:

Theorem 4.24 (PAM is Coalition Strategy Proof) *Let $\mathcal{M} = (\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P}$ be some school market. Let $\mathcal{G} = (PAM,\mathcal{M})$ be the game induced by the PAM on school market $\mathcal{M}$. Then PAM is a coalition strategy-proof mechanism.*

Proof. Say $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P}) = A$. Now suppose for reductio that PAM is *not* coalition strategy proof. Then there exists some allocation $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{Q}) = A'$ and coalition $N \subseteq \mathcal{S}$, s.t. the strategy profile $\mathcal{Q}$ differs from the truthful strategy profile $\mathcal{P}$ only for the coalition members, that is $\forall s \in \mathcal{S} : s \notin N \Rightarrow P_s = Q_s$, and we have:

- $\forall n \in N : U_n(Q_n) \geq U_n(P_n)$;
- $\exists n \in N : U_n(Q_n) > U_n(P_n)$.

Let m be the first coalition member occurring in $\overrightarrow{S}$ that is strictly better off (which must exist). By supposition it must be that, $\forall n \in N : \overrightarrow{S}^{-1}(n) < \overrightarrow{S}^{-1}(m) \Rightarrow U_m(P_m) = U_m(Q_m)$, since no coalition member is worse off in A' and m is the first coalition member to be better off. But if the utility for these coalition members is the same, then also the assigned school is the same, that is, $\forall n \in N : \overrightarrow{S}^{-1}(n) < \overrightarrow{S}^{-1}(m) \Rightarrow A(m) = A'(m)$. Observe, that also $\forall s \in \mathcal{S} : s \notin N \wedge \overrightarrow{S}^{-1}(s) < \overrightarrow{S}^{-1}(m) \Rightarrow A(s) = A'(s)$.

For suppose not. Then there is a first non-coalition member s for which $A(s) \neq A'(s)$. If it has any predecessors, then these are non-coalition members s', s.t. $A(s') = A'(s')$ or coalition members n, for which also $A(n) = A'(n)$. Now say $\overrightarrow{S}^{-1}(s) = j$, then at stage j, in both $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P})$ and $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{Q})$ the same seats are available for s, so in both cases the same *best available seat* for s is assigned to s hence $A(s) = A'(s)$: contradiction. Hence for all predecessors s' of m (coalition members and non-coalition members) it must be that $A(s') = A'(s')$, thus it cannot be that $A(m) \neq A(m')$. For if so, then there is seat available at some school h in $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{Q})$, s.t. $h >_m A(s)$, according to m's truthful preferences. But then this seat is also available at stage $\overrightarrow{S}^{-1}(m)$ in $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P})$, because in all stages upto $\overrightarrow{S}^{-1}(m)$, $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{Q})$ behaved just as $PAM_{\overrightarrow{S}}(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P})$, and thus $A(m) = A'(m)$. But the latter means that m is not strictly better off: contradiction. Hence, PAM is coalition strategy proof. □

Corollary 4.25 (PAM is Strategy Proof) *Given that PAM is coalition strategy proof, it must be that PAM is strategy proof.*

In the actual Amsterdam case, with school priorities, if all students with priorities were assigned to their 'priority school' before the PAM would be used to allocate the others, the PAM is not strategy proof for the student population as a whole, but only for the school market on which the PAM is used. Consider the following. Let s be some student with the following preference list $P_s = \langle a, b, c, ...\rangle$, who has a priority for school b, in some school market $\mathcal{M}$. If the PAM assigns student s to school c, than s could have done better by handing in $Q_s = \langle b, a, c\rangle$, thereby using his priority to get a guaranteed seat at school b. Note, however, that the *expected* utility when handing in his actual preference list, with school a on the first preference position, is not $U_s(Q_s) = (\mathcal{H} - 1) - 2$, but is,

$$\sum_{h\in\mathcal{H}} \Pr_{\overrightarrow{S}\leftarrow\Pi_{\mathcal{S}}} \left[PAM_{\overrightarrow{S}}(\mathcal{M})(s) = h\right] \cdot \left(|\mathcal{H}| - 1 - P_s^{-1}(h)\right)$$

since if some other permutation was generated by the PAM, then s was maybe even assigned to school a. Hence, the expected utility when handing in Q_s is the sum over all high schools $h \in \mathcal{H}$ of the probability to be assigned to some school h times the utility when being assigned to that school h. It is not at all clear if the expected utility with handing in true preferences will be lower than the expected utility for $Q(s)$. Hence, while the PAM may, not by definition, be strategy proof in the case *with* priorities on the school market as a whole, it could still be *de facto* strategy proof, because the expected utility for handing in true preferences is higher than, or equally high as going for the sure bet with the 'priority school' placed highest.

By Theorem 4.19 and 4.5 it is clear that (coalition) strategy-proof and Pareto-optimal school-allocation mechanisms for school markets with no priorities exist, namely the PAM or equivalently DA-STB (cf. Theorem 4.19). Not all Pareto-optimal allocations are also (coalition) strategy proof, since the following example shows that the BAM is not strategy proof and thus also not coalition strategy proof:

Example 4.26 *Let $\mathcal{S} = \{1, 2, 3\}$ and let $\mathcal{H} = \{a, b, c\}$, s.t. all schools have capacity 1. Say we have the following preference lists:*

- $P_1 = \langle a, b, c\rangle$;
- $P_2 = \langle a, c, b\rangle$;
- $P_3 = \langle a, b, c\rangle$;

Now $BAM(\mathcal{S},\mathcal{H},\mathcal{C},\mathcal{P}) = \{(1, b), (2, a), (3, c)\}$, which was produced in the following order:

- *Round 1: all students place school a as their first choice, hence there is one lottery in this round and it takes place for school a. Say $\overrightarrow{S}^a_1 = \langle 2, 1, 3\rangle$. Then student 2 is assigned to school a.*
- *Round 2: the two not allocated students, 1 and 3, both place school b as their second choice, hence there is a one lottery and it take place for school b. Say $\overrightarrow{S}^b_2 = \langle 1, 3\rangle$. Then student 1 is assigned to school b.*

- *Round 3: student 3 is assigned to school c*

With the outcome of the lotteries and the strategies of the others known, could student 3 have done better? Yes, if student 3 hands in the 'strategic' preference list $Q_3 = \langle b, c, a \rangle$, then student 3 would have been allocated to school b and we would have the alternative allocation $A' = \{(1, c), (2, a), (3, b)\}$. We then have $U_3(P_3) < U_3(Q_3)$, since $U_3(P_3) = 2 - P_3^{-1}(c) = 2 - 2 = 0$, $U_3(Q_3) = 2 - P_3^{-1}(b) = 1$ and $0 < 1$. This means that under 'strategic' preferences Q_3 student 3 is better off, hence the BAM cannot be strategy proof.

In the above example it is unwise for student 3 to hand in its actual preferences. In general, it is unwise to place a very popular high school highest on the preference list under the BAM, if there is also another unpopular high school that is preferred (almost) equally. If an unpopular high school is placed highest, then students are almost certain to be assigned to that school by the BAM, whereas it is very uncertain to be assigned a popular high school. Furthermore, under the BAM it is not recommended to place a popular high school on any place below first, since the high school will probably already reach its capacity in the first round, meaning that the student will not participate in any lottery in the round corresponding to the preference position of the popular high school.[22] Note however, that it can also be seen as an advantage of the BAM that it tempts students in placing the unpopular high school first, if it is preferred almost equally much as a popular most preferred high school, since this makes the capacity problem less severe.

The result from Example 4.26 that the BAM is not strategy proof also has implications for the meaning of Corollary 4.13, claiming that the BAM is Pareto optimal. Whether a system is Pareto optimal depends on the preferences of the students, that is, the *handed in* preferences of the students. It is expected when a not strategy-proof school-allocation mechanism is used, that the handed-in preferences do not reflect the students' actual preferences. This could render the allocations, produced by the not strategy-proof mechanism, *not* Pareto optimal when the truthful preferences are considered, in stead of the handed-in preferences.[23] Hence, it is very hard to determine if the BAM is actually Pareto optimal in light of the true preferences. If a school-allocation mechanism is strategy proof and can be shown Pareto optimal, then it can be claimed with almost certainty that there *cannot* exists some trade cycle of students, based on their actual true private preferences, which will benefit all trade-cycle participants.[24]

In the earlier discussion on deferred acceptance school-allocations mechanisms it was shown that the DA-MTB was not Pareto consistent. From the official correspondence of the municipality official of eduction in Amsterdam, which was released after DA-MTB was used in Amsterdam, and after the court case in which its use resulted, it is clear that the city of Amsterdam, does not want to use non-Pareto-consistent mechanisms again. Since the Amsterdam case is taken as the point of departure for this study, the game-theoretic properties of the DA-MTB will not be investigated in detail. It is however important to say, without any formal demonstration, that, while the mechanism is not Pareto optimal, it is coalition strategy proof.[25]

4.3 Simulations with Mechanisms

For testing how the three investigated school-allocation mechanisms (PAM, BAM and DA-MTB) perform in the Amsterdam case, the actual handed-in preference lists of students and the real high-school capacities from 2015 are used.[26] This data was obtained from the municipality of Amsterdam. To make the tests computationally more feasible, the school-allocation mechanisms are only tested on the VWO population.

[22] Note that in the Amsterdam version of the BAM, where students hand in their preferred school *per round*, it is impossible that students cannot participate in some round, since the students are not allowed to hand in a preferred school that already reached its capacity (cf. Section 2.3).

[23] Abdulkadiroğlu & Sönmez (2003): p. 734.

[24] It is impossible to be certain, since in the real world students could hand in untruthful preferences for a strategy proof mechanism.

[25] The DA-MTB as described in this thesis is actually the student proposing deferred acceptance algorithm with multiple tie-breaking, it was first shown in Dubins & Freedman (1981) that this mechanism is coalition strategy proof from the perspective of the students. The result is reconstructed in detail in Roth & Sotomayor (1990).

[26] By Theorem 4.19 we have that the DA-STB is equivalent to the PAM (or RSD) since no priorities are incorporated in the high-school preferences.

The VWO population are all students who, (i) only place VWO high schools in their preference list (including HAVO-VWO schools), and (ii) are allowed to do so. Note that in (i) we mean by high schools, (HAVO-)VWO application possibilities, and not actual high schools. Furthermore, note that (ii) means that the students in the VWO population had a VWO or HAVO-VWO school recommendation. The VWO population consists of 1885 students and 39 high schools, where a null school is included.

The three school-allocation mechanisms are tested on how many students they leave not allocated, how many students are assigned their first choice, and what the mean preference position of the allocation is; all averaged over 100 runs of the respective mechanism. The mean preference position of some allocation $\mu(A)$ for some given school market $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{C}, \mathcal{P})$ is,

$$\mu(A) = \frac{\sum_{s \in \mathcal{S}} 1 + P_s^{-1}(A(s))}{|\mathcal{S}|}$$

For every run a different *seed* is used. A *seed* is an integer from which all pseudo randomness in the school-allocation mechanism is deduced. Normally, a different seed will result in a different allocation produced by a school-allocation mechanism. For the PAM a different seed results in a different permutation; for the BAM a different seed results in different lotteries for each school in each round; for the DA-MTB a different seed results in different (artificial) high-school preferences. The DA-MTB tested here is such that each high school is assigned a random permutation as its preferences over students. The seeds used were the first 100 natural numbers in their natural order, that is $\langle 0, 1, 2, ..., 99 \rangle$. The computational implementations of the mechanisms tested are found in Chapter 7. The tests gave the following results:

	Not Allocated	First Choice	Mean Pref. Pos.	Par. Cons.	Coa. Strat. Proof
PAM:	6.93 (0.36 %)	1566.05 (83.08 %)	1.2970	Yes	Yes
BAM:	8.13 (0.43 %)	1636.00 (86.79 %)	1.2606	Yes	No
DA-MTB:	0.38 (0.02 %)	1379.64 (73.19 %)	1.3545	No	Yes

Note that if some student is not allocated, the preference position is said to be one more than the preference position of the last school mentioned in his preference list. Sometimes, leaving some student not allocated benefits other students and thus brings the mean preference position of the population down, but this is not necessarily the case. From the above table it is clear that the mechanism which allocates the most students is DA-MTB. The results also show that the BAM assigns most students their first choice and that the BAM has the lowest mean preference position on average.

