Analyzing Dropouts in School Matching

Francisco Fonseca, Divi Khanna, Pavithra Srinath
Operations Research in Public Policy
Fu Foundation School of Engineering and Applied Science, Columbia University

1 Introduction

"Matching" is the part of economics that attempts to answer the question of who gets what, particularly when the goods are limited and indivisible and have be allocated; Eg., who works at which job, which students go to which school, who receives which transplantable organ, etc. Studying how particular matching markets succeed at creating efficient matches, or fail to do so, has yielded insights into how markets in general work well or badly.

In general, some of the problems that are general to all kinds of markets are safety issues, congetsion, fairness of allocation and stablilization. Markets can be broadly categorized into two types. Markets in which agents on one side need to be matched with agents on the other side when agents on both sides have their own preferences are called two sided markets. Eg., job markets wher employers and employees have their own preferences. Markets where agents one one side either have no preference or their preference is not not taken into consideration are called one sided markets. Eg., Dormitory allocation in schools where students choices are not considered.

We have seen numerous problems in matching such as the problem of kidney matching, house allocation, housing markets, school choice etc. In this project, we attempt to explore and answer a few questions in the school choice problem.

2 Problem Statement

Consider the student proposing deferred acceptance algorithm (DA) [5]. Apply DA on an instance and call the obtained stable match μ_1. Consider the scenario where a certain number of students drop out of the system. Now, re-compute a new stable match using DA and call it μ_2. The questions we are interested in exploring are:

1. How many allocations change between μ_1 and μ_2 ?
2. Can a pattern be observed in these changes?
3. Are there ways to move from μ_1 to μ_2 without running DA again?
4. Is it possible to approximate μ_2 from μ_1?

As part of this project, we hoped to explore the above questions. The expectation was to design a framework that would enable us to simulate the problem as close as possible to the real world scenario.

3 A brief background of the school choice problem

In this section we present a brief background of the school choice problem. The content of this section is a summary of the much more detailed survey in [1].

3.1 School Choice problem

A school choice problem consists of a finite set of students and a finite set of schools with finitely many seats available for enrollment. Formally, this problem can be mathematically defined by:

- a finite set of students I,
- a finite set of schools S,
- school capacities $q = (q_s)_{s \in S}$, where q_s is the number of available seats at school $s \in S$,
- a profile of student preferences $P = (P_i)_{i \in I}$,
- and a profile of school priorities $\succsim = (\succsim_s)_{s \in S}$

The preferences P_i of each student $i \in I$ are considered as being strict over the set of schools S and the outside option $\{o\}$. On the other hand, each school $s \in S$ has a weak priority relation $\succsim_s$ over the students and the empty seat ($\varnothing$).
A matching of students to schools is a mapping $\mu : I \cup S \rightrightarrows 2^{I \cup S}$. As a result of this mapping a student is matched with a school, the number of students matched with a school cannot exceed its capacity and a student is matched with a school if and only if the school is also matched with the student.
Finally, a mechanism is defined as a student admission procedure. It deterministically selects a matching for every problem, according to a specified set of rules. The designer of a mechanism wishes to fulfil a set of desired properties. The main properties that a mechanism must satisfy will be listed in the next section.

3.2 Desired properties of a mechanism

When designing a mechanism, the designer wishes to accomplish some properties. Some of them are trivially defined (such as feasibility). Others are less clear. In this section, we will present some of the most important properties of school choice mechanism design.

Feasibility

Overcrowding schools is obviously not desired in the solution of a school choice mechanism. In the context of school choice, feasibility is defined as a solution that doesn't exceed schools' capacities. Hence, a matching is deemed feasible if enrollment at each school does not exceed the school capacity and only eligible students are enrolled at every school.

Individual rationality

If a student is assigned a school that is not in his choice list, one may expect his family to opt out for its outside option (which may be interpreted as a private school, home schooling or some other option outside of the public system in question). A matching is individually rational if it matches every student only with schools in his choice list, and leaves him unassigned otherwise. Thus, when the student participates in matching process, his welfare will be greater or equal to his welfare if he doesn't participate in the assignment process.

Efficiency

Perhaps the most obvious property that guides a design of a school choice mechanism is that the matching process should promote student welfare to the greatest extent possible; that is, it should be efficient for students. To define an efficient matching we use the concept that it *Pareto dominates* all other matchings. A matching μ Pareto dominates another matching if μ improves some student's welfare without harming others in comparison to the latter. Hence, a matching is Pareto efficient (or simply efficient) if it is not Pareto dominated by another matching.

Respecting or violating priorities in assignment

A matching violates a student's priority at school s if the student ranks s higher than his assigned school and has higher priority at s than some other student who is assigned to school s. A matching is *stable* if it does not violate priorities and not waste any seat. Also, a stable matching that is not Pareto dominated by any other stable matchings is defined as a *student optimal stable matching*.

