# Joint Seat Allocation: An algorithmic perspective

Technical Report

Surender Baswana  Partha P. Chakrabarti  V. Kamakoti  Yash Kanoria
Ashok Kumar  Utkarsh Patange  Sharat Chandran

September 2015

## Abstract

Until 2014, the admissions to the Indian Institutes of Technology (IITs) were conducted under one umbrella, whereas the admissions to the non-IIT Centrally Funded Government Institutes (CFTIs) were conducted under a different umbrella, the Central Seat Allocation Board (CSAB). The same set of candidates were eligible to apply for a seat in each of the two sets of institutes, and several hundred candidates would indeed receive two seats from the two different sets. Each such candidate could use at most one of the seats, leaving a vacancy in the other seat; this would be noticed much later, in many cases after classes began. Such seats would either remain vacant or would be reallocated at a later stage, leading to inefficiency in seat allocation in the form of unnecessary vacancies, and also misallocation of seats (e.g., a particular CSAB seat could be offered to a candidate A, despite denying the same seat earlier to a candidate B with better rank, who had meanwhile taken some IIT seat). The two umbrellas also operated under different admission windows, with the net result that classes would begin later in the academic year, compared to, say colleges offering the sciences, or the arts.

In 2015, a new combined seat allocation process was implemented to resolve some of the issues. The process brought all CFTIs under one umbrella for admissions – 86 institutes with approximately 34000 available seats. Each candidate submitted a single choice list over all available programs, and received no more than a single seat from the system, based on the choices and the ranks in the relevant merit lists.

We describe the new Multi-Run Multi-Round Deferred Acceptance scheme that was used for this combined allocation process. Crucially, unlike the 2014 and earlier admissions processes, the scheme seamlessly handles multiplicity of merit lists across different institutes and programs; indeed every program may have a separate merit list, and these lists need not have any relation with each other. In addition, the scheme has several other desired objectives. The scheme makes it safe and optimal for candidates to report their true preferences over programs. The seat allocation produced does not waste seats and is fair in the sense that it does not give a seat to a lower ranked candidate when it was denied to a higher ranked candidate. Further, the allocation is optimal in a formal sense, providing each candidate with the best possible seat subject to fairness. Without compromising on these tenets, the scheme factors in various business rules including reservations for different birth categories, reservations for home state candidates, and rules regarding dereservation when sufficient candidates are not available. The scheme also factors in changes that are inevitable when it is discovered, for instance, that candidates have inadvertently or otherwise incorrectly declared their birth category, or when it is discovered that the qualifying criteria have been incorrectly recorded by state education authorities. Our scheme is inspired by the single run Deferred Acceptance algorithm attributed to Gale and Shapley.

In this report, and based on the experience of running the scheme in July 2015. we also present several important implementation considerations, and some recommendations for the future. We strike two notes of caution that is necessary to maximally improve the efficiency of the seat allocation: (i) the calendar should be suitably constructed so that a common allocation is implementable both in theory and practice. (ii) educating candidates to fill choices properly is outside the scope of this report but an equally crucial, and difficult task that must be given attention.

In the end, the ability to gracefully handle multiple merit lists gives us hope to express optimism that all undergraduate engineering admissions in the country, beyond the CFTIs, can beneficially use the suggested scheme.

# Contents

# Chapter 1

# Introduction

Allocation of seats (earlier, counseling) to colleges is a key step in the admission process from an institutional perspective. It decides the final fruit of all the hardships taken by a candidate over years of schooling and special preparations for competitive examinations. Importantly, this decision shapes the candidate's future career. Thus, a lot of care needs to be taken to ensure that the right fruit is delivered to the right candidate.

Two different points of view need to be considered to correctly allocate seats – the point of view of the candidates, and the point of view of the participating institutions. All candidates should get the best available choice, and every institution should fill all seats in the varying courses they offer. Consistent efforts have been taken by different organizations in charge of counseling to satisfy both viewpoints.

Many institutions have grown to a stature of national importance. Each of them have their own entrance examination for admission and also have a separate rank list and a separate allocation process. Previously, the candidate filled a choice sheet for each of these institutions, or sometimes, group of institutions. Thus, there was no provision in the past for the candidate to list her choices in a single choice sheet across all institutions. Every choice sheet he filled could indicate only the preferences of the candidate among available choices in the institution he applies to. As a result, the candidate who filled $K$ choice sheets may receive admission to up to $K$ programs from which he has to select one. However, as counseling dates are not synchronized, and there is no legal provision for overbooking seats, the candidate may rationally block more than one seat by paying required fees for safety reasons. At most one of these seats is occupied by the candidate, whereas the remaining seats go vacant. As different allocation processes have no mechanism to track this, many seats in unpopular (in the eyes of the young candidates) courses end up being unfilled. At the same time, many candidates in the waiting lists could not be admitted as the decision of the deserting candidates who have been allotted a course is known only after the courses start and that becomes too late.

Different institutions and admission boards have tried different mechanisms to alleviate this problem, which in our opinion are not very effective, leading to allocative inefficiency as well as logistical difficulties for candidates and institutes. Some of these mechanisms are:

1. Spot admissions after classes begin: Many institutions do this admission within a very short window of time. It becomes practically impossible for the candidates to move around thousands of kilometers attending one spot round after another. Not only there are severe logistical costs, there is also inappropriate allocation of seats.[1]

2. Penalty: When a candidate blocks a seat in a course the amount he pays may not be refunded if he does not accept the admission offer; further the candidate may not be allowed to write the entrance examination for the corresponding institution the next year. Such penalties may be infeasible[2] or may not be effective.[3]

So what is the solution? Rather than passing the buck to the candidate, one can ask "Why don't the different admission bodies join together and solve the problem?" With this view this document describes a method to join two of the major admission process (for the National Institutes of Technology (the NITs) and the Indian Institutes of Technology (the IITs), and conduct a common allocation procedure. The common procedure was implemented in July 2015 based on an earlier version of this document published in Feb 2015.

The process which we formulated conducts a single window multi-round counseling for admissions to the Centrally Funded Technical Institutions (CFTI) based on the JEE (Main) and JEE (Advanced) examination. Note that there are multiple sets of merit lists.[4] We note that each set comprises merit lists for the General (GEN) category, Other Backward Castes Non-Creamy Layer (OBC-NCL) category, Scheduled Caste (SC) category, Scheduled Tribe (ST) category, and Persons With Disability (PwD) category.[5] The candidate fills only one choice list indicating her choices from programs offered across all CFTIs. Thus, the admission bodies get a global view of the choices and, in principle, a better allocation may appear possible that incorporates both the candidate viewpoint and the institutional view point.

However, the problem of common counseling mentioned is non-trivial. In the current context, we require a careful adaptation of existing methods based on the celebrated Deferred Acceptance algorithm by Gale and Shapley [4, 6, 1]. Our scheme is truthful, namely the scheme makes it safe/optimal for each candidate to report her true preferences over programs in her submitted choice list. The seat allocation produced does not waste seats and is fair/stable in the sense that it does not give a seat to a lower ranked candidate that was denied to higher ranked candidate. Further, the allocation is optimal in a formal sense, providing each candidate with the best possible seat subject to fairness. See Section 2.2 for formal definitions.

There are crucial timing related issues in the present context that are important in the theoretical and practical design of the process. These have a crucial impact on the feasibility

[1] In fact, in 2014 candidates holding a CSAB seat were not permitted to participate in the CSAB spot round.

[2] The courts have typically insisted on a full refund of fees.

[3] Monetary penalties may not be effective. Being debarred from a particular entrance exam next year is immaterial if the candidate is not planning to write the exam anyway, which is true for many candidates who block seats.

[4] Different merit lists need not have any relationship with each other. They may or may not be based on the same set of examinations.

[5] While the existence of these categories is significant, the definitions of these categories is outside the scope of this document.

of conducting combined admissions across a set of institutes, as well as the allocative efficiency of the process. The first practical issue is that each of the merit lists needed for admissions to all involved programs must be ready before admissions are conducted.[6] This may limit the extent to which the set of involved institutes can be expanded. The second issue is that there are always going to be colleges that are not part of the system, but are of interest to candidates who are participating.[7] Thus, a substantial fraction of candidates will vacate the allotted seat, in many cases even after paying the fees and officially accepting the allocation. We discuss methods to minimize the impact of such vacancies in Chapter 7.

## 1.1 Organization of the report

Chapter 2 presents the problem formally along with introducing some basic notations and terminologies to be used in the rest of the report. Chapter 3 first presents the deferred acceptance (DA) algorithm in its simplest form. Later it describes the version that handles two important aspects of multiple rounds, namely, Min-Cutoff and Cat-Change. Chapter 4 describes some fundamental business rules for admissions into three types of institutes, and how they are incorporated using virtual programs and virtual preference tables. Chapter 5 presents the complete description of the multiple round DA algorithm. Chapter 6 presents the way de-reservation is handled in each round by executing multiple runs of our algorithm. In view of this consideration, we term our scheme as MRDA, a multi-run deferred acceptance scheme. Chapter 7 presents the recommendations of the technical committee based on the outcome of the Joint Seat Allocation 2015. These recommendations should be considered seriously for future years.

Our scheme was first described in Feb 2015, and later implemented in July 2015. In Chapter 8, which is the Appendix, we provide the details of the following aspects of MRDA algorithm and the Joint Seat Allocation 2015.

1. Validation modules required to ensure that the output meets all business rules.
2. Some statistics of Joint Seat Allocation 2015.
3. Implementation details of MRDA.

The appendix also contain some notes on the proof of correctness, computational considerations, and an alternate way to handle de-reservation in a single run of MRDA.

[6] In 2015, this posed a real problem. The CSAB merit list preparation required board exam marks, which were obtained only on July 1 (after intense efforts), whereas the first round had been planned to begin on June 25th; thus the first round had to be postponed by about a week.

[7] There are about 1.5 million engineering seats in India, whereas only about 34,000 of these seats were filled using the system described here. Even if, hypothetically, all engineering colleges in the country become part of a single combined system, there will still be candidates who choose not to study engineering, or to study outside the country.

# Chapter 2

# Preliminaries

## 2.1 Background and Challenges

In the case of a single merit list, the candidates are sequentially ordered, and allotment of seats is done by processing candidates in the same order and allowing each candidate to choose from among the seats that are still available.[1] For example, consider three candidates, A, B and C. If in the NIT merit list, they are ranked as A, B and C, then, given the choices of A, B and C for different branches, we give preference to A over B and C. Likewise, we give preference to B over C. The allotment process first allocates a seat to A, then to B, and finally to C.

In practice, the process is much more intricate as one has to take care of various categories like GEN, OBC-NCL, SC, ST, PwD, their cut-offs, then recouping of unallocated OBC-NCL and PwD seats. Nevertheless, a solution can, and has been devised.

Unfortunately, if we have multiple merit lists, the issues are much more complex. Consider a simple scenario of three merit lists, namely those of the IITs, the NITs and a merit list for the Architecture (ARCH) program in which an additional examination needs to be cleared. Again consider our 3 candidates, A, B and C. Let the three merit lists be as follows:

```
NIT merit List: (1, A), (2, B) and (3, C) (as mentioned earlier)

IIT Merit List: (1, B), (2, C), and (3, A)

ARCH merit List: (1, C), (2, A) and (3, B)
```

Note that now, unlike in the case of a single merit list, there is no overall strict ordering among A, B and C as their relative performance is different in the three examinations. In practice, we may have several merit lists covering tens of examinations and lakhs of candidates.

Now consider a joint seat allocation process where each candidate fills up a single choice sheet. An example is shown below where we assume, for illustration, that there is only a single seat available in each program.

[1] This is known as serial dictatorship in the literature [7].

```
A: 1: IIT, 2: ARCH, 3: NIT

B: 1: ARCH, 2: NIT, 3: IIT

C: 1: NIT, 2: IIT, 3: ARCH
```

Note that the choices of the candidates are their personal preferences and may not have any linkages to their relative ranks in different merit lists.

How do we allocate seat? The first condition that comes to one's mind is fairness.

Fairness may appear straight forward: A candidate's choice for a program should be honored in the order of merit for that program. In the example, since candidate B does express a preference to go to NIT, it should never be the case that any candidate in the NIT merit list worse than B displaces B, and denies B a seat; B may however be given a seat which she prefers to NIT, say ARCH.

Given multiple merit lists, there may be several solutions for the same choices that satisfy fairness. Consider three different allocations:

```
Allocation 1: A (NIT), B (IIT), C (ARCH)

Allocation 2: A (ARCH), B(NIT), C (IIT)

Allocation 3: A (IIT), B (ARCH), C (NIT)
```

Allocation 1 is fair because A is ranked first in NIT merit list, B is ranked first in IIT merit list and C is ranked first in ARCH merit list. But Allocation 1 gives each candidate their worst choice. In fact, should A and B meet each other, they would be willing to swap their allocation! Allocation 2 is also fair, and better from the candidate's point of view than Allocation 1 because it gives everyone their second choice. Finally Allocation 3 is also fair, and better than both Allocations 1 and 2 from the candidate's point of view because everyone gets their first choice. This brings about the following scientific question, given that fairness is not good enough.

> Is there a "best" choice among the different available fair allocations, and how can one compute it?

We have highlighted these questions through a simple scenario. The solution gets complicated when we have the practical situations of lakhs of candidates having to fill a common choice list over multiple merit lists where each institution have their own business rules in the handling of state and central SC/ST/OBC-NCL/PwD categories, multi-session allocation, and spot allocations.[2]

[2]In the simple case with no categories, [4] shows that there is a unique candidate optimal fair allocation, which simultaneously gives each candidate their best possible fair allocation. Further, this allocation can be computed using the candidate-proposing deferred acceptance algorithm.