The above statistics suggest that the PAM, on the properties tested here, is always out performed. Note however, that this is only the case when focused on the statistics and not on the formal properties, since the PAM is the only tested mechanism that is Pareto consistent *and* coalition strategy proof. Furthermore, note that the results of the BAM are very difficult to interpret, since the BAM is not strategy proof. Hence, under the BAM, students are *not* expected to hand in their actual true school preferences, the BAM is tested, however, on preference lists handed in for a strategy-proof system. This means that the mechanism is tested on preference lists that resemble the actual true preferences of students very closely. The statistics therefore, are not likely to give a good impression of how the system would perform in practice. To test this, expected strategic preference list of students, corresponding to their true preference lists, should be obtained, on which the mechanism is then ran. After this, the number of students who are not allocated, the number of students assigned their first choice, and the mean preferences positions can be calculated, by using the actual true preferences. Moreover, since the DA-MTB is not Pareto optimal, the above results of the DA-MTB are also hard to interpret, and cannot be translated one-to-one into expectations about the satisfaction of students and parents with the produced allocation. If students find someone, with whom they could switch places to be both better off, it is unlikely that the student will still consider the allocation satisfactory, even though the allocation has nice statistics for the whole population. Hence nice statistics concerning the number of students not allocated, do not say very much, if students who are allocated, can be very unhappy with their assigned school, since they know they can, without harming anyone, be assigned a better school.

Other simulations by Gautier et al. (2015b) on the Amsterdam data, for these respective three systems, showed that DA-MTB places 73.6% of students on their first choice, whereas DA-STB place 82% of students on their first choice. The DA-STB in the tests by Gautier et al. (2015b) differs from the DA-STB/PAM in the simulations here, because Gautier et al. (2015b) incorporated priorities in the school preferences, and then used one single permutation to create the strict preferences of high schools over students.[27] Nevertheless, the obtained DA-STB in the simulation by Gautier et al. (2015b) comes very close to simulations by our PAM, because the system of priorities is not very extensive and complicated in Amsterdam, and confirm that DA-STB/PAM is expected to place more students on their first choice than the DA-MTB. Furthermore, Gautier et al. (2015b) show that the number of students placed in their top 3 and top 5 is almost similar for DA-MTB and DA-STB (their version): top 3, 94.7% for DA-MTB and 94.8% for DA-STB; top 5, 98.5% for DA-MTB and 97.4% for DA-STB.

4.4 Mechanisms for Amsterdam

From Chapter 3 it became clear that it is the wish of the municipality of Amsterdam, to have Pareto-optimal allocations. This desire was made public after a *not* Pareto-optimal allocation was produced in 2015 by the DA-MTB. The only mechanisms that are investigated in Chapter 4 that produce Pareto-optimal allocations are the PAM and the BAM. It became clear in the legal proceedings, after the DA-MTB was used in Amsterdam, that both the municipality and the representative body of high schools in Amsterdam, preferred to have a strategy-proof allocation mechanism. Since only the PAM is both Pareto consistent and (coalition) strategy proof, it seems that from the three investigated school-allocation mechanism, PAM is most in line with the desires of the authorities. Furthermore, the simulations with the three mechanisms showed that the number of students who receive their first choice is very high under the PAM and only differs marginally from the BAM. Moreover, the allocations produced by the PAM are all the Pareto-optimal allocations for some school market, whereas the allocations produced by the BAM are only a strict subset of those. Another advantage of the PAM, not discussed earlier, but found in the literature, is that it is the only one-sided mechanism that is strategy-proof, nonbossy and neutral.[28] A mechanism is nonbossy, if a student cannot change the outcome of the mechanism, by handing in a different preferences, without also changing the school to which he is assigned;[29] a mechanism is neutral if no school is treated differently than some other school.[30] If policy makers thus want a *strategy-proof*, *nonbossy* and *neutral* school-allocation mechanism, then their only option is the PAM.

The simulation also showed, that the PAM had some disadvantages. In comparison with the DA-MTB, the PAM tends to leave more students unassigned and requires students to hand in longer preference lists then needed under DA-MTB. Furthermore, in 2009, Abdulkadiroğlu, Pathak and Roth showed that, if both parties have preferences, then no strategy-proof mechanisms can Pareto improve on allocations produced by any variant of the deferred acceptance school-allocation mechanisms. That is, in any allocation produced by some other strategy-proof mechanism, there will be at least one student, who is worse off under this allocation, then under an allocation produced by a deferred acceptance school-allocation mechanism. This also means that no PAM produced allocation dominates deferred acceptance produced allocations. However, also allocations produced by deferred acceptance, do not dominate PAM produced allocations either (which is expected, since the PAM is Pareto optimal). Intuitively, their result entails that it is never the case that there is a deferred acceptance produced allocation A and a PAM produced allocation A', such that all students are equally well or better off in A as/than in A', or vice versa.[31] Since in the Amsterdam case under discussion in this thesis, the municipality official explicitly suggested that any future allocation mechanisms used should not allow any reasonable trading options under students, in the final allocations, deferred acceptance produced allocations, in which not all high schools have the same preferences, are excluded, since they are not necessarily Pareto optimal (cf. Example 4.18). Hence, notwithstanding the fact that deferred acceptance produced allocations cannot be Pareto dominated by PAM produced allocations, deferred acceptance produced allocations, other

[27] Gautier et al. (2015b): p. 5.
[28] Svensson (1999): p. 558.
[29] Svensson (1999): p. 560.
[30] Svensson (1999): pp. 561-562.
[31] Abdulkadiroğlu et al. (2009): pp. 1692-1693.

than DA-STB, are not recommended for use in Amsterdam, since they are not Pareto optimal.

Another important result by Lin Zhou, from 1990, showed, that there cannot be a one-sided mechanism, in which either party has more than 2 members, that is Pareto consistent, strategy proof and symmetric. A mechanisms is symmetric, if two student's that hand in the same preference list, always have the same utility (N.B. actual utility not expected utility). Zhou's result shows that it is impossible to have a mechanism, that is as the PAM, in that it is strategy proof and Pareto consistent, but that is also symmetric. The PAM is clearly not symmetric, since the order in the permutation discriminates between students. If however, a symmetric mechanism is desired by policy makers, then Zhou's result implies that either strategy proofness or Pareto consistency must be abandoned as a further condition on the school-allocation mechanism. Zhou's result can be interpreted as saying that if strategy proofness and Pareto consistency are required of a mechanism, then a lottery element in the school-allocation mechanism cannot be excluded.

The above shows that the PAM has certain advantages over the other school-allocation mechanism investigated in this thesis and other school-allocations mechanisms not yet designed. However, the PAM on itself, does not always select nor favour Pareto-optimal allocations in which most students are assigned their first choice. That is, the second 'starting point' of the municipality official for education, understood as *as much students as possible should be assigned their first choice*, is not something the PAM can guarantee, as is clear from the simulations and the discussion in Section 3.5. The next chapter will try to find a way in improving the PAM such that its desirable properties will be kept and that it *will* favour allocations in which the most students are assigned their first choice.

Chapter 5

Genetic Optimisation

5.1 Optimisation

From the first chapter it is clear that the municipality official of Amsterdam wished any school-allocation mechanism to be used for solving the Amsterdam allocation problem, to satisfy the following conditions:

1. it needs to respect students' preferences;
2. it needs to assign as much students as possible to their first choice;
3. it needs to be Pareto consistent;
4. it needs to be comprehensible and explicable;
5. it needs to be fast.

As is clear from the foregoing not all school-allocations mechanisms are able to satisfy the third condition. From Theorem 4.5 it is clear that there is a school-allocation mechanisms that is able to produce all the Pareto-optimal allocations that can possibly exist for some given school market, viz. the PAM. In this chapter it will be investigated if it is possible to exploit this property of the PAM, by optimising its results, such that it also satisfies the other conditions put forward by the municipality official. The method of optimisation will be *genetic-algorithmic*. A genetic-algorithmic optimisation method uses concepts from genetics to converge to an optimum in a very large problem space. The optimum in the case of the Amsterdam allocation problem seems to be some allocation that is Pareto optimal and also assigns most students to their first choice. More conditions can be posed on this optimum and genetic algorithms allow its users to alter what the optimum is, without needing to rebuild the entire algorithm, but only to adjust a very small part, viz. the fitness function. By using a genetic algorithm for finding an optimal allocation in the set of Pareto-optimal allocations, it can be said that the PAM is optimised, since the allocations producible by the PAM are equivalent to the set of Pareto-optimal allocations. Moreover, in the following it will become clear that actually the PAM is an integral part in genetic-algorithmically finding the optimal allocation, hence it can rightfully be claimed that the PAM will be genetic-algorithmically optimised. Before the genetic-algorithmic optimisation method of the PAM can be explained, first the concept of genetic-algorithmic optimisation will be introduced.

5.2 Genetic Algorithms

Genetic algorithms were introduced by John Holland in the 1960s and were further developed and improved by Holland's research group at the University of Michigan in the 1960s and the 1970s.[1] In the words of one of Holland's PhD students, "genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics".[2] Genetic algorithms use the idea of survival of the fittest among data, determined

[1] Mitchell (1996): p. 3.
[2] Goldberg (1989): p. 1.

by some given fitness function, combined with a structured yet randomised information exchange, inspired from the concepts of mutation and crossover from genetics, to provide a robust and very efficient optimum search algorithm, to find an optimum set of data points in a given search space. In short, a genetic algorithm reproduces a sample of organisms, from an initial population, in such a way that, without heaving to go through the entire population, the fittest data points can be found.[3] More formally, the search space elements, or organisms, are coded in finite-length strings over some finite alphabet, to form chromosomes; these chromosomes go through some evolutionary process of the genetic algorithm, by undergoing reproduction (or selection), crossover and mutation; finally some fittest chromosomes are obtained, which can be decoded, to give optimal organisms, which form with high probability a search space optimum. The notion of a genetic algorithm is not formally demarcated, but rather is a concept under which different algorithmic techniques fall, which all make use of concepts of genetics to find 'fittest' data points. Genetic algorithms differ from other optimisation procedures in that they (i) work with a coding of the search space elements, not the elements themselves, (ii) search from an initial population of data points and not a single data point, (iii) use an objective fitness functions to determine the fitness of data points, and (iv) use probabilistic instead of deterministic transition rules to traverse the search space.[4]

The most rudimentary genetic algorithm consists of three types of operations: reproduction (or selection), single-point crossover and mutation. Reproduction selects the organisms in the population that will survive the shift to a new generation; normally, the higher the organism's share in the total fitness of the population, the higher its probability to be reproduced. Crossover randomly selects a cut-off point and then exchanges the subsequence before and after the cut-off point between two chromosomes that correspond with two organisms. Mutation randomly changes some letter in the coding of the chromosome to obtain another finite string. Mutation can normally occur at each position in a chromosome, but with only a very small probability.[5] A genetic algorithm then iterates these operations multiple times such that in each iteration, first reproduction is performed, then some reproduced organisms are crossed over, and finally some crossed over organisms will be mutated; each iteration of the three operations will result in a new population. Each such iteration will be called a generation, therefore also the new populations produced by an iteration will be called a generation. The three operations are typically iterated 50 to 500 times. The entire set of generations so obtained will then be called a run. Randomness plays a large role in deciding how the three operations will behave, carrying out a genetic algorithm with different pseudo-random number seeds will therefore often yield different sets of highly fit organisms.[6] To give some intuition on genetic algorithms and its operations a simple genetic algorithm is introduced.

Example 5.1 *Say in the set of bit strings of length 6 we want to find the highest* double prime *bit strings.*

> *A bit string is* double prime*, if (i) the integer value of the string is prime and (ii) the number of ones occurring in the string sum op to some prime. For instance, the bit string 000011 is double prime, since 000011 has integer value 3 (i.e.* $0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 = 3$*) and the string 000011 only has 2 ones.*

In the case of length 6 bit strings, there are only $2^6 = 64$ *possible candidates, hence the highest double prime bit string could be found by hand, however, in the case of length 100 bit strings, there are about* $1.28 \cdot 10^{30}$ *possible candidates (a number of 31 digits), hence computing by hand or by brute force combinatorial methods what the highest double prime bit string of 100 digits is, seems to be cumbersome. A genetic algorithm is often more efficient to find the optimum.*

To clearly explain the genetic approach, the length 6 bit strings case will be treated. As a coding of the organisms, the bit strings, we simply use the bit strings themselves, since these are already finite strings over a finite alphabet. In this case organisms and chromosomes are thus identical (in most other

[3] Goldberg 1989: p. 1.
[4] Goldberg (1989): p. 7.
[5] Mitchell (1996): p. 8.
[6] Mitchell (1996): p. 9.

cases the correspondence is not identity). After a sufficient coding is found, a fitness function should be constructed, to formally capture the optimum desired to be found. Let $v(x)$ be the integer value of some bit string x and let $p(n)$ be the function that returns 1 if integer n is prime and 0 otherwise, furthermore, let $s(x)$ be the sum of ones in some bit string x. Then the following fitness function captures what is needed:

$$f(x) = v(x) \cdot p(v(x)) \cdot p(s(x)) + 1$$

With the above fitness function, all non double prime bit strings have a fitness value of one (which is lower then the first double prime bit string 000011) and the larger the double prime bit string the higher its fitness value. With a fitness function constructed, a reproduction method needs to be chosen. The standard reproduction method is the roulette-wheel reproduction method*, in which the expected number of times a chromosome will be selected to be reproduced is that chromosome's fitness divided by the total fitness of the population; that is, the individual's weight in the total fitness determines the chance of surviving into the crossover and mutation pool. As the mutation function on bit strings, we simply flip a biased coin, according to the mutation parameter, for each bit occurring in a bit string, if the coin comes up on some fixed side, then we flip that bit, otherwise we move on to the next bit in a bit string. For instance, given bit string* 100110 *and mutation parameter* 0.33*, we will flip a coin c s.t. $Pr[c \text{ is heads}] = 0.33$ and $Pr[c \text{ is tails}] = 0.67$. Then for each bit b_i in* 100110 *we flip the coin, say this gives: T, T, H, T, T, H. Then the third and last bit in* 100110 *will be flipped and the string* 101111 *is thus obtained. For crossover we interchange the initial segment and 'leftover' bits of two bit strings, given some random cut off point. For instance, given bit strings* 010101 *and* 101100 *and cut-off point 4, we obtain the following bit strings after crossover:*