Incentives to game the system

A mechanism determines the matching of students with schools for every profile of preferences, priorities and school capacities. Hence, students may in theory change their assignment by changing the preference list of schools that he submits, i.e. by *gaming the system*. An assignment mechanism is *strategy-proof* (for students) if listing schools in true preference order in the application form is optimal for every student, regardless of the priority structure and other students' applications. Strategy-proofness is a desirable property in a mechanism. First, it makes the decision process simpler, as parents only need to list their true preferences. Also, some parents may not have the proper information and ability to game the system and will be left in disadvantage during the assignment process. Strategy-proofness eliminates the imbalance between the players.

3.3 Three different assignment mechanisms

In this section, we will present and compare three different school choice mechanisms.

Boston Mechanism

The *Boston School Mechnism* was developed in Cambridge in the 1980s and was widely used across the United States in the following years. Basically, it tries to assign as many students as possible to their first choices. The matching process is defined by the following algorithm:

- Step 1: For each school, consider the students who have listed it as their first choice. Assign seats to these students one at a time in the order of priority at that school until either there are no seats left or there is no student left who has listed it as his first choice.
- Step k (in general): consider only the k^{th} choices of the students not assigned in the previous step. For each school with available seats, assign the remaining seats to the students who have listed it as their k^{th} choice in the order of priority at that school until either there are no seats left or there is no student left who has listed it as his k^{th} choice.

The algorithm terminates when there are no more students to be assigned.
It has been demonstrated that the Boston Mechanism is not stable and neither strategy-proof. To visualise these facts we may consider the following example:

Example 3.3.1. There are three students $\{a, b, c\}$ and three schools $\{s_1, s_2, s_3\}$ each with one seat. Student preferences and school priorities are given as follows:

$$\begin{array}{lccccc} a: & s_2 & - & s_1 & - & s_3 \\ b: & s_1 & - & s_2 & - & s_3 \\ c: & s_1 & - & s_2 & - & s_3 \end{array} \quad and \quad \begin{array}{lccccc} s_1: & a & - & c & - & b \\ s_2: & b & - & a & - & c \\ s_3: & b & - & a & - & c \end{array}$$

Applying the Boston Mechanism algorithm, we get the following matching:

$$\mu_{Boston} = \begin{pmatrix} a & b & c \\ s_2 & s_3 & s_1 \end{pmatrix}$$

μ_{Boston} is not stable since b was assigned his third choice although it has a higher priority in s_2 (his second choice). Also, we can observe that if b declared s_2 as his first choice, it would be assigned to it, which it prefers than s_3. Hence, b has an incentive to declare preferences that are not truthful, which shows that this mechanism is not stretegy-proof. Actually, this fact was publicly known when this mechanism was in use and parents were explicitly advised by the Boston Public Schools guide to follow this strategy when submitting their preferences.

Student Optimal Stable Matching

The Student Optimal Stable Matching mechanism (SOSM) was developed by Gale and Shapley in 1962. It finds an optimal (in the students point of view) stable matching (i.e., a stable matching that is not pareto dominated by any other stable matchings). The matching is determined by the following algorithm:

- Step 1: Each student applies to his first choice. Each school tentatively assigns its seats to its applicants one at a time according to their priority order until capacity is reached. Any remaining applicants are rejected.
- Step k (in general): Each student who was rejected in the previous step applies to his next best choice (if one remains). Each school considers the set consisting of the students it has been holding from previous steps and its new applicants, and tentatively assigns seats to there students one at a time following its priority order. After all seats are filled, the students not assigned a seat are rejected.

The algorithm terminates when no more students make applications and then the tentative assignments are finalized. Applying this algorithm to the case presented in Example 1, we find the following matching:

$$\mu_{SOSM} = \begin{pmatrix} a & b & c \\ s_1 & s_2 & s_3 \end{pmatrix}$$

As its name states, the SOSM algorithm produces a stable matching. Since it only assigns seats tentatively in each step and students with higher priorities may be considered in subsequent steps, SOSM guarantees that no student looses a seat to a lower priority student and is assigned to a less-preferred school. Also, as already mentioned, all students prefer μ_{SOSM} to any other stable matching (i.e., it pareto dominates all other stable matchings). Also, SOSM is strategy proof. Differently form the Boston Mechanism, assignments are made tentatively. Then, a student may always loose his seat in subsequent steps to a higher preferred student. Thus, students have no incentives to change their preference order.

Efficient Transfer Mechanism

Another mechanism is The Efficient Transfer Mechanism (ETM), proposed by Abdulkadiroglu and Sonmez (2003), also known in the literature as Top Trading Cycles mechanism (TTC). This mechanism takes a different approach from the two previous mechanism presented. It lines up students in each school according to their priorities in that school. Then it tentatively assigns one empty seat at a time to the highest priority students. If a student is satisfied with his assignment, he keeps it. Otherwise, ETM implements seats transfers among students in order to increase welfare. Once such transfers are finished, it goes back and assigns a seat to the next highest priority student. The matching process is defined by the following algorithm:

- Step 1: Every school points to its highest priority student, every student points to his most preferred school. A transfer cycle is defined as an ordered list of schools and students with school 1 pointing to student 1, student 1 to school 2, ..., school k to student k, and student k pointing to school 1. All such cycles are found and every student in a cycle is assigned a seat at the school he is pointing to and is removed; the number of seats at that school is decreased by one.