This document presents details of design, analysis and implementation of a scalable solution to the problem mentioned above.

## 2.2 Formal Problem Statement

We start with a simplified problem with multiple merit lists for the different programs[3]. Differing programs may have the same merit list, or may have different merit lists. As discussed earlier, the problem is complex, but we term it 'simplified' for now because (for example) there are no quotas or ties in the presentation of this section.

Let $\mathcal{P}$ be the set of programs, with $P = |\mathcal{P}|$ being the number of programs. For a program $p \in \mathcal{P}$, let $c(p)$ denote the number of seats in $p$. Let $\mathcal{A}$ denote the set of applicants (or candidates), with $A = |\mathcal{A}|$ being the number of candidates. Each candidate is allowed only one seat in the system. Candidates are asked to submit a choice (or preference) list over programs; the choice list is a strictly ordered list containing any subset of the programs in $\mathcal{P}$.

We denote the preference list of candidate $x$ by

$$\textsc{Pref}(x) = p_{x,1}, p_{x,2}, \ldots, p_{x,n(x)} \tag{2.1}$$

which means that candidate $x$ has listed $n(x)$ programs, with program $p_{x,1}$ being her top choice, program $p_{x,2}$ being her next choice and so on. The candidate is asked to list only programs she is interested in.

Each program must submit its capacity $c(p)$, as well as its merit list of candidates, which is a strictly ordered list containing a subset of the candidates in $\mathcal{A}$. We denote the merit list of program $p$ by

$$\textsc{Merit}(p) = x_{p,1}, x_{p,2}, \ldots, x_{p,m(p)} \tag{2.2}$$

which means that program $p$ has ranked $m(p)$ candidates in its merit list, with candidate $x_{p,1}$ ranked 1, candidate $x_{p,2}$ being ranked 2, and so on.

Let $\mu(x) \in \mathcal{P}$ be the program allotted to candidate $x$ by some mechanism, with some candidates possibly not getting any seat, in which case we write $\mu(x) = \phi$. Denote the overall allocation by $\bar{\mu} = (\mu(x))_{x \in \mathcal{A}}$. We want a mechanism with the following properties:

1. Fairness: Suppose candidate $x$ is allotted program $p$. Then for any other candidate $y$ such that $y$ has a better (smaller) rank than candidate $x$ in the merit list of $p$, the allocation of $y$ should be $p$ or some other program that $y$ prefers to $p$.[4]

   Further, the mechanism must ensure that a candidate is not allotted a program that she did not list, and that no program is allotted to a candidate that was not a part of the merit list of the program.[5]

[3] One may view the word 'program', 'course', 'branch' and 'programme' interchangeably in the rest of the document. For ease of understanding at this stage, one may think of a program as a college having a single course, say, Electrical Engineering in IIT Kharagpur.

[4] This property is called stability in the literature [4].

[5] This property is called individual rationality in the literature [7].

2. Optimality: There does not exist any other allocation $\bar{\mu}'$ that satisfies the Fairness property, and provides any candidate $x$ with an allocation she prefers to $\mu(x)$ based on her preference list.

3. Truthfulness: The mechanism must make it optimal for candidates to report their true preferences.

A priori it is unclear that a mechanism satisfying all these properties exists. However, it turns out that such a mechanism does exist, and was constructed by Gale and Shapley [4]).

## 2.3 Previous Work

An initial solution to the simplified problem was proposed by Gale and Shapley in 1962 [4] by formulating it as a "Stable marriage problem". The proposed solution was shown to have a multitude of desirable properties including fairness and candidate optimality [4], and truthfulness [3] (no student or group of students can benefit from misreporting their preferences, see Theorem 14). The Gale and Shapley mechanism has been adapted and implemented successfully in a multitude of real world settings, e.g., the National Residency Matching Program (NRMP) (running in the USA since 1951, redesigned in 1999 [6]), New York City high school admissions since 2003 [1], and school and college admissions in Hungary [2].

However, our problem involves a variety of business rules governing flows of different systems that need to be streamlined for evolving a sound process of common allocation. We need to suitably adapt the Gale-Shapley mechanism to incorporate these business rules while retaining all these desirable properties. In the sequel, we demonstrate these and come up with a practical algorithm.

# Chapter 3

# A Combined Seat Allocation Scheme

Our proposed mechanism first collects information from the participating programs and candidates. It then uses this information to produce an allocation of seats.

## 3.1 Initial Information collection

Each participating program provides two pieces of information:

- The capacity $c(p)$, i.e., number of available seats
- A merit list of eligible candidates (Equation 2.2). The purpose of a merit list of program is to compare two candidates. For implementation of the algorithm, it is quite possible that this merit list is not explicitly computed. Instead, the relative order of any two candidates can be determined from the information associated with each of them (marks, birth category, PwD status).

Each candidate (who is eligible for at least one program) enters a preference list (Equation 2.1) of programs for which she is eligible, with the first entry being her most preferred program, the next entry being her next most preferred program, and so on.

We now describe the algorithm for allocating programs to candidates which incorporates all the business rules of CFTIs and has the three properties, namely, fairness, candidate-optimality, and truthfulness.

The Deferred Acceptance (DA) algorithm that forms the core of the joint allocation is described in the following section.

## 3.2 Basic DA algorithm

We stress that in the DA algorithm, the "applications", insertions into "waitlists" and "rejections" mentioned are all merely part of the algorithm, and do not actually involve any participation from the candidates and programs in the real world. That is, the algorithm internally generates

applications, using the information that the candidates and programs have provided. We now state the algorithm in words. See Section 3.3 for complete pseudocodes.
Input:

- For each program, its capacity and rank list of eligible candidates.
- For each candidate, a preference list of programs.

Algorithm:

1. All candidates apply (in any order) to the first program in their preference list.
2. Each program $p$ considers the applications it has received. Applications from candidates who are not eligible are immediately dropped. Let the capacity of the program be $c(p) > 0$. If the program has received $c(p)$ or fewer eligible applications, then it retains all candidates on a waitlist. Otherwise, it ranks the candidates[1] making these requests (as per the merit list of the program) and retains only the $c(p)$ best candidates on its waitlist, and rejects other candidates.

   If no rejections are made by any program, the algorithm terminates.
3. Only rejected candidates apply (in any order) to the next program on their list, if any, and the algorithm returns to Step 2.

   If not even a single application is generated, then the algorithm terminates.

Output: When the algorithm terminates, the (final) "wait list" for each program $p$ constitutes the candidates admitted to program $p$.

We present complete details of the DA algorithm through pseudocode in the following sections. A formal proof of correctness and other computational considerations of the algorithm are described respectively in Section 8.1 and 8.2 in Appendix.

## 3.3 Pseudocode

Denote rank of candidate $x$ with respect to $\textsc{Merit}(p)$ by $\textsc{Rank}(x, p)$. For a list $l$, denote the number of entries in the list by $\textsc{Length}(l)$.

We narrate two versions of the pseudocode. In the first version the assumption is that the entire seat allocation happens in a single round. It is presented to understand the general flow of the algorithm. The second version assumes a variety of complications; in particular, it allows multi-round scenarios as described in Chapter 5. Further, it also allows complicated scenarios when the ranks of candidates can change between rounds (due to revision of marks), and when category of candidates change between rounds, and these two cases are treated differently by the business rules.

[1] assuming for the moment that equal ranks do not exist; ties are handled in a later section.

**Algorithm 1** Deferred Acceptance Simple Version

INPUT:
Candidates $\mathcal{A}$, Programs $\mathcal{P}$
Preference list $\textsc{Pref}(x)$ for each $x \in \mathcal{A}$
Capacity $c(p)$ and merit list $\textsc{Merit}(p)$ for each $p \in \mathcal{P}$

OUTPUT:
For each candidate $x \in \mathcal{A}$, the allocation $\mu(x) \in \mathcal{P} \cup \{\emptyset\}$
Also for each program $p \in \mathcal{P}$, the list of admitted candidates $\textsc{WaitList}(p)$

**for all** $p \in \mathcal{P}$ **do**
    Create an empty ordered list $\textsc{WaitList}(p)$ that will consist of
    candidates ordered by their rank in $\textsc{Merit}(p)$
**end for**
Create an empty queue $\mathcal{Q}$
**for all** $x \in \mathcal{A}$ **do**
    $i(x) \leftarrow 1$ ▷ Initialize list position to 1.
    **if** $\textsc{Length}(\textsc{Pref}(x)) > 0$ **then**
        $\textsc{Enqueue}(x, \mathcal{Q})$ ▷ $x$ enters queue $\mathcal{Q}$
    **end if**
**end for**
**while** $\mathcal{Q}$ is non-empty **do**
    $x \leftarrow \textsc{Dequeue}(\mathcal{Q})$ ▷ $x$ is any candidate removed from queue $\mathcal{Q}$
    $p \leftarrow p_{x,i(x)}$ ▷ $x$ applies to program $p_{x,i(x)}$
    **if** $x$ is not eligible for $p$ **then**
        $\textsc{Reject}(x)$
        **continue**
    **end if**

```
19:     if Length(WaitList(p)) = c(p) then                ▷ The waitlist is full
20:         y ← Last candidate in WaitList(p)
21:         if Rank(x, p) < Rank(y, p) then
22:             Remove y from WaitList(p)
23:             Reject(y)
24:             Insert x into ordered list WaitList(p) at correct location
25:         else
26:             Reject(x)
27:         end if
28:     else
29:         Insert x into ordered list WaitList(p) at correct location
30:     end if
31: end while
32: for all x ∈ 𝒜 do
33:     if p_{x,i(x)} exists in Pref(x) then
34:         μ(x) ← p_{x,i(x)}
35:     else
36:         μ(x) ← ∅
37:     end if
38: end for
39: return μ(x) for all x ∈ 𝒜 and WaitList(p) for all p ∈ 𝒫
40:
41: function Reject(x)
42:     Increment i(x)
43:     if p_{x,i(x)} exists in Pref(x) then              ▷ x wants to apply further
44:         Enqueue(x, 𝒬)                                 ▷ x enters queue 𝒬 again
45:     end if
46: end function
```

### 3.3.1 Multi-Round Scenario

Many candidates who obtain seats in the first round of seat allocation will surrender or reject their respective seats at a later stage. In order to utilize these surrendered seats, the business rules allow multiple rounds of seat allocation. Our pseudocode allows for additional optional inputs in order to implement second and later rounds of seat allocation. The full description of how to use Algorithm 2 to implement the seat allocation in each round is provided in Chapter 5.

- MIN-CUTOFF($p$) for each $p \in \mathcal{P}$. This quantity is used in second and later rounds of allocation (see Chapter 5 for details). Intuitively, a candidate with rank better than or as good as MIN-CUTOFF($p$) will never be rejected by $p$ regardless of the capacity of $p$. Candidates offered a seat in the first round will thus be no worse off in subsequent rounds. Note that there might also be candidates whose rank improve (or become worse) in subsequent rounds. In such cases, the allocated seat shall be what the candidate would have got on the basis of the revised rank in the first round. This seat can be a supernumerary seat.

  We use the notation $x|y$ to indicate that RANK($x, p$) is as good as, or better than MIN-CUTOFF($p$), and RANK($y, p$) is worse.

- Another optional input is the list CAT-CHANGE which consists of candidates who might have had their category changed. For example, a person presumed to be with an OBC-NCL in an earlier round may be reclassified as a general category candidate. As per the business rules, in this case, the candidate will be allocated a seat in the best (in terms of the filled-in choices) possible choice of academic program that has unfilled seats and supernumerary seats are NOT created.

- The starting position on the preference list $i(x)$ for each $x \in \mathcal{A}$ and the current queue $\mathcal{Q}$ of candidates to be processed. These starting positions inputs are meant as an optimization mechanism and can be ignored in the first reading of the pseudo-code.

### 3.3.2 Remarks

Although we have described Algorithm 2 keeping in mind CAT-CHANGE and MIN-CUTOFF, the key take away of the algorithm is that we have kept two considerations in mind in the multi-round scenario. In one case, candidates get what may be termed as the MIN-CUTOFF benefit: seats obtained in an earlier round are guaranteed, and so supernumerary seats may be created. In the second case, candidates may have the CAT-CHANGE penalty: although they may clear the MIN-CUTOFF they are denied the benefit of creating supernumerary seats. The two types of cases can be mapped in a variety of scenarios; for example, CAT-CHANGE may be replaced by credential change candidates, i.e., candidates whose credentials were changed for some reason moving from round $i$ to round $i + 1$. See Section 5.5 for more details.