0101 00 and 1011 01,

where $0101 \mid 00$ *is the 4 initial segment of 010101 combined with the last two bits of 101101, and similarly* $1011 \mid 01$ *is the 4 initial segment of 101100 combined with the last two bits of 010101. Say we begin with some random initial population; then randomly, but proportionally according to the fitness value, some strings are selected from the initial population to be reproduced in the new generation. Let the initial population be the strings x_1 to x_6:*

	x_i	$v(x_i)$	$p(v(x_i))$	$p(s(x_i))$	$f(x_i)$	$P(x_i) = \frac{f(x_i)}{\sum_{1 \le j}^{6} f(x_j)}$
x_1	*110111*	*55*	*0*	*1*	*1*	≈ 0.019
x_2	*100101*	*37*	*1*	*1*	*38*	≈ 0.703
x_3	*001011*	*11*	*1*	*1*	*12*	≈ 0.222
x_4	*010111*	*23*	*1*	*0*	*1*	≈ 0.019
x_5	*000100*	*8*	*0*	*0*	*1*	≈ 0.019
x_6	*001100*	*12*	*0*	*0*	*1*	≈ 0.019

Then each string x_i is selected by the roulette-wheel reproduction *with probability $P(x_i)$, hence we expect the selected organisms to be almost all x_2 and some x_3. It seems reasonable that we want more strings with a structure close to x_2 to reproduce, since we are looking for the* highest *double prime, and $v(x_2) > v(x_3)$. Then of the selected strings, some percentage according to some fixed* crossover parameter *will be crossed over and also, some percentage according to some fixed* mutation parameter *which will determine the mutation coin flip, will be mutated. This will yield a new* generation *of length 6 bit strings, of which again some will be selected to undergo the creation of a new generation, of which again some will be mutated and crossed over. Finally, after some given number of generations is created, some population of bit strings will be obtained. In every creation of a new generation, the fittest bit strings are more likely to make it to the new generation. Therefore, it is expected, that after a sufficient number of generations, the fittest length 6 bit string, that is the highest double prime length 6 bit string, is an element of the last generation (this should be 111101, since the integer value of 111101 is 61, which is prime, and 111101 contains 5 ones, which is also prime).*

In the above example the *highest* double prime bit string is searched for. Knowing that we want the highest, in terms of integer value, there are of course more efficient methods of finding this bit string, then to write a

complete genetic algorithm. For instance, all even bit strings (ending with 0) can immediately be discarded, since they can never be double prime. Other similar methods can be used, to decrease the search space and to eliminate the need for a smart searching method. However, such methods are not easily found, when the optimal being searched for, is not so well understood as the property of being double prime. In the case were, among all allocations for some school market, the Pareto-optimal allocations with most students assigned to their first choice is looked for, it is, for instance, not clear how the search space can be greatly minimised.

Example 5.1 also illustrates that genetic algorithms need more than the operation of reproduction, because otherwise no new areas of the search space will be covered. If the only manipulation on the population performed by a genetic algorithm is reproduction, then the genetic algorithm will simply converge to the fittest organism in the initial population. If, however, the genetic algorithm also performs crossover and mutation, then, in light of the specific example from Example 5.1, the string 111101 (the optimum) can be found, whereas with only reproduction, the genetic algorithm is expected to result in a population with just the string 100101 (the optimum of the initial population). All three basic operations of a genetic algorithm (reproduction, mutation and crossover) are as important. Where reproduction ensures convergence, crossover ensures giant jumps within the search space and mutation ensures nearby search in the search space.

5.3 Reproduction

In Example 5.1 reproduction was carried out using a *roulette wheel*. This is just one among many methods for selecting which organisms should be reproduced. Other widely used methods are *total-order reproduction* and *expected number control reproduction*. These other reproduction techniques will not be discussed, since the genetic algorithm introduced in this thesis for optimising school allocations, will also use a *roulette-wheel* reproduction technique.[7] In *roulette-wheel reproduction* each organism is reproduced with a probability corresponding to its share in the total fitness. To carry out this reproduction technique a specific bound on the generation size must be given in advance. The method for carrying out this fitness function differs for different *types* of fitness functions. If the reproduction method is used in combination with a *fittest-0* fitness function, where the fittest organism receives a fitness of 0, then the reproduction technique is slightly more cumbersome, than when used in combination with a *fittest highest* fitness function, where the fitter the organism, the higher the fitness value. In this thesis a *fittest highest* fitness function is used, hence the reproduction technique will only be discussed for this case:

(Definition on next page)

[7]For more on *total-order reproduction* see 'rank selection' in Mitchell (1996): p. 127; for more on *expected number control* consult Goldberg (1989): p. 24.

Definition 5.2 (Roulette-Wheel Reproduction) *Let $X = \langle x_1, ..., x_n \rangle$ be a finite population of organisms and let $f : X \to \mathbb{R}$ be some fitness function, s.t. the further $f(x)$ is away from zero the fitter x is. Let $c \in \mathbb{N}$ be some given bound on the generation size. All procedures extensionally equivalent to the following, are called a* roulette-wheel reproduction technique*:*

input : *Finite list of organisms $\overrightarrow{X} = \langle x_1, ..., x_n \rangle$, bound on generation size $c \in \mathbb{N}$.*
output: *Finite list of organisms $\overrightarrow{Y}$ s.t. all organisms in Y also occur in $\overrightarrow{X}$.*

for $0 \leq i \leq |\overrightarrow{X}| - 1$ **do**
  if $i < |\overrightarrow{X}| - 1$ **then**
    associate $\overrightarrow{X}(i)$ with interval $[\sum_{j<i}^{n} p_j, \sum_{j\leq i}^{n} p_j)$, where $p_j = \dfrac{f(\overrightarrow{X}(j))}{\sum_{0\leq k}^{n} f(\overrightarrow{X}(k))}$;
  else
    associate $\overrightarrow{X}(i)$ with interval $[\sum_{j<i}^{n} p_j, \sum_{j\leq i}^{n} p_j]$;

Generate a list $\overrightarrow{R} = \langle r_1, r_2, ..., r_c \rangle$ of c random real numbers s.t. $0 \leq r_i \leq 1$;
for $0 \leq i \leq |\overrightarrow{R}| - 1$ **do**
  reproduce the $\overrightarrow{X}(i)$ in which $\overrightarrow{R}(i)$ falls;
return *a list Y of all reproduced organisms.*

Note that when the returned list Y is created in the order corresponding to the order of intervals in which the r_i fall, then the list Y is a random permutation of the reproduced organisms. That is, if, for instance, r_1 falls in the interval assigned to x_5 and $Y(0)$ is set to x_5, the list Y so obtained is a random permutation of the reproduced organisms. Furthermore, observe that it is possible to carry out the procedure on *sets* of organisms as well, by just multiplying the fitness of each organism by its frequency in the population. That is, if some organism x occurs k times in the population, then just multiply x's fitness with k and then proceed as above; this will yield the same reproduced organisms. Moreover, note that the probability of some organisms x_i being reproduced in the procedure described in Definition 5.2 is the weight of the organism's fitness in the total fitness, that is:

$$\Pr[x_i \text{ is reproduced }] = \frac{f(x_i)}{\sum_{j=1}^{n} f(x_j)}$$

This also means that the expected number of x_i occurring after reproduction is x_i's share in the total fitness times the given bound c on the generation size, that is:

$$c \cdot \Pr[x_i \text{ is reproduced }]$$

Note that under roulette-wheel reproduction, an organism is only reproduced with certainty if that organism's fitness is the total fitness (since then it will have a probability of 1 to be reproduce). That is, organisms are only guaranteed to be reproduced under roulette-wheel reproduction if the genetic algorithm converged to some chromosome. To ensure that in earlier stages the fittest organisms are not eliminated by accident by the roulette-wheel reproduction technique, the creation of a new generation is often carried out in an *elitist* manner. If *elitism* is applied, it will be ensured that the fittest organism will reproduced. Moreover, in elitism it is normal practice to protect a copy of the fittest organism from crossover and mutation as well. Elitism was first introduced by Kenneth De Jong, in 1975, and was shown by many researchers to significantly improve the performance of genetic algorithms.[8]

[8] Mitchell (1996): p. 126; De Jong (1975): p. 102.

5.4 Genetic Optimisation of School Allocations

The goal of this thesis is to investigate the possibility of using genetic algorithms for optimising school allocations. As a point of departure for this study, the Amsterdam school-allocation problem is chosen. From previous chapters it is clear that the problem can be understood as finding the *fittest* Pareto-optimal allocation. What will be considered fittest is given by the 'starting points' from the municipality official for education. The upside of using genetic algorithms to find optima is that the fitness function can be adjusted to the present need or understanding of what must be considered the fittest allocation. If in the future another Pareto-optimal allocation is considered 'fittest', for instance, the one with the lowest mean preference position, then only the fitness function needs to revised without altering the rest of the genetic optimisation method. In the next sections the genetic algorithm used for optimising the Pareto-optimal allocations will be introduced, which will make clear why it is possible to force the genetic algorithm to produce different allocations by just revising the fitness function.

5.4.1 Organisms and Chromosomes

As indicated before, genetic algorithms find the optimal organisms by manipulating chromosomes. Chromosomes are, in the genetic-algorithmic universe, finite encodings of organisms. The best suited finite strings in some finite alphabet available for encoding Pareto-optimal allocations are *permutations*. From Theorem 4.5 it became clear that to every Pareto-optimal allocation there corresponds at least one permutation for which the PAM produces that allocation, and that to every permutation there corresponds a Pareto-optimal allocation. To obtain school allocations from permutations (decoding) the PAM can simply be used. The encodings for Pareto-optimal school allocations used in the genetic algorithm presented in the following, the chromosomes, will thus be permutations of students; the organisms are then the corresponding school allocations. In the setting of genetics and thus genetic algorithmics, a chromosome can conceptually be divided into genes, such that each gene encodes a trait of the organism. Each gene is located at some particular position on the chromosome: the locos. The different possible settings for a gene, at a certain locus, are called alleles. [9] Hence in the present case, a chromosome, some permutation $\overrightarrow{S}$, can be divided into the different places of that permutation, i.e. $\overrightarrow{S}(0)$, $\overrightarrow{S}(1), ...$, which form the loci, on each locus a student is located, hence students are the genes in the present setting; which student is located at $\overrightarrow{S}(i)$ then determines which allele makes up the gene.

A genetic algorithm optimises organisms by reproducing, mutating and crossing over chromosomes. The only time organisms are needed is when the fitness of some chromosome needs to be determined. The fitness of a chromosome is given by the fitness of its corresponding organism. In the present case, the fitness of some permutation will thus be calculated by calculating how 'fit' the unique corresponding allocation is. By defining the operations that manipulate the chromosomes, crossover and mutation in such a way that a chromosome (some permutation) will never be manipulated in something that is not a permutation of students, we ensure that each generation consists of permutations of students in the given school market. Theorem 4.5 establishes that it is possible to search through the set of Pareto-optimal allocations to find the fittest Pareto-optimal allocations for some given school market by searching through the set of permutations on students in that school market. From Corollary 4.8 and Example 4.7 it is known that some allocation can be encoded into different permutations and that there is thus no one-to-one correspondence between permutations and allocations (but only a surjective correspondence). This is not problematic: it only implies that the genetic algorithm will sometimes perform some redundant search efforts trough the space of Pareto-optimal allocations for some given school market. That is, some manipulation (crossover or mutation) performed by the genetic algorithm on some permutation of students might not yield a new corresponding school allocation (e.g. say the genetic algorithm manipulates $\overrightarrow{S}$ from Example 4.7 s.t. it becomes $\overrightarrow{S}'$, then this manipulation does not lead to a new part of the search space being covered). The probability that some manipulation will be redundant for an actual very large student population, as in the Amsterdam case, is however extremely small.