- Step k (in general): Every school with still available seats points to its highest priority student, every student points to his most preferred school with still available seats. All the transfer cycles are found. Every student in a cycle is assigned a seat at the school he points to and is removed; the number of seats at that school is decreased by one.

The algorithm terminates when no more students are assigned. Applying this algorithm to Example 1, we get the following matching:

$$\mu_{ETM} = \begin{pmatrix} a & b & c \\ s_2 & s_1 & s_3 \end{pmatrix}$$

Comparing this matching to μ_{SOSM}, we observe that μ_{ETM} pareto dominates μ_{SOSM}, since a and b prefer μ_{ETM} to μ_{SOSM} and c is indifferent. Hence, we can see that SOSM is not efficient. According to Abdulkadiroglu, ETM is efficient and strategy proof. However, ETM is cannot guarantee stability (we can observe that μ_{ETM} is not a stable matching, since c has a higher priority in s_1 than b).

4 Methodology

4.1 Framework

We realized that the problem we were interested had four main parameters that influenced the behavior of the system. The four parameters were:

- Capacity of the school
- Number of schools
- Number of students
- Number of dropouts

We decided to approach the problem in a simplistic manner. In order to avoid working with four varying parameters, we decided to allow only one degree of freedom. Keeping the capacity of the schools fixed, we tried to reduce the problem to an instance with one of the three remaining three parameters. Table 1 illustrates the different scenarios described above. However, due to time constraints, we have explored only the first case as part of this project.

Parameters	Case 1	Case 2	Case 3
Capacity of schools	X	X	X
Number of Schools	X	X	
Number of students	X		X
Number of dropouts		X	X

Table 1: Constraints for different cases

Initially, when we started to analyze this problem, we found two issues. One was how to generate preference lists for students. Initially, we were generating lists in a uniformly random fashion. Later, as suggested by Prof. Sethuraman, we used the model described below.

We divide the schools into three zones - good, medium and bad. The preference lists of the schools still follows $Uniform(1, N)$. The preference lists for students are uniformly randomly distributed within the thre zones. Let us suppose that the number of schools in the three zones

are $K_1, K_2 \text{and} K_3$ respectively such that $K_1 + K_2 + K_3 = N$. Let P_S^i denote the preference list for students in zone i. Then,

$$P_S^{Good} \sim Uniform(1, K_1), P_S^{Medium} \sim Uniform(1, K_2), P_S^{Bad} \sim Uniform(1, K_3)$$

By fixing the capacity of the schools to be one, we restrict the scope of this project to one-to-one matchings only. Let us suppose there are N schools and N students, then the preference lists for schools $\sim Uniform(1, N)$ and for students as described above.

The second issue we found was to determine how to simulate the dropout behavior. We have simulated two dropout behaviors described below.

1. Students dropout randomly - In this case, a subset of students drop out in a uniformly random manner ie., students who drop out $\sim Uniform(1, K), 0 < K < N$
2. Only a subset of students allocated to bad schools dropout.

4.2 Metrics

- Kendall tau distance is a metric that enables us to understand the correlation between two lists. It counts the number of pairwise disagreements between two ranking lists. The lower the distance, the more similar the two lists are.

  If T_1 and T_2 are two lists, then Kendall tau distance is defined as:
  $K(T_1, T_2) = \sum_{(i,j) \in P} K_{i,j}^{'}(T_1, T_2)$, where

  P is the set of unordered pairs if distinct elements in T_1 and T_2
  $K_{i,j}^{'}(T_1, T_2) = 0$, if i and j are in the same order in T_1 and T_2
  $K_{i,j}^{'}(T_1, T_2) = 1$, if i and j are in the opposite order in T_1 and T_2

  $K(T_1, T_2)$ will be equal to 0 if two lists are identical and $\frac{n(n-1)}{2}$ if one list is the reverse of the other.
  It is normalized by dividing by $\frac{n(n-1)}{2}$, so a value of 1 indicates maximum disagreement.

- In the second dropout scenario, we compute the number of changes that occur across zones to monitor the improvement induced by our algorithms.
- We also compute the number of blocking pairs in matches that are not stable.

4.3 Algorithm using Truncation of Preference List

- In our objective to reach stable matching after students dropout, we expect that there would be very many algorithms to reach closer to stable matching without running the complete DA again. We study one particular family of algorithms that use the mechanism of truncating preference list of schools, and compare our results with the stable matching achieved after running DA again.
- Optimal truncation has been extensively researched as a mechanism used in matching markets to study the strategic behavior of participants. In large markets where all other participants are being truthful, it is a dominant strategy to truncate your own preference list by 100% [4].
- This idea has been discussed in the *Divorcing Algorithm* by authors [4]. This algorithm is discussed below, and we can see that it comes close to our case of simulating dropouts from a system of school choice.