**Algorithm 2** Deferred Acceptance (Full version allowing multiple rounds)

INPUTS:
Candidates $\mathcal{A}$, Programs $\mathcal{P}$
For each $x \in \mathcal{A}$:
Preference list $\textsc{Pref}(x)$
Optional input: integer $i(x)$. Default value $i(x) = 1$. (Start from beginning of preference list by default.)
For each $p \in \mathcal{P}$:
Capacity $c(p)$ and merit list $\textsc{Merit}(p)$.
Optional input: $\textsc{WaitList}(p)$.
Optional input: $\textsc{Min-Cutoff}(p)$. 0 by default.
Optional input: $\mathcal{Q}$ a queue of candidates. By default contains all Indian candidates $x$ with $\textsc{Length}(\textsc{Pref}(x)) > 0$.
Optional input: Cat-Change. A list of candidates whose category has changed (empty by default)
OUTPUTS:
For each candidate $x \in \mathcal{A}$, the allocation $\mu(x) \in \mathcal{P} \cup \{\emptyset\}$ and $i(x)$.
Also for each program $p \in \mathcal{P}$, the list of admitted candidates $\textsc{WaitList}(p)$

1: ... Everything up to Line 11 in Algorithm 1
2: **while** $\mathcal{Q}$ is non-empty **do**
3: $x \leftarrow \textsc{Dequeue}(\mathcal{Q})$ ▷ $x$ is any candidate removed from queue $\mathcal{Q}$
4: $p \leftarrow p_{x,i(x)}$ ▷ $x$ applies to program $p_{x,i(x)}$
5: **if** $x$ is not eligible for $p$ OR $c(p) = 0$ **then**
6: $\textsc{Reject}(x)$
7: **continue** ▷ move to next person in $\mathcal{Q}$
8: **end if**
9: $y \leftarrow$ Last candidate in $\textsc{WaitList}(p)$ ▷ y can be null
10: $w \leftarrow$ Last candidate in $\textsc{WaitList}(p) \cap$ Cat-Change ▷ $w \stackrel{?}{=} y$

```
11:     Insert x into ordered list WaitList(p) at correct location          ▷ may remove x
12:     if Length(WaitList(p)) ≤ c(p) then continue                         ▷ space in program
13:     end if
14:     if w = ∅ then                          ▷ y ∉ Cat-Change (but x?) Also, now y ≠ ∅
15:         if (x, y)| then
16:             if x ∈ Cat-Change then                    ▷ both x and y clear Min-Cutoff
17:                 PenaltyRemoveAndReject(x, p)                            ▷ Still we reject x
18:             end if
19:         else
20:             Sort(x, y) into q, r                                        ▷ r is the worse
21:             RemoveAndReject(r, p)
22:         end if
23:     else                                                                ▷ w ≠ ∅
24:         if y is not w then                                ▷ x, y, w are distinct people
25:             if (x, y, w)| then
26:                 if x ∈ Cat-Change then
27:                     Sort(x, w) into q, r                                ▷ r is the worse
28:                     PenaltyRemoveAndReject(r, p)                        ▷ We reject r!
29:                 else
30:                     PenaltyRemoveAndReject(w, p)                        ▷ Reject w!
31:                 end if
32:             else
33:                 Sort(x, y, w) into s, q, r                              ▷ r is the worst
34:                 RemoveAndReject(r, p)
35:             end if
36:         else                                                            ▷ y and w are same!
37:             if (x,w)| then
38:                 if x ∈ Cat-Change then
39:                     Sort(x, w) into q, r                                ▷ r is the worse
40:                     PenaltyRemoveAndReject(r, p)                        ▷ We reject r!
41:                 else
42:                     PenaltyRemoveAndReject(w, p)                        ▷ Reject w!
43:                 end if
44:             else
45:                 Sort(x, y) into q, r                                    ▷ r is the worse
46:                 RemoveAndReject(r, p)
47:             end if
48:         end if
49:     end if                                                              ▷ End w ≠ ∅
50: end while
```

```
51: for all x ∈ A do
52:     if i(x) ≤ Length(Pref(x)) then
53:         μ(x) ← p_{x,i(x)}
54:     else                                          ▷ x reached the end of her list
55:         μ(x) ← ∅
56:     end if
57: end for
58: return μ(x), i(x) for all x ∈ A and WaitList(p) for all p ∈ P
59: function Reject(x)
60:     Increment i(x)
61:     if i(x) ≤ Length(Pref(x)) then                ▷ x wants to apply further
62:         Enqueue(x,Q)                              ▷ x enters queue Q again
63:     end if
64: end function
65: function RemoveAndReject(w, p)
66:     Remove w from WAITLIST(p)
67:     Reject(w)
68: end function
69: ▷ The next function is identical but is written for clarity
70: ▷ to indicate penalty for category change people.
71: ▷ This function will change when there are ties
72: function PenaltyRemoveAndReject(w, p)
73:     Remove w from WAITLIST(p)
74:     Reject(w)
75: end function
```

In subsequent sections, we show how various business rules of IITs and NITs (such as handling quotas, applicants with equal ranks, etc.) can be incorporated seamlessly in this algorithm.

# Chapter 4

# Business Rules

As discussed earlier, the Government of India recognizes quotas for certain birth categories. In general, a student applies for a program rather than for a seat in a particular category; she may be eligible for multiple seat categories. The so-called business rules [5] describe how to allocate seats in such scenarios. In this and subsequent sections, we describe how these rules are to be incorporated into the proposed algorithm. The following is a broad classification of the variety in allocation strategies.

1. Allocation based on reservations based on birth-categories.
2. Allocations for persons with disabilities (PwD).
3. Allocation for candidates who are not nationals of India (International students). This particular section applies only to the IITs.
4. Allocation for certain children of military personnel killed in military operations (DS candidates). This particular section applies only to the IITs.
5. Allocation for students who cannot be distinguished on merit (multiple candidates with the same rank).
6. Allocations for candidates who fail to clear certain minimum thresholds but can be groomed for admission a year later (preparatory course (PC) candidates). This particular section applies only to the IITs.
7. Allocations based on reservations on the residency state of a candidate. This particular section does not apply to the IITs.
8. Allocations based on de-reservation of seats.

Allocation based on de-reservation of seats (Item 8), is presented in full details in Chapter 6. In the current chapter we describe how our DA algorithm can incorporate Items 1-7.

## 4.1 Incorporating Quotas

We first consider the important cases of 1-2 above.

### 4.1.1 Virtual programs

All candidates, irrespective of their respective categories, will declare the programs in the decreasing order of preference. However, internally we introduce virtual programs based on the quota for each category. Each virtual program will be associated with a merit list constructed out of rank lists (which in turn is based on marks obtained in the exam). Each candidate will now be associated with a virtual preference list. The concept of virtual program, together with virtual merit lists and virtual preference lists for candidates is the suggested way for handling the majority of business rules.

There will be 8 virtual programs for each actual program as per Table 4.1.

| OPEN | OBC-NCL | SC | ST |
|---|---|---|---|
| OPEN-PwD | OBC-NCL-PwD | SC-PwD | ST-PwD |

Table 4.1: Candidates are partitioned into categories and assigned a tag. A candidate can possess only one of these 8 tags. Further, for each actual program, there is a separate virtual program corresponding to each of these 8 categories.

The DS virtual program for the IITs is considered later (Section 4.4).

### 4.1.2 Virtual Preference List

The sequence of seat allocation mentioned in Rule VII of [5] are very important. These are given in Figure 4.1 and merit careful consideration.[1] For example, the sixth row states that for an OBC-NCL candidate with PwD tag, we try to fill the OPEN seats before any other seat, failing which we consider the OPEN-PwD seat before venturing into OBC-NCL seats.

### 4.1.3 Virtual Merit Lists

**Standard Rank Lists** How does one decide on which candidate to award a seat if there are competing candidates for a program? Associated with each virtual program is a merit list constructed from ranks provided by various examinations.

Quoting rule VI from [5] the following TYPES of rank lists will be prepared based on pre-defined cut-offs:

1. Common rank list (CRL): It includes candidates who are assigned the tag GEN, GEN-PwD, OBC-NCL, OBC-NCL-PwD, SC, SC-PwD, ST or ST- PwD.

[1] In this and all further preference tables, OP, OBC and PD are used instead of OPEN, OBC-NCL and PwD respectively. That is, OP refers to OPEN, OBC refers to OBC-NCL, OBC-PD refers to OBC-NCL-PwD, etc.

| Category | PD Status | Preference List | | | |
|---|---|---|---|---|---|
| GEN | N | OP | | | |
| OBC | N | OP > | OBC | | |
| SC | N | OP > | SC | | |
| ST | N | OP > | ST | | |
| GEN | Y | OP > | OP-PD | | |
| OBC | Y | OP > | OP-PD > | OBC > | OBC-PD |
| SC | Y | OP > | OP-PD > | SC > | SC-PD |
| ST | Y | OP > | OP-PD > | ST > | ST-PD |

Figure 4.1: Virtual preference list[2]for candidates when quotas are involved.

Foreign nationals are also included in the CRL prepared based on JEE (Advanced) 2015.

2. OBC-NCL rank list: It includes candidates who are assigned the tag OBC-NCL or OBC-NCL-PwD.

3. SC rank list: It includes candidates who are assigned the tag SC or SC-PwD.

4. ST rank list: It includes candidates who are assigned the tag ST or ST-PwD.

5. CRL-PwD rank list: It includes candidates who are assigned the tag GEN-PwD, OBC-NCL-PwD, SC-PwD or ST-PwD.

6. OBC-NCL-PwD rank list: It includes candidates who are assigned the tag OBC-NCL-PwD.

7. SC-PwD rank list: It includes candidates who are assigned the tag SC-PwD.

8. ST-PwD rank list: It includes candidates who are assigned the tag ST-PwD.

Once rank lists are available, the rules for allocation for various seat categories are specified in Rule VII of [5].

In each virtual queue, candidates with differing tags are eligible to participate. Thus in the OPEN virtual programs, we can find candidates with different categories from Table 4.1. The virtual merit list encodes the order of consideration for seats in each program. Although the virtual merit list appears to be exactly the same as the standard rank list at this juncture, de-reservation may be implemented by constructing virtual merit lists that suitably combine standard rank lists, cf. Section 8.3.

[2]In this and all further preference tables, OP, OBC and PD are used instead of OPEN, OBC-NCL and PwD respectively. That is, OP refers to OPEN, OBC refers to OBC-NCL, OBC-PD refers to OBC-NCL-PwD, etc.

### 4.1.4 Preparatory courses allocation

The notion of PC (Item 6 at the beginning of Chapter 4) for the IITs, will, however, uncover why only the standard merit lists are not sufficient in allocating seats. Quoting Rule XI of [5] we have the rules for seat allocation to preparatory courses applicable for every round [3].

1. Unfilled OPEN-PwD seats will be allocated to candidates in the CRL-PwD preparatory subject to no more candidates in the CRL-PwD rank list having opted for them.
2. Unfilled OBC-NCL-PwD seats will be allocated to candidates in the OBC-NCL-PwD preparatory rank list subject to no more candidates in the OBC-NCL-PwD rank list having opted for them.
3. Unfilled SC-PwD seats will be allocated to candidates in the SC-PwD preparatory rank list subject to no more candidates in the SC-PwD rank list having opted for them.
4. Unfilled ST-PwD seats will be allocated to candidates in the ST-PwD preparatory rank list subject to no more candidates in the ST-PwD rank list having opted for them.
5. Unfilled SC seats will be allocated to candidates in the SC preparatory rank list subject to no more candidates in the SC rank list have opted for them.
6. Unfilled ST seats will be allocated to candidates in the ST preparatory rank list subject to no more candidates in the ST rank list have opted for them.

In order to allot seats to PC candidates, preparatory rank lists may be prepared. This leads us to the following definition.

**Definition 1.** *We construct the extended merit list for any virtual program, cf. Table 4.1, by taking the standard rank list followed by the corresponding preparatory course (PC) rank list if*

- *such a list has been prepared, and*
- *the corresponding preparatory course exists.*

Up to six lists are possible: PC-CRL-PwD, PC-OBC-NCL-PwD, PC-SC, PC-SC-PwD, PC-ST, PC-ST-PwD. In each case, if the PC list is prepared, it is included as part of the extended merit list for the parent category: e.g., if the PC-SC rank list is prepared, the virtual merit list for SC courses (in institutes that have PC courses) will include the SC rank list followed by the PC-SC rank list. This will automatically lead to unfilled seats being offered to PC candidates. Note that the PC rank lists do not have any commonality with the standard rank lists.

In summary, at this juncture, in the case no PC exists for a program (as is the case of the NITs), the virtual merit list of a virtual program is identical to the corresponding standard rank list. On the other hand if PC courses are present, the virtual merit list of a virtual program is identical to the extended merit list.

Later on, we will see more complex virtual merit lists.

[3] In the case of IITs and ISM, seat allocation in the 4th round will be made only for preparatory courses. This consideration should be kept in mind for multi-round (Chapter 5) of seat allotment.

### 4.1.5 Updated Algorithm: DA With Quotas

Armed with virtual programs, virtual merit lists corresponding to virtual programs, and virtual preferences, we now explicitly construct the internal preference list for each candidate.

**Example**: Consider a candidate with category tag SC-PwD, who is eligible for both the SC virtual program, and the SC-PwD virtual program. This candidate has not cleared the cutoff specified for Open seat. Then for each IIT program $p$ in the preference list of the candidate, we instantiate the virtual preference from Figure 4.1 by excluding the OPEN option, and considering only the three other option corresponding to row 7.

Algorithm 2 is run for each round based on the construction of the virtual preference list for all candidates, and extended merit list for each virtual program.

It is important to note that by modifying the virtual preference tables, the standard DA with quotas can be applied based on variations in business rules on how quotas should be administered when, for example, unfilled seats are found.

## 4.2 DA with multiple candidates having same rank

Suppose multiple candidates obtain exactly the same rank. In the situation when we are forced to reject some candidate due to the possibility that the program is oversubscribed, we have to also reject (and remove) all other candidates in the waiting list who also have the same rank. This may not be always possible.

We modify lines 65-75 of Algorithm 2 as stated in Algorithm 4.2 to allow such candidates to obtain seats in the program on a supernumerary basis. We keep WaitList($p$) fully ordered at each step, by choosing an arbitrary order between candidates in WaitList($p$) who have the exact same rank as per Merit($p$).