Now that it is clear what should be the chromosomes to find the optimal organism(s), the genetic algorithm for optimising school allocations can be introduced. The genetic algorithm presented will be called GOPAM,

[9] Mitchell (1996): p. 5.

which is short for *Genetic Optimisation of the Permutation Allocation Mechanism*. This name is chosen, because the PAM is an integral part of the GOPAM, since it is used for turning chromosomes (permutations) into organisms (allocations), and it is the nature and formal results of the PAM, that ensure that permutations can be used to optimise Pareto-optimal school allocations. As in the example of the *double prime* bit strings, the GOPAM will be introduced by first presenting the fitness function and the method of reproduction. Next, the methods for manipulation chromosomes, crossover and mutation, needed to widen the scope of the search in the Pareto-optimal allocations for some given school market, will be presented. Finally some general remarks on the allocations produced by the GOPAM and the formal properties of these will be discussed.

5.4.2 Fitness Function and Reproduction

Now that is clear what will be optimised, conditions must be set that give when a certain organism is fitter than some other organism. In the present case, all organisms are Pareto-optimal allocations, so the fitness function of the genetic algorithm only needs to capture whether some Pareto-optimal allocation A is considered fitter than some other Pareto-optimal allocation A'. From the official correspondence of the municipality official for education different fitness conditions can be deduced, since the 'starting points' were not univocal: the *fittest* Pareto-optimal allocation is the Pareto-optimal allocation in which either (i) most students are assigned their first choice, or (ii) in which students are only not allocated to their highest preferred school, if it is not possible to allocate all students who prefer that school highest to that school. Both understandings of the second 'starting point' can be implemented in the fitness function, but since only the first understanding gives a comparative concept,[10] that understanding will be considered the primary fitness condition, to which the second understanding can be supplemented in the form of a penalty. Later when variants of the fitness functions are discussed, the addition of the penalty to capture the second understanding will be further investigated, but for now only the first understanding will be incorporated in the fitness function. To capture the first understanding, a *fittest highest* fitness function, is best suited, since it is not clear what the fittest possible Pareto-optimal allocation will be in the search space, but it is clear to say which allocation is *fitter*. Another fitness condition, more in line with the first 'starting point' that a school-allocation mechanism should respect the preferences of students is to deem some allocation A fitter than some allocation A', if in allocation A there are less students unassigned than in A'. This condition makes sure that allocations which disregard certain preference lists by not allocating the owners, will be considered less fit. The following fitness function both captures *the more students assigned first choice the fitter* and *the less students not allocated the fitter*:

$$f(A) = \left(\frac{1}{g(A)+1} \cdot h(A) \right)^2 \text{ where}$$

- A is some allocation;
- $g(A)$ is the number of students not allocated in A, i.e. the number of students assigned to the null school;
- $h(A)$ is the number of students assigned to their first choice.

The above fitness function penalises any allocation for leaving students not allocated. The penalty in the fitness function above is very harsh and can be adjusted to any penalty found appropriate by policy makers for capturing the problem of leaving students not allocated. Note that not allocated students can be allocated by hand after the allocation is produced by the mechanisms, since the total number of school seats far exceeds the number of applying students (e.g. in the VWO case, 1885 VWO students and 3109 VWO application possibilities). Furthermore, observe that the value obtained by multiplying the penalty with the number of first-choice students is squared. This method of enlarging the differences in fitness values between allocations is called *fitness scaling*. This is done to make small differences between allocations entail larger differences in fitness value. For instance, say two allocations A and A' have the same number k of students not allocated, but in allocation A there is one student less assigned its first choice than in A', say l vs. $l+1$. Then the fitness values without the fitness scaling of A and A' would respectively be,

[10] i.e. a concept that allows one to formulate "a comparison in the form of a more-less-statement, without the use of numerical values" (Carnap (1967): p. 9); in casu 'more-less fitter'.

$$\frac{1}{k+1} \cdot l = \frac{l}{k+1} \quad \text{and} \quad \frac{1}{k+1} \cdot (l+1) = \frac{l+1}{k+1},$$

which on larger numbers only gives an almost negligible difference in fitness value. This is problematic, since the odds for both allocations to be reproduced would then be almost similar, which impedes the genetic algorithm from converging to the better allocation A'. With fitness scaling, by squaring the value, the difference in fitness value between A and A' becomes significant.

The fitness function described above is necessary to perform reproduction. The method of reproduction in the GOPAM will be the *roulette-wheel reproduction method*. As explained in the foregoing, this means that each organism has a probability to be reproduced that is equal to its weight in the total fitness of the population. The reproduction in the GOPAM is performed with *elitism*. This means that before reproduction is carried out, the fittest organism in the given population will be located and copied. The copy will then be added into the new generation. This means that this copy of the fittest organism is always reproduced and furthermore cannot be manipulated by crossover nor mutation, since it is added to the new *generation* (and a new generation is only formed *after* crossover and mutation are performed, according to the crossover and mutation parameters, on the organisms selected by reproduction). Note, however, that the fittest organism is not removed from the population that will undergo reproduction, crossover and mutation. This is done such that the manipulating operations could optimise the fittest organism, such that in the new generation, the fittest organism that was copied, might be no longer the fittest organism. Furthermore, since the fittest organisms is only copied and added to the new generation, and the original is still undergoing mutation and crossover, it is even expected that, for sufficient crossover and mutation parameters, this fittest organism will be crossed over and mutated, since the roulette-wheel reproduction is expected to make the share of the fitter organisms in the population grow, thereby increasing the probability that the fittest organism will be mutated and crossed over.

5.4.3 Chromosome Manipulation: Crossover and Mutation

When a fixed number organisms are selected by roulette-wheel reproduction, according to the bound on the generation size, some of the organisms undergo crossover and mutation. It is customary to first perform crossover and then mutation.[11] A crossover parameter, a real number from 0 to 1, determines which percentage of reproduced organisms will be crossed over. Since crossover is always performed on pairs, an even number of organisms on which crossover will be performed, needs to be calculated, given the crossover parameter and the fixed bound on the generation size. The number of organisms on which crossover will be performed, that is the size of the *crossover pool*, is:

$$\text{size crossover pool} = (c \cdot \rho_c) - [c \cdot \rho_c \mod 2] \text{ where}$$

- c is the fixed bound on the generation size;
- ρ_c is the crossover parameter;
- $[x \mod y]$, with $x, y \in \mathbb{R}$, is the remainder after division of x by y as a real number.

Since the GOPAM orders the chromosomes selected by reproduction in the order of the intervals in which the random numbers, created for roulette-wheel reproduction, fell, the first k occurring selected chromosomes, where k is the size of the crossover pool, can be selected for crossover. In this way it is guaranteed that a percentage, corresponding to the crossover parameter, of *randomly selected* chromosomes is crossed over.[12] Furthermore, by just pairing up the crossover pool (e.g. $\langle a, b, c, d\rangle$ becomes $\langle (a, b), (c, d)\rangle$) also random pairs of chromosomes are selected for crossover. Given a pair of chromosomes (A, B), the operation of crossover first determines a random integer r, s.t. $0 < r < |A| = |B|$, which is a cut-off point for splitting the chromosome (permutation) A. For the splitting of A a function *splitAt* is used, such that $splitAt(i, \overrightarrow{S})$ creates two lists, a list of items before $\overrightarrow{S}(i)$ ordered as in $\overrightarrow{S}$, and a list of items from $\overrightarrow{S}(i)$ onwards, ordered as in $\overrightarrow{S}$. For instance, $splitAt(1, \langle a, b, c\rangle) = (\langle a\rangle, \langle b, c\rangle)$. The first list created by *splitAt* on some list A will be called in the

[11] Mitchell (1996): pp. 8-9; Goldberg (1989): p. 12.
[12] cf. the remarks in Section 5.3 after Definition 5.2.

following, $L(A)$, the second list, $R(A)$. After the list A is split into $L(A)$ and $R(A)$, the crossover operation, will create two new lists C and D, s.t.:

- $C = L(A) + +B'$;
- $D = B'' + +R(A)$,

where B' are all students not occurring in $L(A)$ ordered as in B, and B'' are all students not occurring in $R(A)$ ordered as in B. The new permutations C and D so obtained by the crossover operation are the results of performing crossover on permutations A and B. More formally, the crossover operation is the following procedure:

input : A randomized list (population) $\overrightarrow{P}$ of permutations (chromosomes) $\overrightarrow{S}_i$, the crossover parameter ρ_c, the bound on the generation size c.
output: A list (population) $\overrightarrow{P}'$ of permutations (chromosomes) $\overrightarrow{S}_i$.

Let $\overrightarrow{P}' := \varnothing$;
Let $s := (c \cdot \rho_c) - [c \cdot \rho_c \mod 2]$;
Let $Pool := \iota(s, \overrightarrow{P})$;
Let $Pairs := pairUp(Pool)$, where
$pairup(\overrightarrow{X}) = \{(x, x') \mid (i, x), (i+1, x') \in \overrightarrow{X} \text{ and } [i \mod 2] = 0\}$;
for $(\overrightarrow{A}, \overrightarrow{B}) \in Pairs$ **do**
  Generate a random integer r, s.t. $0 < r < |\overrightarrow{A}|$;
  Let $(L(\overrightarrow{A}), R(\overrightarrow{A})) := splitAt(\overrightarrow{A}, r)$, where
  $splitAt(\overrightarrow{X}, i) = (\{(j, x) \in \overrightarrow{A} \mid j < i\}, \{(j, x) \in \overrightarrow{A} \mid j \geq i\}$;
  Let $\overrightarrow{C} := L(\overrightarrow{A}) + +\overrightarrow{B'}$, where
  $\overrightarrow{B'} = \{(i, b) \in \overrightarrow{B} \mid (i, b) \notin L(\overrightarrow{A})\}$ s.t.
  $\forall (i, b)(j, b') \in \overrightarrow{B'} : \overrightarrow{B}^{-1}(b) < \overrightarrow{B}^{-1}(b') \Rightarrow \overrightarrow{B'}^{-1}(b) < \overrightarrow{B'}^{-1}(b')$;
  Let $\overrightarrow{D} := \overrightarrow{B''} + +R(\overrightarrow{A})$, where
  $\overrightarrow{B''} = \{(i, b) \in \overrightarrow{B} \mid (i, b) \notin R(\overrightarrow{A})\}$ s.t.
  $\forall (i, b)(j, b') \in \overrightarrow{B''} : \overrightarrow{B}^{-1}(b) < \overrightarrow{B}^{-1}(b') \Rightarrow \overrightarrow{B''}^{-1}(b) < \overrightarrow{B''}^{-1}(b')$;
  Let $\overrightarrow{P}' := \overrightarrow{P}' + +\overrightarrow{C} + +\overrightarrow{D}$;
return *The list (population)* $\overrightarrow{P}'$.

It is immediate that the permutations $\overrightarrow{C}$ and $\overrightarrow{D}$ so produced will still be permutations of students if the permutations $\overrightarrow{A}$ and $\overrightarrow{B}$ are. Hence, the operation of crossover performed on two permutations $\overrightarrow{A}$ and $\overrightarrow{B}$ will produce organisms, corresponding to $\overrightarrow{C}$ and $\overrightarrow{D}$, that are Pareto-optimal allocations for the school market in question.

Mutation differs from crossover in that no '(mutation) pool' is generated before the operation is performed. Mutation is performed on each chromosome individually, that is, for each gene in the chromosome a biased coin is flipped to determine if the gene will be mutated, thereby mutating the chromosome. The bias of the coin is determined by the mutation parameter, which is some real number $\rho_m \in [0, 1]$. This means that if the mutation parameter is 0.25, then some *gene*, in some chromosome, is mutated with a probability of 0.25. More formally, the operation of mutation generates a random real number $r \leftarrow [0, 1]$ for each gene s s.t. if $r < \rho_m$, then s is mutated. A mutation of s will be an exchange of s with some student s' in the permutation in question (the respective chromosome) s.t. the index of s' is at least as low as the index of s. If the operation has 'flipped a coin' for each allele in the chromosome, then the operation of mutation is finished for that chromosome. More formally the operation of mutation, on some population is the following:

input : A list (population) $\overrightarrow{P} = \langle \overrightarrow{S}_0, \overrightarrow{S}_1, .., \overrightarrow{S}_c \rangle$ of permutations (chromosomes) $\overrightarrow{S}_i$, the mutation parameter ρ_m

output: A list (population) $\overrightarrow{P}' = \langle \overrightarrow{S}'_0, \overrightarrow{S}'_1, .., \overrightarrow{S}'_c \rangle$ of permutations (chromosomes) $\overrightarrow{S}_i$.

Let $\overrightarrow{P}' := \varnothing$;
for $0 \leq i \leq |\overrightarrow{P}| - 1$ **do**
  Let $\overrightarrow{S}'_i := \varnothing$;
  for $0 \leq j \leq |\overrightarrow{S}_i| - 1$ **do**
    Generate a random integer $r \leftarrow [0, 1]$;
    if $r \geq \rho_m$ **then**
      Let $\overrightarrow{S}'_i := \overrightarrow{S}'_i + +\langle \overrightarrow{S}_i(j) \rangle$;
    else
      Generate a random index k s.t. $0 \leq k \leq vect S_i^{-1}(j)$;
      Let $\overrightarrow{S}'_i := \overrightarrow{S}'_i \cup \{(j, \overrightarrow{S}'_i(k))\}$ (i.e. inserting the random student at the new position);
      Let $\overrightarrow{S}'_i := \overrightarrow{S}'_i \setminus \{(k, \overrightarrow{S}'_i(k))\}$ (i.e. removing the student from the old position);
      Let $\overrightarrow{S}'_i := \overrightarrow{S}'_i \cup \{(k, \overrightarrow{S}_i(k))\}$ (i.e. inserting the present student at the random position);
    Let $\overrightarrow{P}' := \overrightarrow{P}' + +\langle \overrightarrow{S}'_i \rangle$;
return *The list (population)* $\overrightarrow{P}'$.