- School choice requires students to provide a preference list of 12 schools they want to attend. In the context of matching algorithms, this is similar to the case when women truncate their preference list in large markets, listing in order the first several men from their true preference list and declaring all other men unacceptable. This gives us motivation to base our study on the family of truncating preference list algorithms.
- It must be noted that truncation generates a tradeoff: it may cause a woman to match with the better-ranked men she leaves on her list, but may also cause her to be left unmatched.

4.3.1 The Divorcing Algorithm

Authors [4] show that when woman w submits a k-truncation of her preference list to Man Proposing-Deferred Acceptance (MP-DA) algorithm, the outcome is identical to that from a two stage Divorcing Algorithm. In the first stage of the algorithm, agents submit preference lists to MP-DA. In the second stage, w 'divorces' her mate and declares all men with rank $\geq k$ unacceptable. This sets off a chain of new offers and proposals, ending in a new match.

The Divorcing Algorithm takes as its input a set P of preference lists, a woman w, and a truncation point $k \in \{0, ..., N\}$ and generates a matching $\mu^{DIV}[P, k, w]$.

1. Initialization: Run MP-DA to find the men-optimal matching $\mu^M[P]$: If w is single or if w's mate has rank $\leq k$ in P_w, terminate. Otherwise, divorce w from her mate. Declare candidates with rank $\geq k$ unacceptable for w.
2. Pick an arbitrary single man who has not exhausted his preference list. If no such man exists, terminate.
3. The man chosen in the previous step makes an offer to the most preferred woman on his preference list who has not already rejected him. If this woman finds the man acceptable and she prefers him to her current mate (or is single), she divorces (if necessary) and holds the new man's offer. Return to step 2.

The Divorcing Algorithm yields a matching identical to the output from w's submission of a k-truncated list to MP-DA.

4.3.2 Truncation of Preference List in School Choice

1. We run Student Proposing DA on a list of preferences P to get a stable match. Then randomly N_d students are selected to drop out of the system.
2. A complete version of Student Proposing DA is run again to get a stable match. This matching does not have any blocking pairs and we assume this to be our benchmark match $\mu^B[P]$.
3. We then take only those Schools who have empty seats as a result of dropouts, and truncate their preference list to obtain a new preference list P' of length l. A School Proposing DA is run now only for those schools who have empty seats.

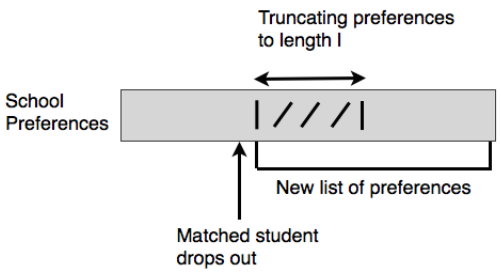

Figure 1: Diagramatic representation of truncation in preference list of schools

4. This matching obtained from School Proposing DA is our test match $\mu^T[P']$, and we then comment on its stability by counting the number of blocking pairs against the stable benchmark match $\mu^B[P]$.

5. The intuition behind a School Proposing algorithm is to avoid the load of applications in a Student Proposing version. Truncating list of prefrences is similar to restricting interest only to those students who have been wait-listed by a particular school.

4.3.3 Simulation Procedure of our Algorithm

1. We have simulated a one-to-one matching between N=100 students and schools. To be closer to real ranking of schools, we have also divided schools into bins of 20% good, 50% medium and 30% bad.

2. We fix total number of dropouts N_d=9 (dropping them: first randomly and then only from bad schools) and generate $\mu^B[P]$.

3. We then truncate the preference list of schools to length l=3 and run School Proposing version of DA. This matching is called test match $\mu^T[P']$.

4. We count the number of blocking pairs in $\mu^T[P']$ as compared to the stable benchmark match $\mu^B[P]$.

5. The algorithm is simulated 200 times to obtain the average number of blocking pairs.

6. We then change the values of l to plot the relation in our results.

4.4 Dropout Algorithm Analysis

We now make a modification to our algorithm to improve efficiency and also aid analysis. However, before we do that, let us first give a clear pseudocode representation of the algorithm.

Algorithm 1 : Recompute-Stable-Match(p)

```
1: s ← School that p dropped out of
2: while s has rejected students do
3:     s gets matched to the most preferred student p' among rejected students
4:     s ← School that p' dropped out of
5: end while
```

Algorithm 1 is a pseudocode presentation of how we can recompute a stable match after a student drops out. In the algorithm, p is the student that drops out. We shall now provide a proof of correctness of the algorithm.