There are two concepts involved here. First, when an attempt is made to remove a candidate $x$, a removal of all candidates with the same rank of $x$ may cause the waitlist which is initially overflowing, to now underflow. To identify this situation, we compute the length $G$ of the waitlist, and the length $L$ of the list of candidates all of whom have the same rank as $x$. If the difference between the two exceeds or matches the capacity, we are free to remove all such candidates. Second, if this condition is not satisfied, it is not possible to remove all candidates with rank of $x$. However, there might be candidates $w \in$ Cat-Change who also have the same rank. The algorithm checks if it is all right to remove only this subset.

The algorithm also considers the case when we want to remove only candidates who belong to category change. This can happen when such candidates clear the Min-Cutoff but because we are not allowed to have supernumerary seats; they need to be out of the waitlist.

We remark in passing that there is a certain simplicity, and thus beauty, in processing candidates in any order. When multiple candidates have the same rank, this poses a challenge. The proposed algorithm responds to this challenge in an elegant fashion.

**Algorithm 3** DeferredAcceptanceWithTies

Same INPUT and OUTPUT as in Algorithm 2

▷ Everything before Line 65 in Algorithm 2
**function** RemoveAndReject(w,p)
    $G \leftarrow |\text{WaitList}(p)|$
    $L \leftarrow$ number of people with rank same as w ▷ L is at least 1
    $L_1 \leftarrow$ number of people with rank same as w and such people in Cat-Change
    **if** $(G - L) \geq c(p)$ **then**
        Remove all people with rank(w)
        Reject all people with rank(w)
    **else** ▷ Removing such people may cause p to be under capacity
        **if** $(G - L_1) \geq c(p)$ **then**
            Remove all people corresponding to $L_1$
            Call Reject on all people corresponding to $L_1$
        **end if**
    **end if**
**end function**
**function** PenaltyRemoveAndReject(w,p)
    $G \leftarrow |\text{WaitList}(p)|$
    $L_1 \leftarrow$ number of people with rank same as w and such people in Cat-Change
    **if** $(G - L_1) \geq c(p)$ **then**
        Remove all people corresponding to $L_1$
        Call Reject on all people corresponding to $L_1$
    **end if**
**end function**

## 4.3 DA with International Students

The business rules for the IITs specify that foreign nationals should be given the best program for which they qualify, on a supernumerary basis. However, if the number of Indian candidates in a program is below capacity, then every foreign candidate, who requests for the program, will be accepted in the program. Algorithm 4 describes the way we need to extend our DA algorithm to incorporate this business rule. (The pseudo-code makes reference to Algorithm 1 for simplicity.)

## 4.4 DA with candidates from defense service quota

The business rule specifies that DS (defense services) candidates should be given the best program they prefer subject to the constraint that there are only two seats for DS candidates per institute. Moreover, seats to DS category candidate needs to be allocated in a preferential manner from open category seats and supernumerary allocation needs to be avoided to the

**Algorithm 4** DeferredAcceptanceWithForeignNationals

Same INPUT and OUTPUT as in Algorithm 1

```
1: ...                                          ▷ Everything before Line 6 in Algorithm 1.
2: for all x ∈ A such that x is an Indian citizen
3: ...                          ▷ Everything after Line 6 to (and including) Line 50 in Algorithm 1.
4: for all x ∈ A and x is a foreign national do
5:     i(x) ← 1                                          ▷ Initialize list position to 1.
6:     while i(x) ≤ Length(Pref(x)) do
7:         p ← p_{x,i(x)}
8:         if AllowsForeign(p) then                ▷ to check if p allow foreign nationals
9:             k ← number of Indian candidates in WaitList(p)
10:            if k < c(p) then
11:                Insert x into WaitList(p)
12:                break while
13:            else
14:                y ← Last Indian candidate in WaitList(p)
15:                if Rank(x, p) ≤ Rank(y, p) then                ▷ x qualifies for p
16:                    Insert x into WaitList(p)
17:                    break while
18:                end if
19:            end if
20:        end if
21:        Increment i(x)                                ▷ x did not qualify for p
22:    end while
23: end for
24: ...                                         ▷ Everything after Line 50 in Algorithm 1.
```

extent possible.[4] This situation applies only for the IITs.

### 4.4.1 Algorithm

To implement DS candidates, a new virtual DS program with capacity two is added per institute, namely IITK-DS, IITB-DS, IITR-DS, and so on. Only DS candidates are eligible for this virtual program. Furthermore, the preference list of each DS candidate is modified as follows. If the preference list of candidate is $\langle p_1, p_2, p_3 \rangle$, then his preference list will be modified as $\langle p_1$, Institute($p_1$), $p_2$, Institute($p_2$),...$\rangle$. Then $p_1, p_2, ...$ will be replaced by virtual programs as usual like any other candidate based on the birth category.[5]

[4]Supernumerary allocation can not be avoided, if for example, there are three candidates from DS with the same rank. There can be more complex cases.

[5]However, any seat to be considered on the DS tag will be only from the open category.

We now run the DA algorithm as in Algorithm 2 by artificially defining the capacity of the institute virtual programs by 2. At this point, we may have artificially increased the capacity of some (open category) programs by maximum of two seats per institute. Therefore, each seat which has been allotted from a DS virtual program is to be mapped to a virtual open program, and is thus processed one by one as follows. Let $s$ be one such seat, say, in IITB, and let $c$ be the candidate who has been given this seat. Suppose $c$ has been mapped to the EE program in IITB. Let $x$ be the candidate with the worst rank in the open category waitlist of the virtual program EE in IITB. If there is one more candidate in this waitlist with the same rank as the rank of $x$, then we just assign program EE to candidate $c$ and this completes the processing of seat $s$ (in a supernumerary manner). Otherwise, to take care of the over allocation, $c$ replaces $x$. We now run the DA algorithm with $x$ as the candidate not allotted any program (and so $x$ applies to the next program in her preference list). Note that $x$ may displace another candidate, and this may lead to a rejection chain.

However, in very rare cases, there is a possibility of an undesired condition arising out of this rejection chain. An example illustrating this condition along with an algorithm for handling DS candidates is described in Section 8.7.

## 4.5 DA for Admission into IITs, NITs, and other GFTIs

In Section 4.1.5 we have seen the standard DA with quotas which is applicable to all central government funded institutions. Further, we have seen various business rules, and how the core algorithm is modified to take care of various business rules. In this section we summarize the process of allocation.

### 4.5.1 Virtual Preference Table for IITs

Business Rule items 1-6 and item 8 from the list at the beginning of Chapter 4 apply. To include PC candidates, Figure 4.1 is modified slightly and is presented in Figure 4.2.

Notice that, as compared to Fig. 4.1, there are new types of categories in the last six rows, and the extended merit list comes into play. We may use the Standard DA algorithm with quotas (Section 4.1.5) with the discussions so far in Sec. 4.2, Sec. 4.3, and Sec. 4.4 if only seats in the IITs are offered.

In reality, a candidate may be eligible to apply for many programs across multiple institutions beyond the IITs. The virtual preference lists need to be considered in an appropriate fashion depending on whether a program is in the IITs or not. The next two sections describe the process for NITs and other GFTIs.

### 4.5.2 Virtual Preference Table for NITs

All items in the beginning of Chapter 4 apply, except items 3,4, and 6. There are however additional qualifiers corresponding to Item 7. We focus on Item 7. Seats for academic programs offered by NITs are divided into Home State quota and Other States quota. These additional

| Category | PD Status | Preference List | | | |
|---|---|---|---|---|---|
| GEN | N | OP | | | |
| OBC | N | OP > | OBC | | |
| SC | N | OP > | SC | | |
| ST | N | OP > | ST | | |
| GEN | Y | OP > | OP-PwD | | |
| OBC | Y | OP > | OP-PwD > | OBC > | OBC-PwD |
| SC | Y | OP > | OP-PwD > | SC > | SC-PwD |
| ST | Y | OP > | OP-PwD > | ST > | |
| PC(SC) | N | SC | | | |
| PC(ST) | N | ST | | | |
| PC(SC) | Y | OP-PwD > | SC > | SC-PwD | |
| PC(ST) | Y | OP-PwD > | ST> | ST-PwD | |
| PC(OBC) | Y | OP-PwD > | OBC-PwD | | |
| PC(GE) | Y | OP-PwD | | | |

Figure 4.2: Virtual preference list for candidates when PC candidates are involved.

quotas are reflected in the virtual preference table as shown in Figure 4.3, and do not change the core of the algorithm.

As an example, consider the virtual preference list for a OBC-NCL-PwD person whose "Home State" is Maharashtra. He may express a preference to apply to the electrical engineering program of NIT in this state, viz., the NIT in Nagpur. In this case, his virtual preference will be represented by the sixth row in the table. If his next preference is the electrical engineering program of the NIT in Tamil Nadu, then the virtual preference will be read from the 14th row of the table and he will not be eligible for the Home State quota of the NIT in Tamil Nadu. Finally, if his next preference is the electrical engineering program in IIT Roorkee, Item 7 does not apply; we refer back to the sixth row in Figure 4.1.

### 4.5.3 Virtual Preference Table for Other GFTIs

For Government Funded Technical Institutions (GFTIs) other than the IITs and the NITs, all items in the beginning of Chapter 4 except for items 3,4, and 6 apply. On the basis of the existence of quota based on residency state of a candidate, there are the following 3 types of other GFTIs and their respective virtual preference tables.

1. All India (AI) quota: If a GFTI has only AI quota, then the virtual preference table is the same as the one shown in Figure 4.1. All Indian Institutes of Information Technology (IIITs) have AI quota only. Other examples of GFTIs that have only AI quota are School

| Category | Which State | Preference List | | | |
|---|---|---|---|---|---|
| GEN | Home | OP.Home | | | |
| OBC | Home | OP.Home > | OBC.Home | | |
| SC | Home | OP.Home > | SC.Home | | |
| ST | Home | OP.Home > | ST.Home | | |
| GEN_PwD | Home | OP.Home > | OP-PwD.Home | | |
| OBC_PwD | Home | OP.Home > | OP-PwD.Home > | OBC.Home > | OBC-PwD.Home |
| SC_PwD | Home | OP.Home > | OP-PwD.Home > | SC.Home > | SC-PwD.Home |
| ST_PwD | Home | OP.Home > | OP-PwD.Home > | ST.Home > | ST-PwD.Home |
| GEN | Other | OP.Other | | | |
| OBC | Other | OP.Other > | OBC.Other | | |
| SC | Other | OP.Other > | SC.Other | | |
| ST | Other | OP.Other > | ST.Other | | |
| GEN_PwD | Other | OP.Other > | OP-PwD.Other | | |
| OBC_PwD | Other | OP.Other > | OP-PwD.Other > | OBC.Other > | OBC-PwD.Other |
| SC_PwD | Other | OP.Other > | OP-PwD.Other > | SC.Other > | SC-PwD.Other |
| ST_PwD | Other | OP.Other > | OP-PwD.Other > | ST.Other > | ST-PwD.Other |

Figure 4.3: Virtual preference list for candidates for seats in the NITs. The first 8 rows represent preferences of candidates applying to the NIT in their "Home State" quota; seats in this quota are available only to state residents. Candidates applying to a NITs in $\nu$ which is not their home state quota (the last 8 rows) will compete with all residents of India, excluding candidates whose home state is $\nu$, the state under consideration.

of Planning & Architecture, Bhopal and Gurukula Kangri Vishwavidyalaya, Haridwar.

2. Home State (HS) and Other State (OS) quota: If a GFTI has HS and OS quota, then the virtual preference table is the same as that for NITs (shown in Figure 4.3). BITS Mesra is one such GFTI that follows the HS/OS quota model.

3. Home State (HS) and All India (AI) quota: The virtual preference table for such GFTIs is shown in Figure 4.4. Currently Assam university, Silchar is the only GFTI institute having HS and AI quota.

As an example, consider the virtual preference list for a ST-PwD person whose "Home State" is Assam. She may express a preference to apply to the mechanical engineering program of Assam University Silchar. In this case, her virtual preference list for this program will be represented by the ST-PwD row (the eighth row) in the table shown in Figure 4.4. Let us consider an OBC-NCL-PwD candidate whose Home State is not Assam. If one of his preference is a program in the Assam University Silchar, then the virtual preference list for this program will be read from the 14th row of the table shown in Figure 4.4. Note that not only will he be ineligible for the seats in the Home State quota of this university, he will also be competing for seats from All India quota with candidates whose home state is Assam.

### 4.5.4 Summary

The core algorithm is the standard DA algorithm with quotas. However, the virtual preference lists must be correctly constructed based on the programs a candidate applies to, and the tag he has. Once virtual program, virtual preference lists and virtual merit lists are constructed, the algorithm is applied for all candidates in the pool, including the PC candidates, DS candidates, international students, as well as the remaining bulk of candidates who do not belong to these special cases.