Note that when some gene is mutated, in most cases that gene, some student s, is assigned a lower index in the resulting new permutation; the side effect of this is of course, that then also some other random student s', with whom s switches places, is assigned a higher index. The student who 'mutates' is never worse off, however, every mutation in which some student is better off (in terms of position in the permutation), will always yield some student who is worse off (in terms of position in the permutation).

5.4.4 GOPAM

With the three main operations of a genetic algorithm (reproduction, crossover and mutation) explained and introduced and one particular fitness function given, the GOPAM, the method for genetic-algorithmic optimisation of school allocations is fully stipulated. The GOPAM, given some school market, a bound on the number of generations, a bound on the generation size, a crossover and mutation parameter, can be called a school-allocation mechanism. Formally, it is not a mechanism in the sense of Definition 4.1, because the GOPAM does not map a school market to a single allocation, but to a set of allocations (a population). However given enough time, the GOPAM is expected to converge to a population consisting of multiple copies of one allocation, which makes it very similar to the formal *mechanisms* seen in Chapter 4. Furthermore, it is straightforward how a single school allocation could be selected from the population to which the GOPAM has converged: just take the fittest organism.

For all the mechanisms in Chapter 4, the PAM, BAM and DA-MTB, the formal properties of the allocations produced were studied. That is, it was investigated whether the allocations produced by those mechanism were Pareto optimal or not. The GOPAM, as indicated before, can only create Pareto-optimal allocations, since its chromosomes are permutations, which by the PAM correspond to Pareto-optimal allocations. The organisms to which the GOPAM converges are thus necessarily Pareto-optimal allocations.

The game-theoretic properties of the GOPAM are not as easily found as the nature of the allocations it produces. The randomness occurring in every generation and every operation of the GOPAM makes it very difficult to say whether the GOPAM is strategy proof or even coalition strategy proof. If a student s could, by handing in some 'strategic' preference list Q_s, force the GOPAM to converge to some allocation, wherein s is assigned a better school, then he would be, had he handed in his actual true preferences P_s, then the GOPAM is clearly *not* strategy proof. It is not clear if the latter is possible. It is an open question whether

the GOPAM is strategy proof or even coalition strategy proof.

Multiple 'strategic actions' could be taken by the student. Not only can he order his schools differently in his 'strategic' preference list, the student can also decide to hand in a preference list with a certain length. The GOPAM converges to certain allocations based on the fitness of those allocations. The fitness function f favours allocations in which (i) more students are assigned their first choice and (ii) less students are not-allocated. To steer this convergence, a student s must be able to hand in a 'strategic' preference list Q_s, which could do one of the following:

(a) allocations based on Q_s tend to have a higher fitness value than allocations based on P_s and in allocations based on Q_s, student s is assigned a better school than in allocations based on P_s;

(b) handing in Q_s tends to benefit allocations where s is better off after crossover;

(c) handing in Q_s results in a higher probability for s of being mutated;

(d) handing in a shorter or longer preference list Q_s than P_s results in allocations in which s is better off.

It is not clear what manipulation of the actual preference list, to obtain some 'strategic' preference list, could result in either (a), (b) or (c). However, it seems that the present fitness function f tends to benefit shorter preference lists, since the fitness function will assign lower fitness values to allocations in which more students are not allocated. The shorter the student's preference list, the more likely it is that it will result in the student not being allocated, thereby lowering the fitness value of allocations in which he is not-allocated, which will also entail that allocations in which he is assigned the higher preference positions, with non-null schools, will receive higher fitness values. This shortcoming could be easily overcome by demanding preference lists to have a certain length, whether this is admissible is unclear. Furthermore, note that the game-theoretic properties of the GOPAM depend on the fitness function used, for instance result (c) can also be easily overcome by deleting the penalty for not allocated students in the fitness function. It could therefore very well be that only certain fitness functions are (coalition) strategy proof or de facto (coalition) strategy proof while others aren't.

Another fitness function that immediately come to mind is the fitness function f', which is just as f, but now the second understanding of the second 'starting point', that students should only not be allocated to their highest preferred school, if it is not possible to allocate all students who prefer that school highest to that school, is incorporated in the fitness function. Say some allocation A is *non-maximal*, if and only if, there is some school $h \in \mathcal{H}$ s.t. the number of students who place h highest on their preference list is higher than c_h and not all students assigned to h in A place h highest on their preference list. The property of some allocation being non-maximal then captures the second understanding of the second starting point. This property can be translated into a penalty similarly to the penalty for not-allocated students by multiplying the result of the fitness function f by 1 divided by the number of *non-maximal students* plus 1. Non-maximal students are students assigned to some school h, who do not place h highest on their preference list and h is such that the number of students who place h highest on their preference list is higher than c_h. The fitness function f' so obtained would capture both understandings of the municipality official for education's second starting point.

Another fitness function that seems to make the GOPAM converge to interesting allocations, is the fitness function f'' which assigns higher fitness values to allocations with a lower mean preference position. Note that the mean preference position of some allocation $\mu(A)$ for some given school market $\mathcal{M} = (\mathcal{H}, \mathcal{S}, \mathcal{C}, \mathcal{P})$ is

$$\mu(A) = \frac{\sum_{s \in \mathcal{S}} 1 + P_s^{-1}(A(s))}{|\mathcal{S}|}$$

This fitness function can be adjusted to have a penalty for not-allocated students or not, just as the fitness function that assigns higher fitness values for allocations which assign more students their first choice. The fitness function f'', capturing *the lower mean preference position the fitter* and *the less students not allocated the fitter*, is then:

$$f''(A) = \left(\frac{1}{g(A) + 1} \cdot h'(A) \right)^2 \text{ where}$$

- A is some allocation;
- $g(A)$ is the number of students not allocated in A, i.e. the number of students assigned to the null school;
- $h'(A) = \sum_{s \in \mathcal{S}} |\mathcal{H}| - \left(P_s^{-1}(A(s)) + 1\right)$, s.t. $h'(A) > h'(A')$ iff $\mu(A) < \mu(A')$.

Reasonable other options for fitness functions could be: the less low preference positions the fitter, more popular allocations are fitter, or better preference profile then fitter. An allocation A is said to be more popular than some allocation A', if more students prefer A over A' than the other way around; an allocation's preference profile 'dominates' some other allocation's preference profile, if it assigns more students their first choice, second choice, third choice, etc. All these options are easily programmable, and the GOPAM can thus easily be adjusted to converge to either of these other fitness conditions or to combinations of those. The upside of using a genetic approach is that policy makers always have the opportunity to decide every year anew which fitness conditions they consider just. As indicated before, the 'strategy proofness' of the GOPAM depends heavily on the fitness function chosen, hence before simply deciding upon another fitness function, the game-theoretic consequences need to be investigated.

5.5 Simulations with the GOPAM

With the GOPAM fully introduced and variants of fitness functions discussed, it remains to be shown what kind of allocations the GOPAM, with certain fitness functions, produces. As was tested for the other mechanisms, also for the GOPAM with certain fitness functions, it will be shown what are the number of students not allocated, how many students are assigned their first choice and what is the mean preference position, but now for the fittest allocation to which the GOPAM converges. The GOPAM will as well be tested on the VWO population, such that its results can be compared to the other mechanisms. The GOPAM has been ran with the following fitness functions:

F_1 : the more students assigned first choice, the fitter;

F_2 : the more students assigned first choice and the less students not allocated, the fitter;

F_3 : the lower the mean preference position, the fitter;

F_4 : the lower the mean preference positions and the less students not allocated, the fitter;

F_5 : the more students assigned first choice, the less non-maximal students, and the less students not allocated, the fitter.

For each fitness function the GOPAM will be ran for 100 generations with an initial population of 30 random generated chromosomes (permutations), with a bound on the generation size of 30 as well. The crossover parameter is 0.5, which means that 14 permutations will crossed over in each generation. The mutation parameter is 0.00025, which means that about $(0.00025 \cdot 1885) \cdot 2 = 0,9425 \approx 1$ student(s) per allocation are expected to be switched. The seed used in all runs is zero. This gives the following results:[13]

	Not Allocated	First Choice	Mean Pref. Pos.	Par. Cons.	Coa. Strat. Proof
F_1:	9 (0.48 %)	1581 (83.87 %)	1.2809	Yes	?
F_2:	0 (0.00 %)	1566 (83.08 %)	1.2928	Yes	?
F_3:	4 (0.21 %)	1572 (83.40 %)	1.2632	Yes	?
F_4:	1 (0.05 %)	1567 (83.13 %)	1.2633	Yes	?
F_5:	0 (0.00 %)	1559 (82.71 %)	1.2976	Yes	?
PAM:	6.93 (0.36 %)	1566.05 (83.08 %)	1.2970	Yes	Yes
BAM:	8.13 (0.43 %)	1636.00 (86.79 %)	1.2606	Yes	No
DA-MTB:	0.38 (0.02 %)	1379.64 (73.19 %)	1.3545	No	Yes

[13]The average running time for the GOPAM with the different fitness functions is about one hour. The GOPAM was tested on a commercially available laptop with a dual-core quadruple-thread 2.50 GHz processor.

From the table above it is clear that the GOPAM makes it possible to optimise the PAM in such a way that the number of first choice students stays high, the mean preference position stays low, but with allocating as much students as possible. If no penalty is given for not-allocated students, then the GOPAM really outperforms the PAM in the number of first choice students (in the case of F_1) and in the mean preference position (in the case of F_3).

5.6 Genetic-Algorithmic Optimisation for Allocation Mechanisms

In this chapter a genetic-algorithmic optimisation for the PAM was introduced and studied. To say if the resulting mechanism, the GOPAM, can be interesting for the Amsterdam school-allocation problem it must be studied more rigorously, since it is still unclear what the exact game-theoretic properties are of the GOPAM with different fitness functions. This uncertainty makes it hard to say if the GOPAM meets all the 'starting points' put forward by the municipality official for education, but it can nevertheless be shown that the GOPAM meets most.

It can be said that the GOPAM meets the first starting point of the municipality official, since it can be adjusted to respect certain aspects of the preferences of the student population. For instance, it was demonstrated that the GOPAM can be adjusted to converge to some allocation in which most students are assigned their first choice, or to some allocation in which the least possible number of non-maximal students are present. Furthermore, it is possible to alter the GOPAM by introducing a 'mean-preference' fitness function as to make the GOPAM converge to some allocation which benefits the student population as a whole. From this it can be concluded that, the GOPAM respects the students' preferences in some way and thus meets the first 'starting point'. However, with important game-theoretic properties of the different GOPAM versions unknown, it remains difficult to say if the GOPAM really respects the preferences of the students in every possible way.

The second starting point could be understood in different ways (cf. the three interpretations in Section 3.5). From the simulations and discussion on the different fitness function for the GOPAM it is clear that the GOPAM meets the second starting point in each of the three understandings. The GOPAM can converge to the allocation in which most students are assigned their first choice (covering the first understanding), it can converge to the allocation in which the least number of non-maximal students are present (covering the second understanding) and it was shown to place more students on their first choice than the DA-MTB is expected to do in Section 5.5 (covering the third understanding). The design of the GOPAM also ensures that all versions of the mechanism satisfy the third starting point, since it always produces Pareto-optimal allocations. This means that it is impossible for there to be reasonable trade options for students based on their handed-in preferences (cf. Section 3.5). Moreover, all versions of the GOPAM are fast and easily ran on commercially available computers, which makes it hard to see how the GOPAM could not meet the fifth starting point.

The fourth starting point mentioned by the municipality is the most unclear and it is therefore hard to judge if the GOPAM meets this demand. The municipality official desires a school-allocation mechanism of which the workings are clear and transparent to parents, students and schools. The only point of reference to judge if the GOPAM satisfies this demand, is that the DA-STB, in the way used by Gautier et al. (2015b), is considered clear and transparent. Note that the DA-STB as in Gautier et al. (2015b), is what is considered in this thesis as a DA-MTB (since not all high schools have the same preferences). This means that the DA-STB from Gautier et al. (2015b) is *not* reducible to the PAM, which makes the mechanisms not at all straightforward. If *that* mechanism is considered clear and transparent, it can be said that the GOPAM might also be. The GOPAM viewed as an optimisation method that just tries to locate the Pareto-optimal allocation that satisfies certain further conditions, is easily explained and understood. The exact workings of the GOPAM are probably less clear, but are nevertheless transparent and explainable. In short, it is hard to say if the GOPAM meets the fourth starting condition, but it is definitely also not clear that it does not do so, since also the mechanism from Definition 4.16 is considered clear and transparent.