Theorem 4.4.1. *Algorithm 1 returns a stable match.*

Proof. By way of contradiction, let us suppose that the algorithm does not return a stable match. This means that there is at least one blocking pair. Let the blocking pair be such that school s_a is matched to pupil p_b and school s_b is matched to pupil p_a, but s_a prefers p_a over p_b and p_a prefers s_a over s_b. We know that p_a is sure to have proposed to s_a before proposing to s_b given the way the original algorithm works. This means that p_a is sure to be on the *rejected list* of s_a. We can see that Line 3 of Algorithm 1 would then have assigned p_a to s_a, and not s_b, which is a contradiction. □

Lemma 4.4.2. *There is at least one school which has received no more than one proposal.*

Proof. By way of contradiction, let us assume that every school received more than one proposal from a student. Consider the last school, say s_l that received a proposal. Since, s_l received another proposal after this, we have that there was more than one unmatched student before s_l received a proposal. By the pigeon-hole principle, this means that, during the time of this last proposal, there was some other school, say s_p, that was unmatched. Given that the current school is the last school to be receiving a proposal, this means that s_p received a proposal but was not matched. This is a clear contradiction to how the algorithm works. □

Theorem 4.4.3. *Algorithm 1 terminates.*

Proof. The algorithm terminates when we find a school that has no students it has previously rejected. This follows directly from Lemma 4.4.2 □

This concludes the proof of correctness of the algorithm. Let us now analyze the runtime of the algorithm. Before we start, let us note that re-computing the stable match right from scratch takes asymptotic runtime of $\Omega(n^2)$. It is also easy to see that re-running DA with truncated preference lists has the same runtime. In light of this, we now prove the following theorem:

Theorem 4.4.4. *Algorithm 1 takes $O(n)$ running time to terminate.*

Proof. It is easy to maintain, for each school, a list of students that were rejected, in order of preference. This causes no overhead to the original DA algorithm. For a school s, let $\mathcal{R}_s$ be the list of rejected students. Line 3 of Algorithmalg:recomputesum will take $O(1)$ lookup time to match s to the first student in $\mathcal{R}_s$. Every school gets re-matched only once, and we know that the algorithm executes *re-matching* for at most $n-1$ schools. □

We have now proved that Algorithm 1 returns a stable match, and that it takes $O(n)$ running time. We complete the analysis with the following theorem:

Theorem 4.4.5. *The stable match returned by Algorithm 1 is the same as the stable match that would be returned by re-running DA.*

Proof. This is self-evident. Notice how, for each school, we *re-match* it to a student that already proposed. This is equivalent to the case where the school did not receive a proposal from the student that dropped out. Thus, in other words, we are just undoing the extra proposals that were caused due to the presence of the student (who later dropped out). □

5 Simulations and Results - Analysis

5.1 Initial simulation results and analysis

We initially computed the student proposing DA stable match (μ_1). We then simulated students dropping from the system. We then recomputed the student proposing optimal (μ_2). We compared the two matchings μ_1 and μ_2 using the Kendall tau distance metric.

1. Dropout case 1 results - Figure 2 illustrates the results for the first dropout scenario where students drop out in a uniformly random fashion.

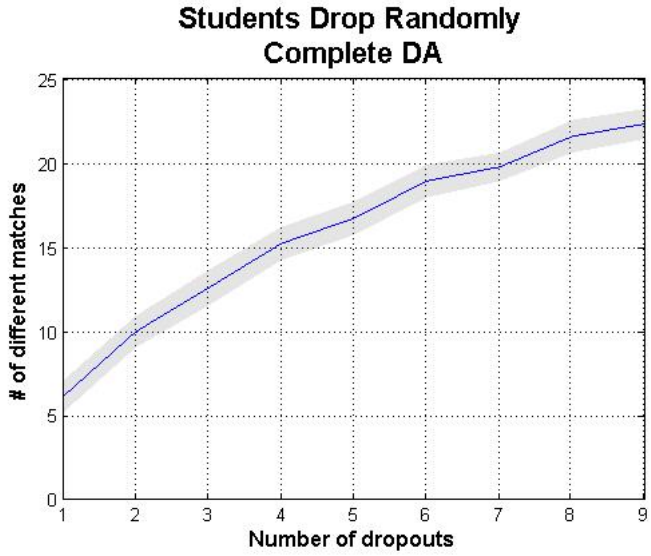

Figure 2: Dropout Case 1:Comparing μ_1 and μ_2 using Kendall tau distance

Figure 3 below graphically represents the number of changes that occurred across zones in the case of each dropout.

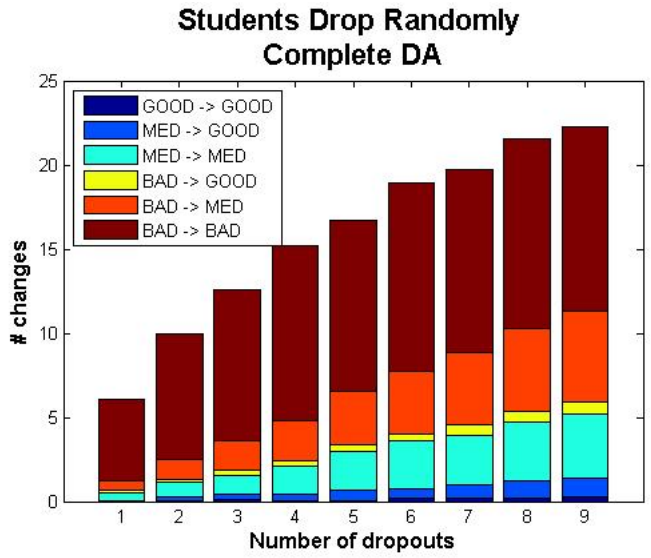

Figure 3: Dropout Case 1:Comparing μ_1 and μ_2 based on changes across zones

2. Dropout case 2 results - Figure 4 illustrates the results for the first dropout scenario where only a subset of students who were allocated to bad schools dropout. The number of changes tends to remain constant after a certain number of dropouts. This is quite intuitive because since the schools that have empty spots are bad schools, no student would prefer to go to any of those schools.