Note that the business rules involve complex de-reservation. As such, in order to complete the allocation, de-reservation as per the business rules must be employed on top of the process described so far. Chapter 6 is devoted to handling de-reservation in a holistic way.

| Category | Which State | Preference List | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| GEN | Home | OP.AI > | OP.HS | | | | | | | |
| OBC | Home | OP.AI > | OBC.AI > | OP.HS > | OBC.HS | | | | | |
| SC | Home | OP.AI > | SC.AI > | OP.HS > | SC.HS | | | | | |
| ST | Home | OP.AI > | ST.AI > | OP.HS > | ST.HS | | | | | |
| GEN_PwD | Home | OP.AI > | OP-PwD.AI > | OP.HS > | OP-PwD.HS | | | | | |
| OBC_PwD | Home | OP.AI > | OP-PwD.AI > | OBC.AI > | OBC-PwD.AI > | OP.HS > | OP-PwD.HS > | OBC.HS > | OBC-PwD.HS | |
| SC_PwD | Home | OP.AI > | OP-PwD.AI > | SC.AI > | SC-PwD.AI > | OP.HS > | OP-PwD.HS > | SC.HS > | SC-PwD.HS | |
| ST_PwD | Home | OP.AI > | OP-PwD.AI > | ST.AI > | ST-PwD.AI > | OP.HS > | OP-PwD.HS > | ST.HS > | ST-PwD.HS | |
| GEN | Other | OP.AI | | | | | | | | |
| OBC | Other | OP.AI > | OBC.AI | | | | | | | |
| SC | Other | OP.AI > | SC.AI | | | | | | | |
| ST | Other | OP.AI > | ST.AI | | | | | | | |
| GEN_PwD | Other | OP.AI > | OP-PwD.AI | | | | | | | |
| OBC_PwD | Other | OP.AI > | OP-PwD.AI > | OBC.AI > | OBC-PwD.AI | | | | | |
| SC_PwD | Other | OP.AI > | OP-PwD.AI > | SC.AI > | SC-PwD.AI | | | | | |
| ST_PwD | Other | OP.AI > | OP-PwD.AI > | ST.AI > | ST-PwD.AI | | | | | |

Figure 4.4: Virtual preference list for candidates for seats in certain other GFTIs. The last 8 rows represent preferences of candidates applying to the GFTI not in their "Home State" (HS); seats in this quota are available to all Indian nationals. Candidates applying to a GFTI in their home state (the first 8 rows) will first be considered for seats in the All India (AI) quota failing which they will be considered for seats in the Home State quota.

# Chapter 5

# Multi-round Implementation

Allocation of seats is conducted over multiple rounds for a variety of reasons, including the reality that offered seats may not be accepted. To maintain truthfulness, fairness, and optimality across rounds requires careful algorithmic design. Second and later rounds facilitate the utilization of surrendered seats, including the possibility of awarding a surrendered open category seat to a person earlier awarded a seat in a particular restricted quota.

The planned seat allocation involves three rounds of admissions, followed by a fourth round where only preparatory candidates are considered for new allotments in the IITs, whereas all candidates are considered in the non-IITs, (i.e., not in the IITs). The decision of whether to have 3 rounds, or fewer rounds is an empirical decision, and also a function of the admission calendar. There might also be the need for a final closure round (that occurs after classes start and some candidates do not report for class) where all candidates are considered for new allotments only in the non-IITs.

We describe the algorithm with the assumption that no new candidates can enter the system after the initial filling in of choices before any allotment is done. Thus, no candidate is allowed to fill or change preferences once given, and no fresh candidates are allowed after the fourth round (or after any round)[1].

After initial information collection (Section 3.1) our algorithm proceeds in rounds with the following activities taking place to provide inputs.

1. We execute the algorithm described in Section 4.5 using Algorithm 2 with the candidates initiating applications (proposing to the programs in the decreasing order of their preferences). Before executing it the first time, the optional inputs are not provided[2]. In subsequent rounds, optional inputs are expected to be provided. Specifically, the MIN-CUTOFF($p$) for each virtual program $p \in \mathcal{P}$ is chosen appropriately as described below in Section 5.5 based on the output of the prior round. The list CAT-CHANGE of

[1] In 2015, this consideration was not followed in the closure round for the CSAB. We discuss this in Chapter 7.

[2] As an implementation note, the CAT-CHANGE flag is arbitrarily set to 2 to indicate the default situation that the credentials of none of the candidates have changed.

candidates whose category has changed at any time after the first round is also provided as an input. The outcome of DA algorithm is an allocation of programs to candidates.

2. Candidates are informed about the programs assigned to them. The candidates have one of the following choices: reject, freeze, float, or slide. This is the case also before the closure round (i.e., after the fourth round). The details of these options are described in Section 5.4.

3. Based on the options chosen by the candidates, the candidate preference lists are suitably modified. These modified preference lists serve as the input to DA algorithm for the next round[3].

## 5.1 MRDA

The process described in Steps 1–3 is summarized in Figure 5.1 and it constitutes the multi-round Deferred Acceptance scheme. The process mentioned here does not take care of de-reservation which is addressed in Chapter 6. In brief, in de-reservation, multiple runs of the DA algorithm in Step 2 are performed, and we term the modified version of the DA algorithm as the multi-run DA algorithm, MRDA.

"Multi-Round" is a term that refers to the overall mechanism that occurs over days and possibly weeks, with the repeated involvement and inputs of humans, especially the candidates. In the context of a mechanism, the MR in MRDA may actually be considered to refer to Multi-Round Deferred Acceptance scheme. These two concepts – run and round – are orthogonal: one can perform multiple runs of the DA in situations where only one round is planned, and de-reservation is necessary; that would be a multi-run deferred acceptance algorithm. One could also perform a single run per round, and have multiple rounds, and run the algorithm as in Figure 5.1; this would be a multi-round scheme. Or one could do both, as was performed in July 2015. In terms of nomenclature, we choose not to distinguish between these cases as the intent is clear from the context.

## 5.2 Updating inputs after first round

Each round produces an allocation of programs to candidates that is fair and optimal. If there is another round remaining, the preference list are pre-processed. This is done in a special way for the fourth and closure rounds as described in Section 5.3, but always as described in Section 5.4. Changes to DA inputs other than preference lists are described in Section 5.5.

[3]A special note is made about the pre-processing the preference lists appropriately in Section 5.3 to implement the different rules for the fourth and closure rounds.

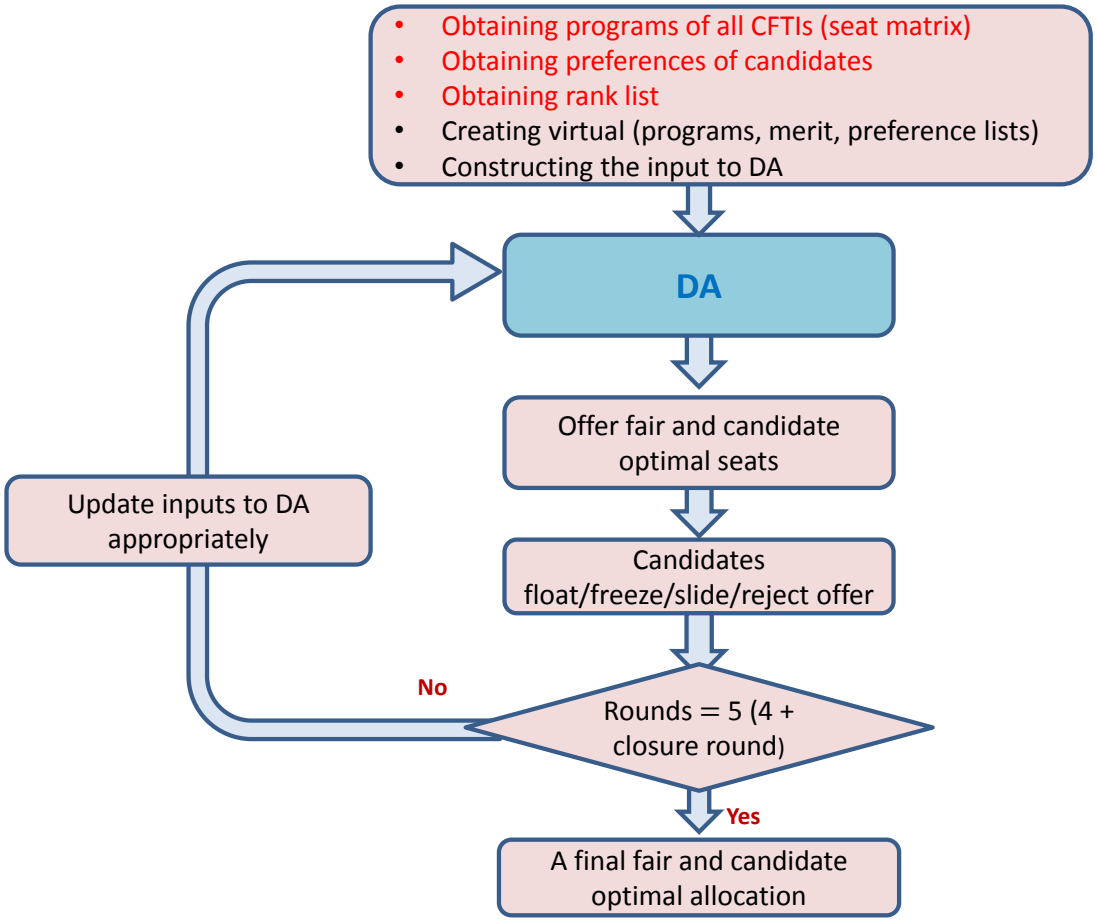



Figure 5.1: Multi-round deferred acceptance allocation.

## 5.3 Pre-processing applicable only after third round

**Before fourth round:** For all non-PC candidates (i.e., for those not on any of the IIT PC merit lists), we eliminate all IIT programs from the preference list, except the allotted IIT program (if an IIT program has been allotted.) Thus, in terms of output of the fourth round, there may be new allotments in IIT-PC programs as well as in non-IIT programs, but there are no new allotments in IIT programs.

REMARK: The step mentioned above is needed when an institute decides to take no more candidates in the regular program, whether seats are vacant or not. (IITs are simply an example.) Fairness considerations must be kept in mind while formulating this business rule. Consider

**Example 2.** *Ram wants to go to the mechanical engineering program in IIT Roorkee but the last available seat is taken by Shyam with a better rank in the third round. In the fourth round,*

*no new allocations are available to the IIT program. However Shyam is allowed to move out to say, NIT Jaipur which was his higher choice, and not available in earlier rounds. The seat that is vacated by Shyam is not given to Ram, but to a PC candidate with a rank lower than Ram.*

*This is not considered a merit violation as per business rules.*

**Before closure round:** This round is expected to be completed shortly after classes begin if there are a non-trivial number of vacant seats, due to candidates not showing up for classes at the assigned institute. We eliminate all IIT programs from the preference list of all candidates except the allotted IIT program (if an IIT program has been allotted). Thus, in terms of output of the closure round, there may be new allotments or sliding in non-IIT programs. There are no new allotments in IIT programs, regular or preparatory.

Cases like the one in Example 2 may occur in this situation also, and so the design of the closure round is important (see Chapter 7).

## 5.4 Preference list editing

Each candidate who is allocated a program in the previous round (call it $N$) chooses one of the following options before the next round ($N + 1$) is conducted. In each case we describe how the preference lists (after, if required, pre-processing as in Section 5.3) are appropriately edited before round $N+1$. These edited preference lists are then used to construct virtual preferences as usual.

- Reject: The candidate rejects the program. In this case, the preference list of the candidate is set to empty. (Note that this reference to Reject is different from the REJECT in the algorithm pseudocode.)
- Freeze: The candidate confirms the acceptance of the program allocated. In this case, we leave the allotted program[4] on her preference list and the entries below the allotted program and eliminate all other entries.
- Float: The candidate expresses that the program is acceptable but would like to be considered for future rounds to get a more desirable program, if possible. In this case, we leave the candidate preferences unchanged.
- Slide: The candidate wishes to be considered for future rounds but would prefer programs in the same institute where he is offered the current program. In this case, we remove all programs from his list that are above the current allotment and belong to the institute different from the institute of the program offered to him[5]. The allotted program and the entries below the allotted program are left unchanged.

[4] In other words, the virtual preference list for the candidate now contains only virtual programs corresponding to the allotted program and programs ranked below it by the candidate, as per the relevant table (using the complete row corresponding to the candidate category).

[5] Virtual preferences over virtual programs remain unchanged for programs that are not removed.

## 5.5 Other Inputs Needed After First Round

After the allocation of seats, a number of changes in the assumptions of the input materialize. The variety of these changes are large, and unpredictable; some of them are due to the inadequacies of the candidates, and some are due to the inadequacies of the institutional framework. We believe, that at least in the near term, some of these are hard to avoid. The MRDA algorithm makes provision for the following kinds of possibilities.

- MIN-CUTOFF **revisited:** The seat guarantee across multiple rounds is needed because, as discussed above, the rules provide for a freeze, float, and slide option. The seat guarantee is implemented by the notion of MIN-CUTOFF: A candidate is allowed to retain a seat in round $J+1$ if her rank is better off, or equal to the rank corresponding to the minimum cut-off rank in a virtual program at the end of $J$th round. This rank might be based on her rank, or, more subtly, the rank of any other candidate allotted a seat in the virtual program under consideration. Further, to avoid merit violation, the candidate may get something even better (from her perspective) in yet another program based on the minimum cut-off of that program, and this is permitted even on a supernumerary (SN) basis.

  This was described briefly as the MIN-CUTOFF benefit in Section 3.3.2.