This thesis explores a new route for school-allocation mechanisms, in that it tries to see if genetic optimisation can be interesting to find better allocations for school markets. The first test results look very promising and the genetic optimisation method is so adaptive that it can be easily adjusted to meet certain conditions not yet investigated here. However, there remain many open questions which need to be answered before this method could ever be used as a school-allocation mechanism. The most prominent question is whether the GOPAM with the fitness functions considered in this thesis is (coalition) strategy proof or *de facto* coalition strategy proof. Furthermore, it is interesting to study how long the preferences lists of the students need to be. From the research by Cohen et al. (2012) and the satisfaction survey of Panel Inzicht (2015), it became clear that parents and students prefer to hand in short preference lists. Since the GOPAM is an optimisation method, it can maybe be adjusted in such a way that it can cope with very short preference lists, which would definitely be very well received by the population of Amsterdam.

In the foregoing all school allocations used in Amsterdam have been discussed and a new mechanism, the GOPAM has been introduced. For every mechanism important formal mathematical properties have been explained and demonstrated, and it is shown in what way each mechanism meets the starting points of the municipality official. The starting points are not all easily translatable to the mathematical setting, which makes it hard to say precisely which mechanism comes closest to what the municipality of Amsterdam desires. It is the hope of the author that this thesis not only shines light on a possible new approach in solving the Amsterdam allocation problem by using genetic algorithms, but also helps policy makers in making the 'starting points' more formal, as to guide future research on school-allocations mechanisms in the city of Amsterdam.

Chapter 6

Bibliography

Abdulkadiroğlu, A., **Pathak**, P.A. & **Roth**, A.E. (2009), *"Strategy-proofness versus Efficiency in Matching with Indifferences: Redesigning the NYC High School Match"*, American Economic Review, Vol. 99(5), pp. 1954-1978.

Abdulkadiroğlu, A., **Pathak**, P.A., **Roth**, A.E. & **Sönmez**, T. (2006), *"Changing the Boston School Choice Mechanism: Strategy-proofness as Equal Access"*, Study on the previously used school allocation mechanism used by Boston Public Schools, Unpublished, URL: `http://seii.mit.edu/wp-content/uploads/2012/04/Boston-Mechanism.pdf`, Retrieved June 14, 2016.

Abdulkadiroğlu, A., **Pathak**, P.A., **Roth**, A.E. & **Sönmez**, T. (2005), *"The Boston Public School Match"*, American Economic Review, Vol. 95(2), pp. 368-371.

Abdulkadiroğlu, A. & **Sönmez**, T. (2003), *"School Choice: A Mechanism Design Approach"*, American Economic Review, Vol. 93(3), pp. 729-747.

Abdulkadiroğlu, A. & **Sönmez**, T. (1999), *"House Allocation with Existing Tenants"*, Journal of Economic Theory, Vol. 88(2), pp. 233-260.

Abdulkadiroğlu, A. & **Sönmez**, T. (1998), *"Random Serial Dictatorship and the Core from Random Endowments in House Allocation Problems"*, Econometrica, Vol. 66(3), pp. 689-701.

Carnap, R. (1967), *"Logical Foundations of Probability"*, Third Impression, University of Chicago Press.

Cohen, L., **De Jong**, I., **Jakobs**, E. & **Slot**, J. (2012), *"Berlage of Barlaeus?"*, Department for Research, Information and Statistics for the Minicipality of Amsterdam (NL: Onderzoek, Informatie en Statistiek van de Gemeente Amsterdam), URL: `http://www.ois.amsterdam.nl/pdf/2012_berlage%20of%20barlaeus.pdf`, Retrieved June 14, 2016.

Dubins, L. E. & **Freedman**, D. A. (1981), *"Machiavelli and the Gale-Shapley Algorithm"*, The American Mathematical Monthly, Vol. 88(7), pp. 485-494.

Gale, D. & **Shapley**, L. S. (1962), *"College Admissions and the Stability of Marriage"*, The American Mathematical Monthly, Vol. 69(1), pp. 9-15.

Gautier, P.A., De **Haan**, M., Van der **Klaauw**, B. & P.A., **Oosterbeek** (2016), *"Eerste Analyse Matching en Loting Voortgezet Onderwijs Amsterdam 2016*"*, Evaluation Paper, URL: `http://www.verenigingosvo.nl/wp-content/uploads/2013/09/Rapport-loting-en-matching-2016-beschrijvend-rapport.pdf`, Retrieved June 14, 2016.

Gautier, P.A., De **Haan**, M., Van der **Klaauw**, B. & **Oosterbeek**, H. (2015), *"Eerste Analyse Matching en Loting Voortgezet Onderwijs Amsterdam 2015*"*, Evaluation Paper, URL: `https://sites.google.com/site/pietgautier/Evaluatie_Matching_%28beschrijvend_rapport%29.pdf?attredirects=0&d=1`, Retrieved June 14, 2016.

Gautier, P.A., De **Haan**, M., Van der **Klaauw**, B. & **Oosterbeek**, H. (2015b), *"Simulatie Analyse Matching en Plaatsing Voortgezet Onderwijs Amsterdam 2015"*, URL: https://sites.google.com/site/pietgautier/Evaluatie_Matching_%28simulaties%29.pdf?attredirects=0&d=1, Retrieved June 14, 2016.

Goldberg , D.E. (1989), *"Genetic Algorithms in Search, Optimization, and Machine Learning"*, Addison-Wesley Publishing Company.

De **Haan**, M., **Gautier**, P.A., **Oosterbeek**, H. & Van der **Klaauw**, B., (2015), *"The performance of school assignment mechanisms in practice"*, Discussion Paper, URL: https://sites.google.com/site/pietgautier/schoolchoice2015may.pdf?attredirects=0&d=1, Retrieved June 14, 2016.

Hylland, B. & **Zeckhauser**, R. (1979), **"The Efficient Allocation of Individuals to Positions"**, The Journal of Political Economy, Vol. 87(2), p. 293.

De **Jong**, K. (1975), *"An Analysis of the Behavior of a Class of Genetic Adaptive Systems"*, Ph.D. thesis, University of Michigan, Ann Arbor.

Kagel, J.H. & **Roth**, A.E. (2000), *"The Dynamics of Reorganization in Matching Markets: A Laboratory Experiment Motivated by a Natural Experiment"*, The Quarterly Journal of Economics, Vol. 115(1), pp. 201-235.

Kukenheim, S. (2016), *"Voortgang Kernprocedure 2016"*, Letter to the city council of Amsterdam on proceedings of the 'Kernprocedure 2016', URL: http://zoeken.amsterdam.raadsinformatie.nl/cgi-bin/showdoc.cgi/action=view/id=287932/type=pdf/1_brief_Voortgang_Kernprocedure_2016.pdf, Retrieved June 14, 2016.

Kukenheim, S. (2015), *"Uitkomst Evaluatie en Tevredenheidsonderzoek Matching PO-VO"*, Official Reply from Municipality Official for Education, URL: http://www.stichtingvsa.nl/pdf/20151006/bijlage_1_20151005_WH_brief_evaluatie_matching_PO-VO.pdf, Retrieved June 14, 2016.

Leyton-Brown, K. & **Shoham**, Y. (2008), *"Essentials of Game Theory: A concise, Multidisciplinary Introduction"*, Synthesis Lectures on Artificial Intelligence And Machine Learning # 3, Morgan & Claypool.

Manlove, D. F. (2013), *"Algorithmics of Matching under Preferences"*, Series on Theoretical Computer Science Vol. 2, World Scientific Publishing.

Mitchell, M. (1996), *"An Introduction to Genetic Algorithms"*, MIT Press.

OSVO (2016), *"Persbericht 7 April 2016"*, URL: http://www.verenigingosvo.nl/wp-content/uploads/2016/04/Persbericht-Kernprocedure-VO-OSVO-II.pdf, Press Release, Retrieved June 14, 2016.

OSVO (2016), *"Jaarlijkse Matching Leerlingen Groep 8 naar Amsterdams Voortgezet Onderwijs is Afgerond"*, Press Release, URL: http://www.verenigingosvo.nl/wp-content/uploads/2013/09/16053-ALG-Persbericht-Kernprocedure-VO-tweede-matching-OSVO-DEF-150416.pdf, Retrieved June 14, 2016.

OSVO & **Gemeente Amsterdam** (2015), *"Kernprocedure Overstap PO-VO Amsterdam 2016"*, URL: http://www.verenigingosvo.nl/wp-content/uploads/2015/10/Kernprocedure-2015-2016.pdf, Retrieved June 14, 2016.

OSVO & **Gemeente Amsterdam** (2014), *"Kernprocedure Overstap PO-VO Amsterdam Schooljaar 2014-2015"*, URL: http://www.onderwijsconsument.nl/wordpress/wp-content/uploads/kernprocedure-2014-2015.pdf, Retrieved June 14, 2016.

OSVO & **Gemeente Amsterdam** (2013), *"Kernprocedure Overstap PO-VO 2013-2014"*, URL: http://www.barlaeus.nl/wp-content/uploads/2014/02/kernprocedure-2013-2014.pdf, Retrieved June 14, 2016.

Panel Inzicht (2015), "Rapport Tevredenheidsonderzoek Matching", Report on Satisfaction Survey of DA-MTB commissioned by the OSVO, URL: http://www.verenigingosvo.nl/wp-content/uploads/2015/10/Eindrapport-tevredenheidsonderzoek-matching-2015.pdf, Retrieved June 14, 2016.

Rechtbank Amsterdam (2015), *"Amsterdamse plaatsingssysteem middelbare scholen met ruilverbod houdt stand."*, Court Ruling Number: ECLI:NL:RBAMS:2015:4085, URL: http://uitspraken.rechtspraak.nl/inziendocument?id=ECLI:NL:RBAMS:2015:4085, Retrieved June 14, 2016.

Roth, A.E. (1990), *"New Physicians: A Natural Experiment in Market Organization"*, Science, Vol. 250(4987), pp. 1524-1528.

Roth, A.E. (1991), *"A Natural Experiment in the Organization of Entry-Level Labor Markets: Regional Markets for New Physicians and Surgeons in the United Kingdom"*, The American Economic Review, Vol. 81(3), pp. 415-440.

Roth, A.E. & **Sotomayor**, M. A. Oliveira (1990), *"Two-sided matching: a study in game-theoretic modeling and analysis"*, Econometric Society Monographs, No. 18, Cambridge University Press.

Svensson, L.-G. (1999), *"Strategy-proof allocation of indivisible goods"*, Social Choice and Welfare, Vol. 16(4), pp. 557-567.

Svensson, L.-G. (1994), *"Queue allocation of indivisible goods"*, Social Choice and Welfare, Vol. 11(4), pp. 323-330.

Zhou, L. (1990), *"On a conjecture by Gale about one-sided matching problems"*, Journal of Economic Theory, Vol. 52(1), pp. 123-135.

Chapter 7

Appendix: Computational Implementations

7.1 Haskell

In the following the computational implementations of all the school-allocation mechanisms tested in this thesis will be presented. All mechanisms were implemented using the programming language Haskell (Version 2010). Haskell is a general-purpose purely functional programming language, with non-strict semantics and strong static typing. The code was ran using the Glasgow Haskell Compiler on a Windows computer using GHCi, version 8.0.1. The code will be presented non verbosely and is attached for completeness.