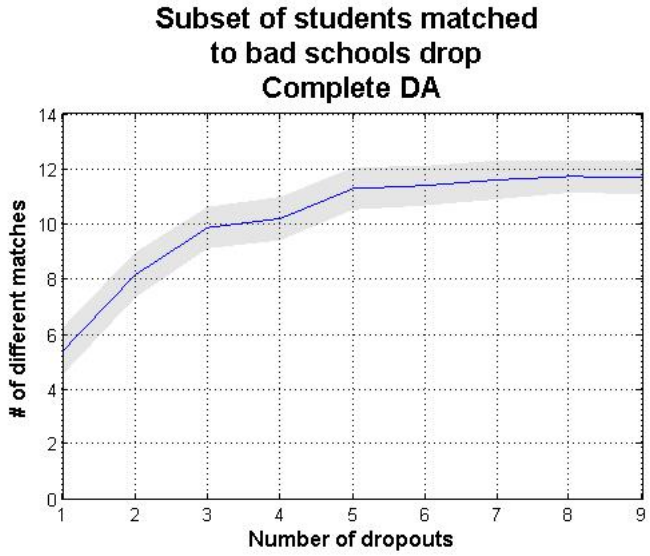

Figure 4: Dropout Case 2:Comparing μ_1 and μ_2 using normalized Kenall tau distance

Figure 5 below graphically represents the number of changes that occurred across zones in the case of each dropout. In this case, we clearly, see that most of the changes are occuring only within the bad school zone.

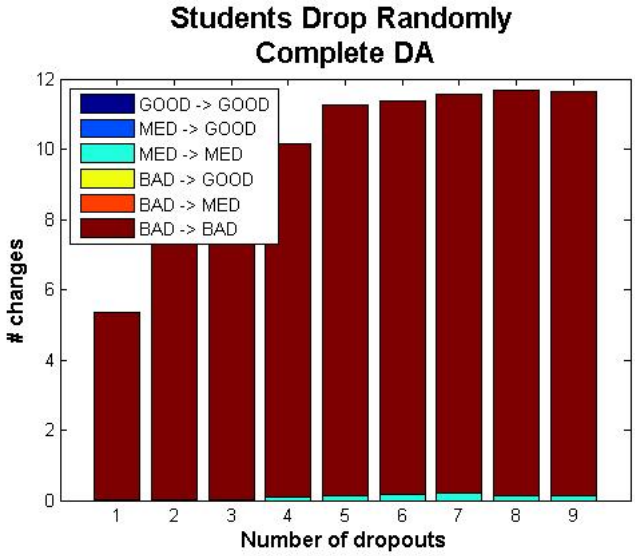

Figure 5: Dropout Case 2:Comparing μ_1 and μ_2 based on changes across zones

5.2 Implementation of Algorithm & Stability-Workload Tradeoff

In our truncation of preference list algorithm, we move from *Dropout* matrix to a *Stable* matrix to a $Semi - stable$ matrix, and then collect the variables for each matrix to better compare our notion of stability. We have used both the number of blocking pairs and the Kendall Tau distance as our metrics for comparison.

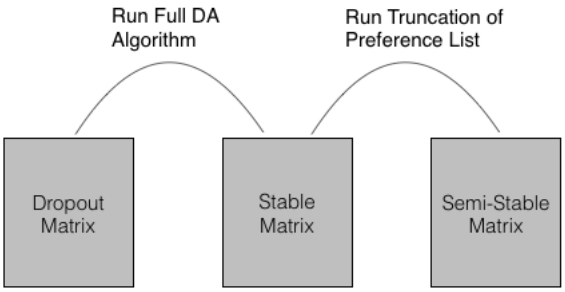

Figure 6: Diagramatic representation of different stages of the algorithm

5.2.1 Workload of Applications

- It is quite intuitive to see that by changing school matching from a student proposing to a school proposing algorithm, we reduce the application workload in the system. In our specific simulation example, we dropped 9 students, and then selected schools with empty seats to propose to students of their choice. This resulted in 9x3=27 extra applications in our base case, which is much lesser than running the complete student-proposing DA for 100 students and 100 preference lists again.

5.2.2 Number of Blocking Pairs

- The blocking pairs reduce as we increase the length of preference list. It proves that truncation does create more blocking pairs in the system.
- However, there is a tradeoff between increase in blocking pairs and workload of more applications in the system. There will be an optimal truncation length l^* that balances this tradeoff.
- The improvement is better in the case where students dropout uniformly from the entire preference list, than when they leave only from bad schools.