- **Credential Changes:** After an allocation in a particular round, the seats of some candidates may get cancelled. The reason for the cancellation could be, for example,

  1. Birth category change: A candidate allotted an SC seat is unable to produce a valid SC certificate.
  2. PwD status change: A candidate allotted a PwD seat is unable to produce a valid PwD certificate.
  3. Medically capability change: A candidate while reporting to accept the offered seat is declared medically unfit for the offered program. Note that the candidate is unaware of this situation a priori[6]

  Note that the above list is not exhaustive. Depending upon the reason for seat cancellation, a candidate may or may not be awarded a penalty. For example, if the seat is cancelled due to situations similar to 1 or 2 above, one may reasonably infer that the fault lies with the candidate. While it is considered too harsh to remove the candidate from participating in future rounds, the candidate certainly should not get the same seat, and further, it may be desirable that he may not be given the MIN-CUTOFF benefit but be given the CAT-CHANGE penalty. To recall, the MIN-CUTOFF benefit is available only after the first round, and may be a supernumerary seat created when the program is full, and the candidate is better than the MIN-CUTOFF value.

[6]Example: A seat in the mining engineering program is offered to a student who turns out to be color blind. The mining program may not contain color blind students, as per law.

In yet another situation, a candidate's birth category needs to be changed (similar to the case above) despite the candidate not given any birth category reservation concession for that seat. In such cases, it is imperative that the credentials are changed, but the seat is not cancelled.

- **Merit Changes:** At present, marks obtained in the final secondary school examination are used in determining the rank of a candidate in the merit list of non-IITs programs. In some cases, candidates may be allowed a revision in ranks (subsequent to their publication) due to an error detected by education authorities. Merit lists will then need to be modified. Note that supernumerary seats is permitted for such candidates to simultaneously satisfy, e.g., the freeze assurance. As mentioned earlier, this corresponds to the MIN-CUTOFF benefit in Section 3.3.2.

Clearly, changes in the input after a round of seat allocation requires computing the MIN-CUTOFF and setting the CAT-CHANGE flag. The abstraction that the MRDA algorithm allows is the concept of the MIN-CUTOFF benefit, and the CAT-CHANGE penalty. In this chapter we have mentioned a subset of the possibilities. The precise implementation followed in July 2015 is described in Section 8.4.

## 5.6 Summary

We have described a multi-round deferred acceptance algorithm. After each round, candidates may reject, freeze, slide, or float the program offered. As a result, the program allocated to a candidate may change. It is guaranteed that the new program allocated to the candidate will be preferred (by the candidate) to the earlier one, unless the candidate experiences a downward revision of marks between those two rounds.

Many other business rules have been incorporated using the concept of virtual programs, virtual preference lists, and virtual merit lists. To make all desirable effects happen between rounds, internally in the data structures, the preference of the candidates need to be modified, their ranks may need to be changed, and possibly their credentials may need to be changed. Seats may become unavailable in the fourth round to the IITs. However, to maintain the performance guarantees, the pool of competing candidates does not change, and there is a predictability and trust in the system for these competing candidates.

**Remark:** Section 8.9 in Appendix presents the complete implementation details of the generic MRDA algorithm along with all input-output formats used for the year 2015. The reader interested in the actual implementation of the DA algorithm should refer to this section.

# Chapter 6

# De-reservations

Recall that the DA algorithm as described in Section 4.5 takes care of all business rule of categories as described in Chapter 4 except de-reservation. When seats are unfilled due to lack of candidates in a particular category, seats may be de-reserved.

After a possible allocation to preparatory candidates, if seats in (OPEN-PwD, OBC-NCL-PwD, SC-PwD, ST-PwD) seat category are still vacant in any program, they are de-reserved. For example, unfilled SC-PwD seats (that are also not filled with PC candidates) will be de-reserved and treated as SC category seats for allocation in every round of seat allotment. Similarly, unfilled OBC-NCL seats will be de-reserved and treated as OPEN seats (there is no allocation to PC candidates in this case). However, unfilled SC and ST category seats will NOT be de-reserved.

The de-reservation rules stated above are succinctly depicted in Figure 6.1.

There are two methods of incorporating de-reservation rules. The first and a very simple method is to change the input seat capacities available in a seat matrix, and move unfilled seats to the appropriate category, and run Algorithm 2 using the revised seat matrix. This method involves multiple runs of DA and was used for the joint seat allocation for the year 2015. This method is described in the following section. The second method involves modifying the virtual preference lists and the virtual merit lists. This method, along with its advantages and disadvantages over the first method, is described in Section 6.1.

## 6.1 Multi-Run DA with De-reservation

Multi-run DA may be viewed as running the method in Section 4.5 several times, each time updating the seat matrix provided as input by appropriately de-reserving seats that were not filled in the previous run.

### 6.1.1 Multi-Run DA Algorithm

The Multi-Run DA algorithm works as follows.

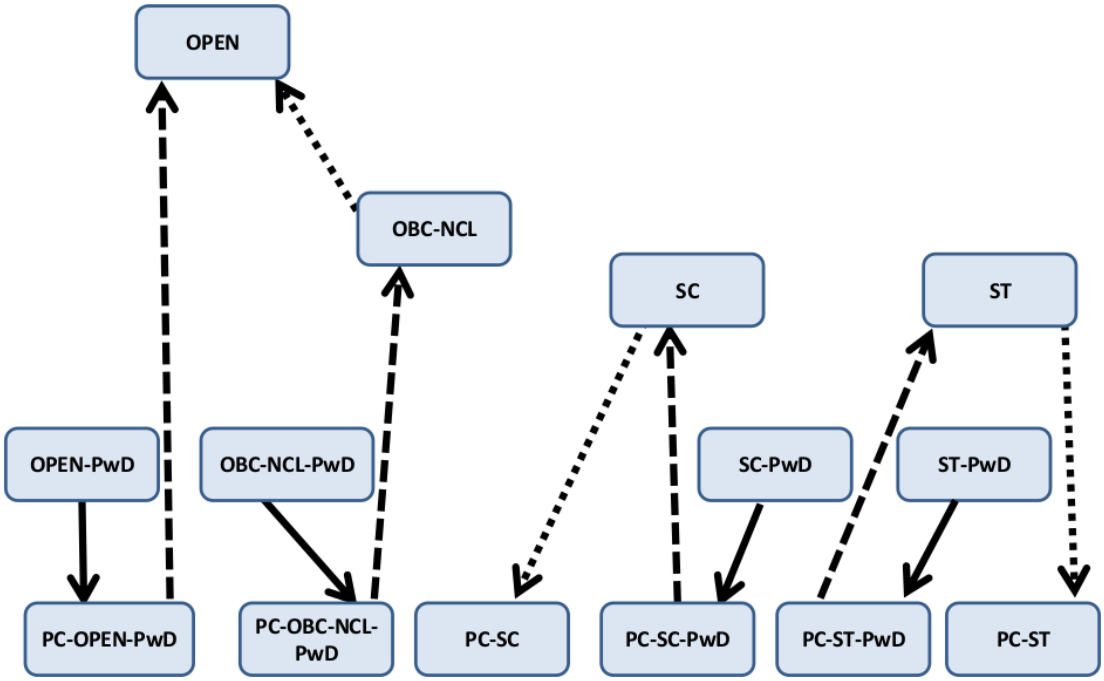



Figure 6.1: De-reservation. Starting from the bold arrow and following dotted arrows, one arrives at a possible sequence in which seats are de-reserved. For example, unfilled OPEN-PwD seats are considered to all eligible preparatory candidates (PC-OPEN-PwD); if at this stage, seats are still unfilled, they are offered to all eligible candidates (OPEN).

1. We run the algorithm in Section 4.5.[1]

2. If there are unfilled seats that can be de-reserved, then we update the capacities by de-reserving unfilled seats to the parent category. [2] We then re-run the algorithm as in Step 1 on all candidates.

   If there are no unfilled seats that can be de-reserved, i.e., all unfilled seats are in OPEN or SC or ST virtual programs, then we terminate.

This algorithm enjoys monotonicity across runs: The options available to candidates are only enhanced in going from one run to the next. More seats become available in parent programs,

[1]With the appropriate table for the virtual preference list depending on the program a candidate applies to, and the corresponding simplified virtual merit list.

[2]We de-reserve any unfilled OPEN-PwD seats to OPEN seats, unfilled SC-PwD seats to SC seats, unfilled ST-PwD to ST, and unfilled OBC-NCL-PwD to OBC-NCL, and unfilled OBC-NCL seats to OP, and update the seat matrix appropriately. As an example, if there are two unfilled seats in an OPEN-PwD virtual program, then we de-reserve them by decreasing the capacity of that virtual program by 2, and increasing the capacity of the corresponding OP virtual program by 2. It may be noted that the OPEN virtual program may simultaneously get some additional seats due to de-reservation from OBC-NCL during this step.

whereas the programs that "lose" seats due to de-reservation do not hurt the allocation since there is no demand for those seats anyway (in that run or in any future run). Thus, candidates get the same or better allocation than before. Seats only flow upstream towards parent categories, cf. Figure 6.1. On tests consisting of various synthetic as well as the actual data during the Joint Seat Allocation 2015, the number of runs were almost always around 5 only.

**Additional Output:** The primary purpose of the seat allocation algorithm is to allocate seats. However, in addition to the allocation of seats, we can output the opening and closing ranks in the Multi-Run Multi-Round algorithm after the last round. We read off the opening rank for each category if there is any candidate in the virtual program. The closing rank is infinity if one or more of the following occur:

- Any de-reservation occurred from the corresponding virtual program (at any stage).
- There are non-zero unfilled seats in the corresponding virtual program when the multi-run algorithm terminates. (This can occur in OPEN, SC and ST virtual programs. For other types of virtual programs, such seats are de-reserved before the algorithm terminates.)
- If some of the seats are occupied by PC candidates.[3]

If none of the above occur, the closing rank is the rank of the last admitted candidate in the corresponding virtual program.

In addition, we also output the initial seat matrix, and the final seat matrix for each program, with unfilled seats marked. The final seat matrix is the matrix used in the final run of DA during which no further de-reservation occurred. The following example shows how to interpret the initial and final seat matrix together.

**Example 3.** *Consider a program with:*

- *Initial seat matrix: 20 OPEN seats, 10 OBC-NCL seats, 2 OBC-NCL-PwD seats, and 1 OPEN-PwD seat. (Suppose there are no other seats.)*
- *Final seat matrix: 21 OPEN seats (18 filled), 11 OBC-NCL seats, 0 OBC-NCL-PwD seats and 1 OPEN-PwD seat (filled by PC candidate).*

The immediate inference is that all 11 OBC-NCL seats in the final seat matrix are occupied, since if not, they would have been de-reserved. We also notice that the 1 OPEN-PwD seat is occupied by a PC candidate.

We further interpret the above starting at the lowest level of Figure 6.1. We infer that 2 OBC-NCL-PwD seats were both de-reserved to OBC-NCL, leading to 12 OBC-NCL seats. Of these OBC-NCL seats, 1 was unfilled (hence the final seat matrix contained 11 OBC-NCL seats), and further de-reserved to OPEN, leading to 21 OPEN seats. Finally, we infer that 3 of these OPEN seats were unfilled.

[3]In this case, there is a closing rank for the corresponding PC course, which is the rank of the last PC candidate to be admitted if there was no de-reservation and null/infinity if there was de-reservation.

# Chapter 7

# Recommendations

Up until this chapter, the intent of this report was to present an algorithmic perspective on joint seat allocation in the context of centrally funded technical institutions. MRDA was conceived to be a generic method that would work whenever there are multiple merit lists, and multiple rounds, possibly involving reservations. In 2015, Algorithm MRDA was implemented independently by two different teams, one from the Indian Institute of Technology Kanpur, and the other from the National Informatics Centre. It is a testimony to the algorithm, and to the implementers' skills that the validation team found no errors in several runs of the implementation (See Section 8.8.) We therefore have very little to say about the algorithm design in this chapter. We make an exception to our overall goal in this report and make some broad recommendations (based on our interaction with various stake holders in the 2015 admission season.) We include technical notes as well as key practical considerations. Priority recommendations have been marked as **strong** recommendations.

## 7.1 Business Rules Considerations

1. **Board Marks** Board marks are to be used for qualifying criteria only (as in JEE Advanced). In other words, **it is strongly recommended** that Board marks should not be used for determining ranks.

2. **Credential Verification** As discussed in Section 5.5 credential changes causes major upheaval in various aspects of the algorithm. **It is recommended** that verification of the top 200,000 candidates (at least) is completed prior to choice filling.

3. **Asymmetry within joint allocation** Recall again Example 2. Since PC candidates are allowed to compete for seats at every stage, there appears to be no particular advantage in having a fourth round exclusively for the PC. This rule may have been conceived as a relic of the older system when PC candidates were allotted seats only in the last round. The DA algorithm elegantly avoid this issue, including the handling of de-reservation.

**It is recommended** that any asymmetry in the allotment be restricted to the closure round.

4. **Float to Freeze Option** Once a candidate floats, say in round 2, in 2015, he is (currently) not allowed to freeze in round 3.

   **Example 4.** *Consider Akbar who gets a seat in NIT Delhi in round 1, and chooses to float. In the next round he gets a seat in IIT Goa. Given that he was allowed to freeze NIT Delhi, it appears common sense to allow him to freeze IIT Goa even though he might have earlier mentioned NIT Hamirpur as a higher choice. Without the freeze option, he will float to NIT Hamirpur in the third round.*

   **It is recommended** that a float to freeze option be allowed.

5. **Withdraw**. Consider Sania.

   **Example 5.** *Sania got a seat in NIT Delhi. She accepted and and chose float because she was optimistic about an IIT seat. She also gets a seat a few days later in Delhi College of Engineering. After the third round, she realizes that she is not going to get the IIT seat. She wants to release her seat to someone else.*

   We can allow candidates who have previously accepted an allocation to surrender their seat at any time, and further provide incentives for early surrender, so that the seat can be reallocated. Such surrenders typically occur when a candidate obtains an outside offer of admission, and determines that the outside offer is superior. (Note that there is no way penalize such candidates, and one may not want to penalize them any way. Instead one can try to incentivize and encourage voluntary early surrender of seats.)