7.1.1 Imported Modules and Introduced Types

```
import System.Random
import Data.List
import Data.Ord (comparing)

-- Types --
type Seed         = Int
type Student      = Int
type School       = Int
type Frequency    = Int
type Allocation   = [(School,[Student])]
type Permutation  = [Student]
type Highschools  = [(School,Frequency)]
type Preferences  = [(Int,[Int])]
type Organism     = (Allocation, Permutation)
type Population   = [((Allocation, Permutation), Frequency)]
```

7.1.2 General Help Functions

```
-- The function allocationAdd updates a school assignment in some allocation by
--  simply adding the student to the assignment.
allocationAdd :: Allocation -> Student -> School -> Allocation
allocationAdd allocation studentIn school = let
    oldschoolassignment = lookup' school allocation
    newschoolassignment = studentIn : oldschoolassignment
   in (school,newschoolassignment) : delete (school,oldschoolassignment) allocation
```

```
-- The function allocationAddStuds adds multiple students to some high school in a
--  given allocation.
allocationAddStuds allocation [] _ = allocation
allocationAddStuds allocation (stud:studs) school =
    allocationAddStuds (allocationAdd allocation stud school )
        studs school

-- The function allocationUpdate adds studentIn to the assignment of some school in an
--  allocation and removes studentOut from that assignment.
allocationUpdate :: Allocation -> Student -> Student -> School -> Allocation
allocationUpdate allocation studentIn studentOut school = let
    Just oldschoolassignment = lookup school allocation
    newschoolassignment = studentIn : delete studentOut oldschoolassignment
   in (school,newschoolassignment) : delete (school,oldschoolassignment) allocation

-- The function exchange takes a list xs and two indices i and j and exchanges xs[i]
--  and xs[j] in xs.
exchange :: Eq a => [a] -> Int -> Int -> [a]
exchange xs i j = let
    elemj = xs !! j
    elemi = xs !! i
    spliti = splitAt i xs
    addj = ((fst spliti) ++ [elemj] ++ (delete (xs !! i) (snd spliti)))
    splitj = splitAt j addj
   in (fst splitj) ++ [elemi] ++ (delete (xs !! j) (snd splitj))

-- The function generatePermutation generates a permutation of a list based on some
--  seed, according to the Fisher Yates Shuffle, first implemented by Durstenfeld.
generatePermutation :: Eq t => Seed -> [t] -> [t]
generatePermutation _ [] = []
generatePermutation _ [a] = [a]
generatePermutation seed xs = let
    len = length xs
    seeds = map (`mod` 10000) (randoms $ mkStdGen seed :: [Int])
   in fisherYates seeds (len-1) xs where
       fisherYates _     0 ys = ys
       fisherYates (s:sds) i ys =
           fisherYates sds (i-1)
               (exchange ys i (fromIntegral $ fst (random (mkStdGen s)) `mod` (i+1)))

-- Alternative version of lookup for lists (without Maybe).
lookup' :: Eq a => a -> [(a, [t])] -> [t]
lookup' _ [] = []
lookup' k ((x,y):rest)
    | k == x = y
    | otherwise = lookup' k rest

-- Lookup function for snd element of a pair, returns its first element if existing.
lookupsnd :: Eq b => b -> [(a,b)] -> Maybe a
lookupsnd _key [] = Nothing
lookupsnd key ((x,y):xys)
          | key == y    = Just x
          | otherwise   = lookupsnd key xys
```

7.2 School Allocation Mechanisms

7.2.1 Permutation Allocation Mechanism

```
-- The function permutationAllocationMechanism performs the PAM.
permutationAllocationMechanism :: Seed
                                -- seed
                               -> Highschools
                                -- list of schools and their capacity
                               -> Preferences
                                -- preferencelists for all students
                               -> (Allocation,Permutation)
                                -- Pair of an allocation produced by PAM on the
                                --  permutation
permutationAllocationMechanism seed highschools prefs = let
    studs         = map fst prefs
    permutation   = generatePermutation seed studs
   in (sortBy (comparing fst)
            (allocationByPermutation permutation prefs highschools),
       permutation)

-- The function allocationByPermutation generates an allocation, given some permutation.
--  using the PAM method
allocationByPermutation :: Permutation -> Preferences -> Highschools -> Allocation
allocationByPermutation permutation prefs highschools = let
    startalloc = map (\(s,c) -> (s,[])) highschools
   in allocationByPermutation' permutation prefs highschools startalloc
    where
        allocationByPermutation' [] _ _ allocation = allocation
        allocationByPermutation' (stud:studs) prefs highschools allocation =
            allocationByPermutation' studs prefs highschools
                (highestLeftover stud prefs highschools allocation)


-- The function highestLeftover places a student in the highest preferred school that
--  is still available in some given allocation.
highestLeftover :: Student -> Preferences -> Highschools -> Allocation -> Allocation
highestLeftover student prefs highschools allocation = let
   prefstud  = lookup' student prefs
  in highestLeftover' prefstud highschools allocation where
        highestLeftover' [] _ _ = error "No Highest Leftover Found"
        highestLeftover' (school:rest) highschools allocation = let
            Just capacity = lookup school highschools
            allocated    = lookup' school allocation
           in if (capacity - (length allocated)) > 0
                then allocationAdd allocation student school
                else highestLeftover' rest highschools allocation
```

7.2.2 Boston Allocation Mechanism

```
-- The function boston performs the Boston Allocation Mechanism.
boston :: Seed -> Highschools -> Preferences -> Allocation
boston seed highschools studprefs = let
    studs  = map fst studprefs
    seeds  = map (`mod` 10000) (randoms $ mkStdGen seed :: [Int])
   in sortBy (comparing fst) $ performBoston seeds studprefs highschools 0 ([], studs)
    where
        performBoston _ _ _ _ (allocation, []) = allocation
        performBoston (seed:seeds) prefs highschools index
                      (allocation, notallocated) = let
            studs        = map fst prefs
            schools      = map fst highschools
            round        = findstuds notallocated prefs index schools schools
            newallocation = bostonRound seed allocation round schools
            notallocated' = studs \\ (concat (map snd newallocation))
           in performBoston seeds prefs highschools (index+1)
                        (newallocation, notallocated')
            where
                findstuds notallocated preferences index schools [] = []
                findstuds notallocated preferences index schools (sch:schs) = let
                    prefs     = filter (\(stud,prefs) -> stud `elem` notallocated)
                                preferences
                    candidates = map fst $
                                 filter (\ (stud,prefs) ->
                                     (prefs !! index) == sch ) prefs
                   in (sch, candidates) : findstuds notallocated preferences index
                       schools schs

                bostonRound seed allocation round schools = let
                    lotteryseeds = map (`mod` 10000) (randoms $ mkStdGen seed :: [Int])
                   in bostonRound' lotteryseeds allocation round schools
                    where
                        bostonRound' (seed:seeds) allocation round [] = allocation
                        bostonRound' (seed:seeds) allocation round (school:schools) =
                           let
                            candidates     = lookup' school round
                            Just capacity  = lookup school highschools
                            atschool       = length $ lookup' school allocation
                            seatsleft      = capacity - atschool
                           in if (length candidates) <= seatsleft
                               then bostonRound' seeds
                                     (allocationAddStuds allocation candidates school)
                                      round schools
                               else let
                                      lottery = generatePermutation seed candidates
                                      winners = take seatsleft lottery
                                     in bostonRound' seeds
                                         (allocationAddStuds allocation winners
                                             school)
                                                 round schools
```

7.2.3 Deferred Acceptance

```
-- The function deferredAcceptance performs the DA-MTB.
deferredAcceptance :: Seed -> Highschools -> Preferences -> Allocation
deferredAcceptance seed highschools studprefs = let
    students   = map fst studprefs
    schools    = map fst schoolprefs
    schoolprefs = generateSchoolpreferencesMTB seed highschools students
    allocation = zip schools $ replicate (length highschools) []
   in sortBy (comparing fst) $
            (\(a,b,c) -> c) $
            daRecursion highschools schoolprefs (students, studprefs, allocation)
    where
        generateSchoolpreferencesMTB :: Seed -> Highschools -> [Student] -> Preferences
        generateSchoolpreferencesMTB seed highschools students = let
            schools    = map fst highschools
            seeds      = generateSeeds seed $ length schools
            lotteries  = generatePermutations seeds students
           in zip schools lotteries where
               generatePermutations [] _ = []
               generatePermutations (sd:sds) studs = generatePermutation sd studs :
                                                     generatePermutations sds studs

-- The function daRecursion performs the recursive step over the list of students
--  in the DA algorithm.
daRecursion ::   Highschools
              -> Preferences
              -> ([Student], Preferences, Allocation)
              -> ([Student], Preferences, Allocation)
daRecursion highschools schoolprefs ([],studprefs,allocation) =
    ([],studprefs,allocation)
daRecursion highschools schoolprefs ( (stud:studs), studprefs, allocation) =
 let
    Just studpref   = lookup stud studprefs
    topschool       = head studpref
    Just schoolpref = lookup topschool schoolprefs
    Just enrolled   = lookup topschool allocation
    prefposlist     = map (\x -> (x, head $ elemIndices x schoolpref)) enrolled
    studprefpos     = head $ elemIndices stud schoolpref
   in if capacitycheck enrolled topschool highschools
       then daRecursion highschools schoolprefs
              (   studs,
                  studprefs,
                  allocationAdd allocation stud topschool )
       else let
        Just lowestranked = lookupsnd (maximum $ map snd prefposlist) prefposlist
       in if studprefpos < (maximum $ map snd prefposlist)
              then daRecursion highschools schoolprefs
                     (   lowestranked : studs,
                         preferenceUpdate studprefs lowestranked topschool,
                         allocationUpdate allocation stud lowestranked topschool)
              else daRecursion highschools schoolprefs
                     (   stud:studs,
                         preferenceUpdate studprefs stud topschool,
                         allocation)

       where
```

```
-- The function capacitycheck returns True if some school has seats left
capacitycheck :: [Student] -> School -> Highschools -> Bool
capacitycheck students school highschools = let
    Just capacity = lookup school highschools
   in length students < capacity

-- The function preferenceUpdate removes a school from a student's preference list.
preferenceUpdate :: Preferences -> Student -> School -> Preferences
preferenceUpdate studprefs student schoulOut = let
    Just oldprefs = lookup student studprefs
   in (student, delete schoulOut oldprefs) :
           delete (student, oldprefs) studprefs
```

7.3 GOPAM

7.3.1 Help Functions

```
-- cumList converts a list of probabilities (summing up to 1) into a list of
--  intervals corresponding to those probabilities.
cumList :: Fractional t => [t] -> [(t, t)]
cumList [] = []
cumList (p:rest) = (0.0,p) : cumList' p rest where
    cumList' _ [] = []
    cumList' p (q:rest) = (p,p+q) : cumList' (p+q) rest

-- Generalisation of 'div' to any instance of Real (from Data.Fixed).
div' :: (Real a,Integral b) => a -> a -> b
div' n d = floor ((toRational n) / (toRational d))

-- The function generateseeds generates random numbers based on some seed.
generateSeeds :: Seed -> Int -> [Int]
generateSeeds seed k = map (`mod` 10000) (take k (randoms $ mkStdGen seed :: [Int]))

-- The function freqRep represents a population of individual organisms as pairs.
-- (organism, frequency) similar to the type Population.
freqRep :: [Organism] -> Population
freqRep [] = []
freqRep (org:rest) = (org, (count org rest) + 1) :
                  freqRep (filter (\x -> x /= org) rest)
                    where
                        count _ [] = 0
                        count x (y:ys)
                                  | x == y = 1 + (count x ys)
                                  | otherwise = count x ys

-- Generalisation of 'mod' to any instance of Real (from Data.Fixed).
mod' :: (Real a) => a -> a -> a
mod' n d = n - (fromInteger f) * d where
    f = div' n d

-- The function nelem is the dual of elem.
nelem :: (Eq a, Foldable t) => a -> t a -> Bool
nelem x y = not (x `elem` y)

-- The function normalize makes a list of numbers sum up to one and keeps the
--  proportions intact.
```

```
normalize :: Fractional b => [b] -> [b]
normalize xs = let
    sumlist = sum xs
    coefficient = (1 / sumlist)
   in map (\x -> x * coefficient) xs

-- pairUp takes a list and pairs up every two elements.
pairUp :: [a] -> [(a,a)]
pairUp [] = []
pairUp [_] = error "pairUp ERROR: not an even number"
pairUp (x:y:xs) = (x,y) : pairUp xs

-- The function randIndex returns a random index from a list with length length.
randIndex :: (Integral a, Random a) => Int -> a -> a
randIndex seed length = if length == 0
                        then error "length of the list for the random index is 0"
                        else (fst $ random (mkStdGen seed)) `mod` length

-- randomRange takes a seed, an interval and a quantity n and returns n values
--  uniformly picked from the interval using the seed.
randomRange :: Random a => Seed -> (a, a) -> Int -> [a]
randomRange seed (a,b) k = take k $ randomRs (a,b) (mkStdGen seed)

-- tripleUp takes a list and pairs up every three elements
tripleUp :: [a] -> [(a,a,a)]
tripleUp [] = []
tripleUp [_] = error "tripleUp ERROR: not a multiple of three"
tripleUp [_,_] = error "tripleUp ERROR: not a multiple of three"
tripleUp (x:y:z:xs) = (x,y,z) : tripleUp xs
```