Students Randomly (uniform) dropout

Length l	Blocking Pairs After	Blocking Pairs before	Improvement
1	68.4	81.1	12.7
2	62.8	82.0	19.2
3	60.1	81.3	21.2
4	59.0	81.9	22.9
5	57.9	81.5	23.5
6	56.8	81.5	24.7
7	58.6	81.8	23.1
8	57.5	81.4	23.9
9	58.2	82.1	23.9
10	58.0	82.4	24.4

Figure 7: Simulation runs when students dropout uniformly across all schools

9 Students only from bad schools dropout

Length l	Blocking Pairs After	Blocking Pairs before	Improvement
1	9.4	9.6	0.2
2	9.0	9.3	0.3
3	9.3	9.6	0.3
4	9.3	9.8	0.6
5	9.0	9.5	0.5
6	8.6	9.4	0.8
7	8.5	9.2	0.8
8	8.7	9.5	0.8
9	8.4	9.4	1.0
10	8.3	9.5	1.1

Figure 8: Simulation Runs when students drop uniformly across bad schools

5.3 Changes Across Classes of Schools

- The following figures compare, on average, the kendall tau-distance of the complete DA executed after the dropouts (our benchmark) and the truncated school choice. They present the pattern of the pairwise differences between these matchings and the original one.

- Since reproducing the complete DA is too costly, we would like our semi-stable matching to be as efficient as possible. In other words, we would like it to waste fewer seats in good and medium schools.

- We can observe that the truncated school proposal algorithm approaches the benchmark value. For example, with a truncated list of length 10, the benchmark value is approximatelly 23 and the school proposal value is almost 7. Also, one important point is that most of the difference between these two values is concentrated in the BAD to BAD case, which means that the truncated school proposal algorithm is mainly throwing away changes from one bad school to another bad school.

- In the truncation preference list algorithm, most changes occur from bad zone to medium zone, and this we believe was the most interesting result in our project. We will probably need to explore more to validate this result, but it seems a strong point for further delve into this area.