   Plans change. **It is strongly recommended** to offer a one-time withdraw option with full fee refund in the last but-one round. The refund should be released quickly with no questions asked, except for a brief survey. In contrast, an implicit withdraw when candidates do not attend classes should involve a longer process and engagement with the candidate.

6. **Opening Rank** Publishing Opening Rank (OR) appears to be an exercise "because it was done earlier". It appears to reveal very little when we publish that Kanpur CS opened at rank 48 in OBC category. It may give bragging rights to Kanpur at best. It also prompts the question: What exactly was the candidate's "general" rank?

   It is remarkable that the business rules of JoSAA published in 2015 had no mention of opening rank.

   **It is recommended** that OR is not published.

7. **Closing Rank** Publishing Closing Rank (CR) is intended to give a target. However, as the reader would have observed with the Min-Cutoff guarantee, the published closing rank is invalid for the next round

**Example 6.** *Amin has the last rank 2345 at the end of the second round. The second last rank is of Radha (1234); she chose freeze. JoSAA publishes the closing rank as 2345 but unknown to JoSAA, Amin prefers to go to BITS Pilani and will not accept his seat. In reality the closing rank is 1234.*

Publishing closing rank during the rounds is unscientific, confusing, and also sends a wrong target to the candidates. In general, we are going to have false positives and false negatives. (see Section 8.4 for details).

**It is recommended** that CR is published only before the closure round.

8. **Number of attempts** There is currently an asymmetry in the number of attempts a candidate has for admissions to the NITs and the IITs. There also appears to be some confusion whether a candidate who has accepted an NIT seat is eligible to write the JEE Advanced in subsequent years. **It is recommended** that the number of attempts, and the disqualification criteria be clearly spelt out.

9. **Timing** Along with choice filling, the calendar of events is critical to the process. **It is recommended** that there be a careful consideration of the calendar and the timeline with **adequate publicity** to the stake holders.

10. **De-reservation of HS to OS in NITs** There were many vacancies in HS quota of some programs in NITs in each round. This could have been avoided by de-reserving these vacant seats to OS quota. **It is recommended** that seats may be dereserved from HS to OS and vice-versa.

11. **Defense Service (DS) Priority Quota** The way DS is handled currently significantly increases the complexity of the code and the time required to run the software. There are a few other ways to handle DS:

    (a) Allocate seats to DS candidates manually, change the capacities of the OPEN virtual program and then run DA. This is the way DS has been handled prior to 2015. No doubt this simplifies the algorithm and reduces the running time. However, this introduces a new problem of unfairness.

    **Example 7.** *Consider Ram with an IIT JEE Advanced Rank of 45 whose preference is IIT Bombay CS followed by IIT Kanpur, Mechanical. Unfortunately, Sania and Michael, both DS, are ahead of him, and they also choose IIT Bombay CS. An algorithm that assigns seats to "DS" candidates (before considering any other candidate) would, keeping in mind the limit of 2 per institute, assign a mechanical seat to Ram. However, Ram could very well have got IIT Bombay CS based solely on his rank. The example could be generalized to other ranks easily.*

    It is well understood that such a case of DS is extremely improbable. However, while designing a software, one must take care of all possible scenarios and the correctness of the algorithm should not depend upon statistics. Hence this approach is extremely undesirable.

(b) DS candidates are handled in the same way as the CSAB has been handling them till now: a DS candidate is allotted a seat as per her rank, but once she joins the institute, she may be moved to any program of her choice within the institute (the "preferential" quota). This will again simplify the algorithm and reduce the running time. However, it must be noted that if such an approach were to be carried out, the number of DS allotments in IITs may reduce to zero. In 2015, if this rule had been used, only one of the DS candidates would have got a seat.

(c) DS candidates should be assigned supernumerary seats (2 per institute). Although the number appears to be large, from empirical evidence it appears that the number of seats created due to such a rule in the IITs is always in single digits. In 2015 the number would have been 3. This will simplify the algorithm considerably. Here again, the supernumerary seats are dependent on statistics, but note that the correctness of the algorithm is still independent of the statistics. This approach is **strongly recommended**.

If none of the above 3 approaches is implemented, only the 2015 approach along with race condition consideration must be used.

## 7.2 Closure Round

Despite all efforts, it is envisaged that thousand students or more will not report for classes. **It is strongly recommended** that these vacancies be filled by a special closure round, which occurs roughly during the second week of classes. Proper structure of the closure round is important to retain the good properties of the overall scheme, (see properties listed at the end of Section 2.2). For this reason, and because vacancies are sure to occur, it is desirable to make the special round a formal part of the combined admission process that is planned beforehand.

**Design:** A systematic closure round should have a format that fits the following description to ensure good properties of the overall mechanism: The round will be centralized. Each institute or program can choose to participate or not. If an institute participates, students may join or leave during the closure round. If an institute does not participate, students may leave but may not newly join the institute's programs. Candidates should be permitted to choose freeze/float/slide once more. Candidates may newly register, or choose to change their preferences, but in these two cases they do not get the Min-Cutoff benefit, nor are supernumerary seats created to accommodate them.[1]

## 7.3 Choice filling

During the choice filling period of the Joint Seat Allocation 2015, on many occasions, the servers were not able to handle the load. There were many phone calls at various IITs and NITs with

[1] this is accomplished by including these candidates in the credential change list implemented as Cat-Change, given as input to Algorithm 2

students complaining about the slow response of servers because of which they were not able to enter or update their choices. This led to extending the deadline for choice filling by one extra day. However, even on the last day, the servers were not able to handle the load. There were complaints that the login sessions were getting expired instantly and the response from servers was again very slow. **It is recommended** that this problem be addressed.

While load balancing is a relatively easy problem to solve, education of candidates in the filling of choice is an outstanding and difficult problem and may jeopardize the entire scheme. **It is strongly recommended** that the process of choice filling be greatly simplified. (See Section 8.5 for some ideas.)

## 7.4 New Institutes

**Example 8.** *According to a rule which was declared around the middle of June in 2015, a new institute called NIT AP came into existence. 60 supernumerary seats in NIT Warangal were allocated exclusively for candidates from AP state, until NIT AP would be able to function.*

This rule was implemented by creating two different institutes visible to candidates. This led to a lot of confusion for many candidates. In the future, **it is recommended** that such rules should be implemented in a manner that the candidates see only one institute (in this example, "NIT Warangal") and the software incorporates this rule by augmenting the virtual preference list suitably in case the candidate is from Andhra Pradesh state. One way to handle it is as follows: We can introduce a new 'AP' quota for all NITs. Only NIT Warangal will have seats under this quota. Like the HS and OS quota, we define the order in which candidates apply to NIT Warangal: First OS then AP. Only Andhra Pradesh candidates should be eligible for the AP quota of NIT Warangal. This will take care of everything with the least changes in software required and it is also a clean solution algorithmically as well. However, if there are many such new institutes in future, the generalization of this solution does not remain elegant any more. In such scenario, the technical committee has a couple of alternative solutions which may be used as per the requirement.

## 7.5 Data collection and analysis

We cannot overemphasize this. **It is strongly recommended** that the candidates who surrender their seat even after reporting at the reporting centers be required to provide the reason for the same in order to obtain refund of fees (but with no reasons being labelled as bad or penalized in any way). This will help in determining the cause of the vacant seats, and the timing of when candidates found out about better options outside. This data may be used in future years to reduce vacancies. The timing of seat allocation rounds can be tuned accordingly to make best use of surrendered seats. The institutes that take away most of such candidates may even be invited to participate in the joint seat allocation in future.

A sample survey form is shown in Section 8.6.

The survey responses must be collected such that the responses can be associated with the record of the candidate who filled them in. For instance, the survey can be posted on the NIC portal and when the survey response is submitted, the candidate is given a code which he must take to the bank in order to obtain their refund.

## 7.6 Algorithmic Considerations

1. The seat guarantee across rounds was implemented using the concept of Min-Cutoff. However, there are surely other alternatives. The authors explored keeping a `no-reject-list` instead of a flat Min-Cutoff but due to efficiency considerations, and the uncertainty in the rules at that time recommended Min-Cutoff. `no-reject-list` is not described in this report.

2. An alternative to the multi-run deferred acceptance algorithm is a single pass deferred acceptance algorithm described in Section 6.1. The number of runs in multi-run is not deterministic, and the run time in the NIC implementation could be in the order of hours. However, the advantage of multi-run is a shorter description of the virtual merit lists and an easier to understand algorithm. We offer no new recommendation in this section.

3. In case a candidate opts for the freeze option, all of her choices above the one she was allotted are removed from the choice list in the 2015 implementation. To understand why this may lead to a problem, consider the following case:

   **Example 9.** *Alice filled 25 choices and got her 24th choice of CSE in NIT Rourkela through GEN category at the end of first round. Thinking that she would not be able to get any better choice, she opted for Freeze at the reporting center. Now, her first 23 choices will be removed from the list and she will be left with just her 24th and 25th choices as far as the allotment software is concerned. if Alice's ENG marks reduce, and this information is received after her reporting, she might not get any program at all. Suppose that some candidate who was blocking her seat at her 2nd preference of Mechanical at IIT Kanpur opted for Reject then another candidate with ADV rank worse than Alice might get that seat.*

   *Since her marks reduced, she will not be given a guarantee of the CSE seat in NIT Rourkela and she may be displaced by a candidate with better rank than hers. She may further not get her 25th preference either. As such she may be left with no seat, even though she could have got her 2nd preference (Mech at IIT Kanpur).*

   It is recommended that, for freeze, the candidate's allotted choice is moved to the top of her preference list retaining the rest of her choices. That way, even if her marks reduce, she will be given a chance to apply to the programs further down in her preference list.

   Similarly, if a candidate opts for slide, all of her choices in her allotted institute should be taken to the top of her preference list retaining the rest of the preference list and the relative ordering.

# Chapter 8

# Appendix

## 8.1 Proof of Correctness

We consider the basic DA algorithm, Section 3 and start with some obvious observations.

**Observation 10.** *A candidate applies to programs in the decreasing order of preferences. Further, if a program p rejects a candidate x during some iteration, then x will never apply to p again.*

Observation 10 also implies termination of the DA algorithm in time which is polynomial in the number of candidates and programs.

**Observation 11.** *Let x and y be two candidates in waiting list of program p at any moment of time. Let x have rank better than y in program p. If, ever in future, x is rejected by program p, y must also be rejected by p.*

We shall show that DA algorithm is fair, and candidate optimal.

**Theorem 12.** *DA algorithm terminates with a fair assignment.*

*Proof.* First, note that a candidate never applies to a program that is not part of his list, and hence cannot possibly be allotted such a program. Also, programs immediately reject any candidates that are not eligible, hence no program is allotted a candidate who is not eligible.

Hence, In order to show fairness of DA algorithm, it suffices to establish the following assertion for each candidate $x$ and program $p$:

Suppose $x$ applies to $p$, but fails to get admitted into the program; then every other candidate $y$ with rank worse than $x$ in the merit list of $p$ will also fail to get admitted to program $p$.

The proof deals with two cases depending upon the order in which $x$ and $y$ apply to program $p$ during DA algorithm.

Suppose $y$ applies to $p$ before $x$. Consider the iteration in which $x$ applies to $p$. (We have already assumed that $x$ has applied to $p$.) At this iteration, $y$ may have already been rejected in which case our assertion is true from Observation 1. If $y$ was not rejected, then, since $x$ has

a better rank than $y$, and $y$ is still in the waiting list of $p$, $x$ will also join the waiting list. It follows from Observation 11 that if ever in the future, $x$ is rejected, then $y$ will also be rejected.

Suppose $x$ applies to $p$ before $y$ during the algorithm. Consider the moment when $y$ applies. If $x$ is already there, then it follows from Observation 11 that if, ever in future, $x$ is rejected, then $y$ will also be rejected. If $x$ is not in the waiting list, it would imply that the program has full capacity at this moment and every candidate in this waiting list has rank better than that of $x$. In this case, when $y$ applies to $p$, $y$ will be rejected.

□

**Theorem 13.** *DA algorithm is candidate optimal.*

*Proof.* Suppose DA ware NOT optimal. Then there must be candidate $x$ who prefers a program $p$ to the program DA finally allots her, namely, $\mu(x)$. Naturally $x$ must have been rejected by $p$ in some DA iteration. We show that this cannot happen. It suffices if we can establish assertion $\mathcal{B}(i)$, defined below, holds for iteration $i$ of the DA algorithm:

| $\mathcal{B}(i)$: If during $i$th iteration of DA, any program $p$ rejects any candidate $x$, then there can not exist any fair assignment that assigns $p$ to $x$. |
|---|

Let program $p$ have capacity $c$. We shall prove $\mathcal{B}(i)$ by induction on $i$.

**Base case:** $(i = 1)$ Suppose $p$ rejects $x$ in the first iteration. It must be because of a set $S$ of $c$ candidates with rank better than $x$ applying to $p$ during the first iteration of DA. Let there exist another assignment $\mathcal{M}'$ that assigns $p$ to $x$. So there must be at least one candidate $y \in S$ such that $p$ is not assigned to $y$. Suppose $\mathcal{M}'$ assigns $p'$ to $y$. It follows from Observation 10 that $y$ prefers $p$ the most. In particular, $y$ prefers $p$ to $p'$. Hence $\mathcal{M}'$ has been unfair to $y$ in NOT assigning program $p$.