7.3.2 The GOPAM

```
-- The function createSample generates an initial (or sample) population for the
--  genetic algorithm, in casu this is a list of pairs of allocations produced by the
--  PAM and the permutation used to produce this allocation.
createSample :: Seed -> Int -> Preferences -> Highschools -> [Organism]
createSample seed size prefs highschools = let
    seeds = generateSeeds seed size
   in createSample' seeds highschools prefs where
       createSample' [] _ _ = []
       createSample' (s:rest) highschools prefs =
           permutationAllocationMechanism s highschools prefs :
           createSample' rest highschools prefs

-- Function Evolve
-- The function evolve preforms the GOPAM.
evolve ::  Seed
          -- seed
        -> Highschools
          -- list of schools and their capacity
        -> Preferences
          -- preferencelists for all students
        -> Int
          -- bound on generation size
        -> Int
          -- maximum number of generations
        -> Float
```

```
          -- crossover parameter
        -> Float
          -- mutation parameter
        -> Population
          -- Population after evolution
evolve seed highschools prefs gensize nrgen cpar mpar = let
       primeseeds = generateSeeds seed 2
       seedpop    = primeseeds !! 0
       seedgen    = primeseeds !! 1
       pop        = freqRep $ createSample seedpop gensize prefs highschools
       seedsgen   = generateSeeds seedgen nrgen
   in evolve' seedsgen prefs pop nrgen where
       evolve' _ _ pop 0 = pop
       evolve' (s:seeds) prefs pop k =
         evolve' seeds prefs (createGen s highschools gensize cpar mpar prefs pop)
                (k-1)

-- The function createGen creates a new generation from a given population, by
--  performing reproduction, mutation and crossover. The actual elitism takes
--  place here, in the sense that the best reproduced organism cannot crossover
--  nor mutate.
createGen ::  Seed
            -- seed
           -> Highschools
            -- list of schools and their capacity
           -> Int
            -- bound in generation size
           -> Float
            -- crossover parameter
           -> Float
            -- mutation parameter
           -> Preferences
            -- preference lists for all students
           -> Population
            -- old generation
           -> Population
            -- new generation
createGen seed highschools gensize cpar mpar prefs pop = let
       [(seedPool, seedCross, seedMut)] = tripleUp $ generateSeeds seed 3
       reproduced                   = reproduction seedPool pop gensize
                                       highschools prefs
       bestentity                   = head reproduced
       pool                         = tail reproduced
       nrstudents                   = fromIntegral $ length $ map fst prefs
       seedsmut                     = generateSeeds seedMut gensize
   in let
       sizecrossoverpool = round $ (fromIntegral gensize)*cpar -
                           (mod' ((fromIntegral gensize)*cpar) 2)
       crossoverpool   = pairUp $ take sizecrossoverpool pool
       clonepool       = drop sizecrossoverpool pool
       seedscross      = take gensize (randoms $ mkStdGen seedCross)
       seedsmut        = generateSeeds seedMut gensize
     in freqRep $ sortallocations $ bestentity : (mutate seedsmut highschools prefs
                ((crossover' seedscross highschools prefs crossoverpool) ++
                clonepool))
       where
        crossover' _ _ _ []                          = []
        crossover' (s:srest) highschools prefs ((a,b):prest) =
           (crossover s highschools prefs a b) ++
```

```
            (crossover' srest highschools prefs prest)

        mutate _ _ _ []                          = []
        mutate (s:srest) highschools prefs (o:orest) =
           (mutation s (mpar :: Float) highschools prefs o) :
              mutate srest highschools prefs orest

        sortallocations :: [Organism] -> [Organism]
        sortallocations [] = []
        sortallocations (org:orgs) = sortallocation org : sortallocations orgs
           where
              sortallocation (alloc,perm) =
                 (map (\(sch,stds) -> (sch, sort stds)) alloc,
                  perm)

-- Function Reproduction
-- The function reproduction performs reproduction using a roulette wheel
--  reproduction technique in combination with elitism.
reproduction ::  Seed
               -- seed
             -> Population
               -- population
             -> Int
               -- bound on generation size
             -> Highschools
               -- list of schools and their capacity
             -> Preferences
               -- preference lists for all students
             -> [Organism]
               -- allocations that are reproduced s.t. the first allocation is the
               --  best entity of the population, the order of the tail is totally
               --  random, since the probability that some organism occurs at some
               --  position in the tail is equal to its fitness weight in the total
               --  fitness, which is its probability to survive, hence according to
               --  the probability of survival organisms occur.
reproduction seed pop size highschools prefs = let
       nrstudents = length $ map fst prefs
       pop'       = map fst pop
       fitnesspop = zip pop (map (\x -> fitness x prefs highschools) pop')
       bestentity = fst $ fst $ maximumBy (comparing snd) fitnesspop
       totalfit = sum $ map (\(((alloc,permut), freq), fit) ->
                           fromIntegral freq * fit) fitnesspop
       fitProb = normalize $ map (\ (((alloc,permut), freq),fit) ->
                              fromIntegral freq * fit / totalfit ) fitnesspop
       popInterval = zip pop' (cumList fitProb)
       coinflips  = randomRange seed (0.0,1.0) (size-1)
      in bestentity : reproduce coinflips popInterval
       where
          reproduce [] _ = []
          reproduce (flip:rest) popInterval =
             reproduce' flip popInterval : reproduce rest popInterval
             where
                reproduce' flip ((organism,(a,b)):rest) = if b == 1
                   then if a <= flip && flip <= b
                          then organism
                          else reproduce' flip rest
                   else if a <= flip && flip < b
                          then organism
                          else reproduce' flip rest
```

```
-- The function mutation given some organism (alloc,perm) picks a random real r in the
--  interval [0.0,1.0] for every student in alloc. If r is smaller than the mutation
--  parameter, then the student is switched with some random selected student which has
--  a lower index than or equal index as the student in perm.
mutation ::  Seed
             -- seed
          -> Float
             -- mutation parameter, percentage (<1.0) of students who will 'mutate'
          -> Highschools
             -- list of schools and their capacity
          -> Preferences
             -- preference lists for all students
          -> Organism
             -- organism that will be mutated
          -> Organism
             -- mutated allocation
mutation seed mpar highschools prefs (allocation,permutation) = let
    studs       = concat $ map snd allocation
    seedsmut    = pairUp $ generateSeeds seed ((length studs) * 2)
   in (allocationByPermutation (mutate seedsmut mpar permutation studs)
          prefs highschools,
      permutation)
    where
       mutate _ _ permutation [] = permutation
       mutate ((seedCF, seedRS):seedpairs) mpar permutation (stud:studs) = let
           coinflip  = fst $ randomR (0.0,1.0) (mkStdGen seedCF)
          in if coinflip >= mpar
                  then mutate seedpairs mpar permutation studs
                  else let
                         Just indexstud = elemIndex stud permutation
                         newindices     = [x | x <- [0..indexstud] ]
                         newindex       = randIndex seedRS (length newindices)
                        in exchange permutation newindex indexstud

-- The function Crossover given two organisms (alloc1, perm1) and (alloc2, perm2),
--  splits perm1 in two pieces perm1a and perm1b, s.t. no piece is the empty list, and
--  creates two new permutation perm1a++perm2' and perm2''++perm1b, where
--     perm2' are all the students not occurring in perm1a ordered as in perm2
--     perm2'' are all the students not occurring in perm1b ordered as in perm2.
crossover ::  Seed
              -- seed
           -> Highschools
              -- list of schools and their capacity
           -> Preferences
              -- preference lists for all students
           -> Organism
              -- organism that will crossover
           -> Organism
              -- organism that will crossover
           -> [Organism]
              -- list of two crossed over allocations
crossover seed highschools prefs (alloc1, perm1) (alloc2, perm2) = let
    cutoff         = 1 + (fst $ random (mkStdGen seed)) `mod` ((length perm1) - 1)
    (perm1a,perm1b) = splitAt cutoff perm1
   in [(allocationByPermutation (crossover' True perm1a perm2)
          prefs highschools, (crossover' True perm1a perm2)),
      (allocationByPermutation (crossover' False perm1b perm2)
          prefs highschools, (crossover' True perm1a perm2))]
```

```
    where
        crossover' head partof1 perm2 = let
            missing = filter (`nelem` partof1) perm2
          in if head
               then partof1 ++ missing
               else missing ++ partof1
```

7.3.3 Fitness Functions

N.B. the variants with no penalty for not-allocated students are omitted, because these can be obtained from the variants with the penalty by just deleting the function `schoolzerofactor`.

Most First Choice Students with Penalty for Not-Allocated

```
-- Fitness by number of students assigned their first choice with penalty for not allocated.
fitness :: Organism -> Preferences -> Highschools -> Float
fitness (allocation, permutation) prefs highschools =
    (schoolzerofactor allocation * numberOne allocation highschools prefs) ** 2
    where
    -- The function schoolzerofactor returns 1 if no students are allocated to school
    --  0, and lower values the more students are allocated to zero.
    schoolzerofactor alloc = (fromIntegral 1) /
                             (fromIntegral $ 1 + (length $ lookup' 0 alloc))

    -- The function numberOne returns the number of students who are assigned their first
    --  choice in some given allocation.
    numberOne :: Allocation -> Highschools -> Preferences -> Float
    numberOne alloc highschools prefs = let
        studs = map fst prefs
       in snd $ numberOne' alloc highschools prefs (studs,0.0)
        where
            numberOne' alloc highschools prefs ([],score) = ([], score)
            numberOne' alloc highschools prefs ((stud:studs),score) = let
                firstchoice = head $ lookup' stud prefs
                school      = fst $ head $ filter (\(sch,stds) -> stud `elem` stds) alloc
              in if firstchoice == school
                   then numberOne' alloc highschools prefs (studs,score+1.0)
                   else numberOne' alloc highschools prefs (studs,score)
```

Lowest Mean Preference with Penalty for Not-Allocated

```
-- Fitness by mean preference position with penalty for not allocated.
fitness :: Organism -> Preferences -> Highschools -> Float
fitness (allocation, permutation) prefs highschools =
    (schoolzerofactor allocation * fitness' allocation prefs) ** 2
    where
    -- The function schoolzerofactor returns 1 if no students are allocated to school
    --  0, and lower values the more students are allocated to zero
    schoolzerofactor alloc = (fromIntegral 1) /
                             (fromIntegral $ 1 + (length $ lookup' 0 alloc))
    -- The function fitness' calculates the fitness of some allocation by summing
    --  the distance of the actual preference position of each student allocated to
    --  some school in the allocation from the lowest possible preference position
    fitness' [] _ = 0.0
    fitness' ((sch,stds):rest) prefs = (fromIntegral $
```

```
                                        fitnessschool sch stds prefs) +
                                        fitness' rest prefs
        where
        -- The function fitnessschool returns the sum of the preference positions of
        --  the students allocated to that school.
        fitnessschool :: School -> [Student] -> Preferences -> Int
        fitnessschool _ [] _ = 0
        fitnessschool sch (std:rest) prefs = let
            least  = length highschools
            stdpref = lookup' std prefs
            prefpos = if elemIndices sch stdpref == []
                      then least
                      else (head $ elemIndices sch stdpref) + 1
                      -- if a school is not in a student's pref. list, then it is
                      --  said that the school is on the least possible pref position
          in (least - prefpos) + fitnessschool sch rest prefs
```

Most First Choice Students with Penalty for Not-Allocated and Penalty for Non-Maximal

---

```
-- Fitness by number of students assigned their first choice with penalty for not allocated
--  and a penalty for the number of students who are assigned to some school which they do
--  not prefer most and for which there are more students who place it highest than its
--  capacity.
fitness :: Organism -> Preferences -> Highschools -> Float
fitness (allocation, permutation) prefs highschools =
    (schoolzerofactor allocation *
        nonmaximal allocation highschools prefs *
            numberOne allocation highschools prefs) ** 2
    where
    -- The function schoolzerofactor returns 1 if no students are allocated to school
    --  0, and lower values the more students are allocated to zero.
    schoolzerofactor alloc = (fromIntegral 1) /
                             (fromIntegral $ 1 + (length $ lookup' 0 alloc))

    -- The function numberOne returns the number of students who are assigned their first
    --  choice in some given allocation.
    numberOne :: Allocation -> Highschools -> Preferences -> Float
    numberOne alloc highschools prefs = let
        studs = map fst prefs
       in snd $ numberOne' alloc highschools prefs (studs,0.0)
        where
            numberOne' alloc highschools prefs ([],score) = ([], score)
            numberOne' alloc highschools prefs ((stud:studs),score) = let
                firstchoice = head $ lookup' stud prefs
                school     = fst $ head $ filter (\(sch,stds) -> stud `elem` stds) alloc
               in if firstchoice == school
                   then numberOne' alloc highschools prefs (studs,score+1.0)
                   else numberOne' alloc highschools prefs (studs,score)

   -- the function nonmaximal returns 1 if the allocation is non-maximal and returns
   --  1/1+#nm, where #nm are the number of non-maximal students.
    nonmaximal alloc highschools prefs = let
        studs = map fst prefs
       in (fromIntegral 1) /
                (fromIntegral $ 1 + nonmaximal' (studs,0) alloc highschools prefs)
            where
                nonmaximal' ([],score) alloc highschools prefs = score
```

```
nonmaximal' ((stud:studs),score) alloc highschools prefs = let
    school       = fst $ head $
                        filter (\(sch,stds) -> stud `elem` stds) alloc
    Just capacity = lookup school highschools
    firstchoicers = map fst $ filter
                        (\(stud,prefs) -> (head prefs) == school) prefs
    assignees    = lookup' school alloc
   in if length (firstchoicers) > capacity
       then nonmaximal' (studs,score) alloc highschools prefs
       else let
              nelem x xs = not (x `elem` xs)
              score'     = length $ filter
                                (\stud -> nelem stud firstchoicers)
                                   assignees
             in nonmaximal' (studs,score+score') alloc highschools
                 prefs
```