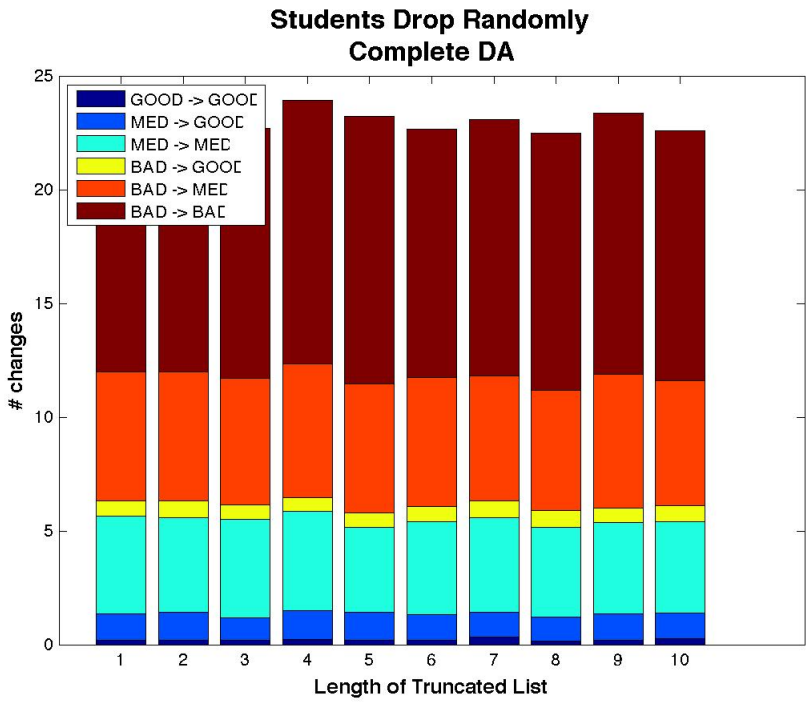

Figure 9: Changes across zones using complete DA again i.e. for a stable match

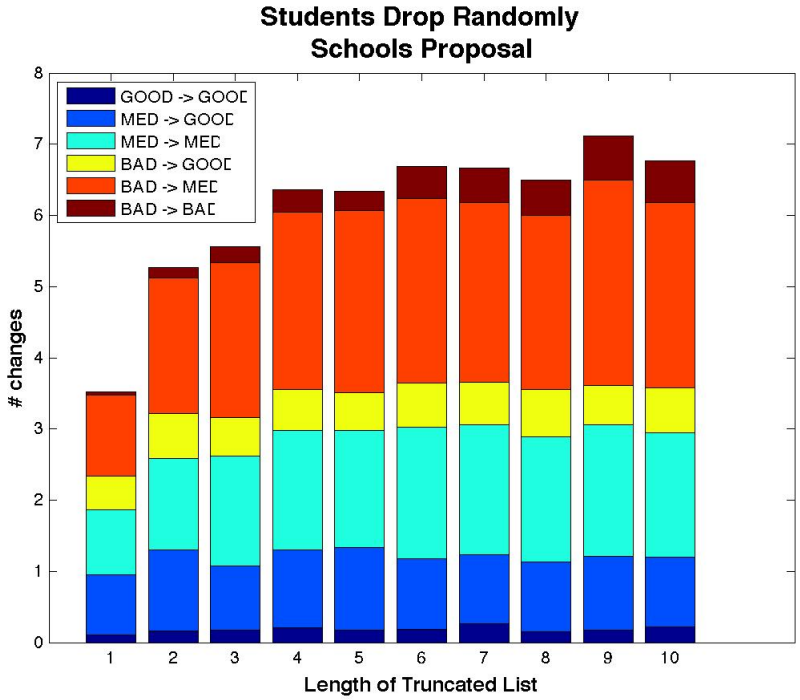

Figure 10: Changes across zones using truncation of preference list algorithm

6 Scope for further work

- In the truncation preference list algorithm, most changes occur from bad zone to medium zone, and this we believe is the most interesting result in our project. We will probably need to explore more to validate this result, but it seems a strong point to further delve into this area.

- There is a trade-off between application workload and stability while deciding the truncation length in our algorithm. We need to explore mechanisms that produce an optimal truncation length l^* for our algorithm. It will be an interesting problem to first quantify the application workload in the system and then compare it to the loss of stability due to blocking pairs.

- Research reveals that in New York City public high school admission process more than 35,000 students enroll late and around 15000 students drop out after having been assigned schools.The students who enroll late include new immigrants, over age students, homeless youth, special needs students etc. Allocating these students along with students already in the system is an interesting challenge. Analyzing dropouts in many-many matchings are a lot more complicated. Also, in reality school admissions involve many constraints such as,
  (i) students must be matched within a particular zone
  (ii) students must satisfy certain score metrics
  (iii) varying capacity constraints

- Currently, late students are assigned to low performing high schools. Better allocations for late students along with existing students in a way that ensures no party predominantly ends up in low performing schools will be an interetsing problem to explore. Further research in this area will help us better understand if dropouts can be handled in an efficient manner.

7 Lessons Learned

- This project gave us some insights about modelling real world scenarios. We understood the complexity of problems that occur in the real world.

- We learned how to approach problems that involve multiple varying parameters. Our initial confusion and frustrations in trying to figure out a reasonable way to simplify the problem taught us how to think in a focused manner.

- We learned that although once the framework and the parameters that define a framework are well defined, analyzing the results is not so straighforward. This project gave us a chance to understand the rigorous mathematical procedures that are necessary to prove or disprove claims based on analysis of results.

- The school choice problem alone is more complex than we thought it was. In general, we came to realize that market matching is a vast field and this project also gave us an opportunity to explore ongoing research in this field.

8 Conclusion

In summary, we intended to analyse how dropouts in the school choice problem influence the system. We hoped to analyze the dynamics of the system and help determine efficient ways of dealing with problems that arise in the case when students dropout. Specifcally, we tried to come up with an algorithm that can reach a semi-stable match when compared to the DA algorithm, but with substantially lesser workload of applications in the system.

As part of this project, we defined the problem and its scope, we conceptualized a framework to aid problem solving and we simulated the framework to be as close to the real world as possible. We explored a basic mechanism of creating a semi-stable match (truncation of preference list) and utilised our defined framework to gain some insight into the working of our algorithm.

Given this, we analyzed the results obtained, drew inferences and arrived at a conclusion that there is enough scope for improvement in the system. The simulations we ran were very basic and our assumptions were simplistic. However, modelling a school choice problem and relating to the real world, requires a more rigorous framework with fewer assumptions.

After these considerations, we as a team, believe that we were able to justify the objectives we had set for ourselves at the beginning of the project. However, we must admit that this exposure to research has made us realize that there is a lot of room for improvement and also that there is a lot to learn.

We thank Prof. Sethuraman and Shyam Chandramouli for their support and encouragement throughout the course and this project.

References

[1] A. Abdulkadiroglu. School choice. In *The Handbook of Market Design*. Oxford University Press, 2013.

[2] A. Abdulkadiroğlu, P. A. Pathak, and A. E. Roth. The new york city high school match. *American Economic Review*, pages 364–367, 2005.

[3] A. Abdulkadiroğlu, P. A. Pathak, A. E. Roth, and T. Sönmez. The boston public school match. *American Economic Review*, pages 368–371, 2005.

[4] P. Coles and R. Shorrer. Optimal truncation in matching markets. 2013.

[5] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, pages 9–15, 1962.

[6] T. S. Muriel Niederle, Alvin E. Roth. The New Palgrave Dictionary of Economics, 2nd edition, Palgrave Macmillan. `http://www.stanford.edu/~niederle/Palgrave%20Matching.Approved.pdf`, 2007. [Online; accessed 8-May-2014].

[7] N. F. Phil Gloudemans. Annenberg Institute for School Reform, Brown University. `http://annenberginstitute.org/news/2013/10/10/research-reveals-late-enrolling-new-york-city-students-disproportionately-assigned-s`, 2013. [Online; accessed 8-May-2014].