**Induction step:** Consider iteration $i$ of the DA algorithm. Let some program $p$ rejects $x$ in $i$th iteration. It must have been because of some set $S$ of $c$ candidates in the waiting list of $p$ at the end of $i$th iteration, each having rank better than $x$. Since $p$ has capacity $c$ only and $\mathcal{M}'$ assigns $p$ to $x$, so there must be at least one candidate $y \in S$ such that $\mathcal{M}'$ does not assign $p$ to $y$. Suppose $\mathcal{M}'$ assigns $p'$ $(\neq p)$to $y$. We shall now show that $y$ prefers $p$ to $p'$ (this will lead to unfairness of $\mathcal{M}'$). We proceed as follows. By the end of iteration $i$ of DA, the set of programs to which $y$ applied is $R_y \cup \{p\}$ where $R_y$ (possibly empty) is the set of programs that rejected $y$. It can be observed that the set of programs that rejected $y$ by the end of $(i-1)$th iteration of DA algorithm is $R_y$. So using induction hypothesis, it follows that if $\mathcal{M}'$ is a fair assignment then it would not assign any of the programs from $R_y$ to $y$. And since $\mathcal{M}'$ assigns $p'$ to $y$, this would imply that $p' \notin R_y$. Now recall that each candidate applies to programs in decreasing order of his/her preferences during DA (Observation 10). Therefore, each program in $R_y \cup \{p\}$ is preferred to $p'$ by $y$. In particular, $y$ prefers $p$ to $p'$. Therefore, $\mathcal{M}'$ is unfair: It assign $p$ to $x$ and $p'$ to $y$ in spite of the fact that $y$ prefers $p$ to $p'$ and has a better rank than $x$. □

Finally, the following truthfulness result was proved by Dubins and Freedman [3].

**Theorem 14.** *Each candidate can do the best possible for herself by reporting her true preferences, whatever be the program merit lists, program capacities, and preferences of the other candidates. In fact, there is no group of candidates that can each obtain better allocations by misreporting their preferences.*

## 8.2 Computational considerations of the DA algorithms

The number of applications made during this algorithm is bounded above by $O(AP)$ where $A$ denotes the number of candidates and $P$ denotes the number of programs. This is because each candidate can apply at most once to each program (see Observation 10 further). With an efficient implementation, each application will require only $O(\log A)$ computational operations, and the memory used will also be bounded by $O(AP)$ at all times. Overall, the algorithm should run in seconds (or even milliseconds) on a personal computer, even with $A = 30,000$ and $P = 1,000$.

Algorithm 2 consists of pseudocode towards an efficient implementation of the Deferred Acceptance algorithm. We create a variable $i(x)$ associated with the current position in the preference list for candidate $x$. For each program $p$, we maintain a list WaitList($p$) of candidates who are currently on the wait list, sorted according to the merit list of the program. The rank of the "bottom" candidate who is currently last on the list (if the list is full to capacity $c(p)$) serves as the current "cutoff". We maintain a queue of candidates $\mathcal{Q}$ who are not currently on any wait list and want to apply further (they have not reached the end of their preference list). These candidates make applications (any order works, we make use of a queue which corresponds to first-in-first-out). If a candidate $x$ with rank better (smaller) than the cutoff applies to a program $p$, then the bottom candidate $y$ in WaitList($p$) gets rejected and $x$ is inserted at the right location in the sorted wait list of $p$, else $x$ is rejected. Let $z$ be the rejected candidate. The current position on the preference list of $z$ is incremented. If the end of $z$'s list has not been reached then we add $z$ to $\mathcal{Q}$. We continue processing candidates in $\mathcal{Q}$ in this way until $\mathcal{Q}$ becomes empty. The final allocation of a candidate $x$ can be read off from the current position on the preference list of $x$ (if the end of the list has been reached then $x$ has no allocation). The candidates allotted to a program $p$ can be read off from the wait list of $p$ at termination.

## 8.3 Single Run DA with De-reservation

In contrast to the method presented in Section 6.1, we present a single-run version that addresses de-reservation.

Specifically, we run the method from Section 4.5 but modify the preference list. For example, Figure 8.1 shows the modification of the virtual preference table for the IITs. (The virtual preference tables for the other institutions are similar, and described later.)

Ignore for the moment any entry with the `(d)` tag. Then, for example, the sixth row states that for an OBC candidate with PwD tag, we try to fill the OPEN seats before any other seat,

failing which we consider the OPEN-PwD seat before venturing into OBC-NCL seats.

However, as mentioned earlier, de-reservations are possible, and are thus mentioned in Figure 8.1 using the symbol "d". The number of d's in parenthesis for each (candidate type, virtual program) pair indicates levels of de-reservation needed before a candidate in that category is considered for that particular virtual program. Once these "d" symbols are removed, the seats really manifest themselves in the home category. So, e.g., for GEN candidates, the order of virtual programs is really

```
OPEN > OPEN > OPEN > OPEN
```

(obtained by dropping the d's) which is as good as saying that GEN candidates can get only open seats.

Together with the virtual merit list, the order of trying for a seat for a candidate, and his entitlement leads to an allocation of seat category. For example,

1. The (d) specification in, say OBC-NCL (d), manifests not in the virtual preference list of the candidate, but in the virtual merit list of the OBC-NCL virtual program, where the top group consists of only OBC-NCL candidates, and the GEN student appears only in the second group which includes all remaining students in the CRL.

   The number of d's in parenthesis is indicative of the priority of that candidate type in the virtual merit list of the corresponding virtual program, It may be useful to visualize a (d)

| Category | PD Status | Preference List | | | | | |
|---|---|---|---|---|---|---|---|
| GEN | N | OBC.d > | OP-PD.dd > | OBC-PD.ddd > | OP | | |
| OBC | N | OP-PD.dd > | OP > | OBC-PD.dd > | OBC | | |
| SC | N | OBC.d > | OP-PD.dd > | OBC-PD.ddd > | OP > | SC-PD.dd > | SC |
| ST | N | OBC.d > | OP-PD.dd > | OBC-PD.ddd > | OP > | ST-PD.dd > | ST |
| GEN | Y | OBC.d > | OBC-PD.ddd > | OP > | OP-PD | | |
| OBC | Y | OP > | OP-PD > | OBC > | OBC-PD | | |
| SC | Y | OBC.d > | OBC-PD.ddd > | OP > | OP-PD > | SC > | SC-PD |
| ST | Y | OBC.d > | OBC-PD.ddd > | OP > | OP-PD > | ST > | ST-PD |
| PC(SC) | N | SC-PD.ddd > | SC.d | | | | |
| PC(ST) | N | ST-PD.ddd > | ST.d | | | | |
| PC(SC) | Y | OP-PD.d > | SC.d > | SC-PD.d | | | |
| PC(ST) | Y | OP-PD.d > | ST.d > | ST-PD.d | | | |
| PC(OBC) | Y | OP-PD.d > | OBC-PD.d | | | | |
| PC(GE) | Y | OP-PD.d | | | | | |

Figure 8.1: Virtual preference table for candidates applying to programs in the IITs. The last six rows apply to candidates in the preparatory lists. In reality, a candidate applies for only the seats he is eligible for; e.g., a GEN category candidate applies only for open seats (some of these might have been made available due to de-reservations). Notice, for example, a GEN category candidate is ineligible for any type of SC seats because these are not de-reserved.

as a push or a pop. For example, OBC-NCL-PwD (ddd) may be visualized as a push into PC, then a pop into OBC-NCL, and a further pop into OPEN, following the direction of arrows in Figure 6.1.

2. OBC-NCL(d), OBC-NCL-PwD(ddd) and OPEN-PwD(dd) are all "equivalent" to OPEN
3. SC-PwD(dd) is equivalent to SC. ST-PwD(dd) is equivalent to ST.
4. OBC-NCL-PwD(dd) is equivalent to OBC-NCL.

In detail, the virtual merit lists of virtual programs are constructed as follows:[1]

- For OPEN virtual programs we consider in order
  - All candidates in CRL order.
- For OBC-NCL virtual programs: we consider in order
  - The OBC-NCL merit list
  - (d): the CRL (all remaining candidates)
- For SC virtual programs: we consider in order
  - the SC merit list
  - (d): the PC-SC merit list
- For ST virtual programs: we consider in order
  - the ST merit list,
  - (d): the PC-ST merit list.
- For OPEN-PwD virtual programs: we consider in order
  - the OPEN-PwD merit list, then
  - (d): all candidates in the PC-CRL-PwD merit list, then
  - (dd): all candidates in the CRL (remaining candidates not listed in the two steps above).
- For OBC-NCL-PwD virtual programs: we consider in order
  - the OBC-NCL-PwD merit list, then
  - (d): the PC-OBC-NCL-PwD merit list
  - (dd): the OBC-NCL merit list (remaining candidates)

[1]The virtual merit list is identical for each IIT virtual program of a particular type; e.g., IITB CSE BTech OPEN-PwD has the same virtual merit list as IITK Chemical BTech OPEN-PwD.

  - (ddd): the CRL merit list (remaining candidates)

- For SC-PwD virtual programs: we consider in order
  - the SC-PwD merit list, then
  - (d): the PC-SC-PwD merit list, then
  - (dd): the SC merit list (remaining candidates), then
  - (ddd): the PC-SC merit list (remaining candidates)

- For ST-PwD virtual programs: we consider in order
  - the ST-PwD merit list, then
  - (d): the PC-ST-PwD merit list, then
  - (dd): the ST merit list (remaining candidates), then
  - (ddd): the PC-ST merit list (remaining candidates)

### Virtual preference table for de-reservation for NITs

The virtual preference table in this case appears in Figure 8.2. The same ideas of merit list as described above must be applied in this case.

### Virtual preference table for de-reservation for other GFTIs

The virtual preference table in this case appears in Figure 8.3.

### Summary, discussion and comparison

We have given two options for implementing de-reservations in Section 8.3 and 6.1 above.

- The advantage of the single run DA is that it is faster, since it involves a single run. We see no inherent disadvantage in the allocation except that careful definitions are required for opening rank, closing rank, category of each assigned seat, etc. See example below.

- The advantage of the multirun DA is that the description of the virtual preferences and virtual merit lists is very simple (the latter resembling Fig. 4.1); further the opening and closing ranks are easy to output. In addition, on a single merit list input, it matches the output of a single merit list algorithm with de-reservations.[2]

  The disadvantage is that it requires multiple runs and so is expected to take longer to run. In almost all of the synthetic and real data sets, the multirun DA took around 5 runs, thus making its running time about 5 times that of the single run DA.

[2]It is not immediately obvious what such an algorithm is. We consider the following natural definition of the benchmark single merit list algorithm: Follow the common rank list (CRL) order and make allotments, de-reserve unfilled seats, and repeat from the beginning. Iterate until there are no new de-reserved seats.

| Category | Which State | Preference List | | | | | |
|---|---|---|---|---|---|---|---|
| GEN | HOME | OBC.d.HOME > | OP-PD.d.HOME > | OBC-PD.dd.HOME > | OP.HOME | | |
| OBC | HOME | OP-PD.d.HOME > | OP.HOME > | OBC-PD.d.HOME > | OBC.HOME | | |
| SC | HOME | OBC.d.HOME > | OP-PD.d.HOME > | OBC-PD.dd.HOME > | OP.HOME > | SC-PD.d.HOME > | SC.HOME |
| ST | HOME | OBC.d.HOME > | OP-PD.d.HOME > | OBC-PD.dd.HOME > | OP.HOME > | ST-PD.d.HOME > | ST.HOME |
| GEN_PwD | HOME | OBC.d.HOME > | OBC-PD.dd.HOME > | OP.HOME > | OP-PD.HOME | | |
| OBC_PwD | HOME | OP.HOME > | OP-PD.HOME > | OBC.HOME > | OBC-PD.HOME | | |
| SC_PwD | HOME | OBC.d.HOME > | OBC-PD.dd.HOME > | OP.HOME > | OP-PD.HOME > | SC.HOME > | SC-PD.HOME |
| ST_PwD | HOME | OBC.d.HOME > | OBC-PD.dd.HOME > | OP.HOME > | OP-PD.HOME > | ST.HOME > | ST-PD.HOME |
| GEN | OTHER | OBC.d.OTHER > | OP-PD.d.OTHER > | OBC-PD.dd.OTHER > | OP.OTHER | | |
| OBC | OTHER | OP-PD.d.OTHER > | OP.OTHER > | OBC-PD.d.OTHER > | OBC.OTHER | | |
| SC | OTHER | OBC.d.OTHER > | OP-PD.d.OTHER > | OBC-PD.dd.OTHER > | OP.OTHER > | SC-PD.d.OTHER > | SC.OTHER |
| ST | OTHER | OBC.d.OTHER > | OP-PD.d.OTHER > | OBC-PD.dd.OTHER > | OP.OTHER > | ST-PD.d.OTHER > | ST.OTHER |
| GEN_PwD | OTHER | OBC.d.OTHER > | OBC-PD.dd.OTHER > | OP.OTHER > | OP-PD.OTHER | | |
| OBC_PwD | OTHER | OP.OTHER > | OP-PD.OTHER > | OBC.OTHER > | OBC-PD.OTHER | | |
| SC_PwD | OTHER | OBC.d.OTHER > | OBC-PD.dd.OTHER > | OP.OTHER > | OP-PD.OTHER > | SC.OTHER > | SC-PD.OTHER |
| ST_PwD | OTHER | OBC.d.OTHER > | OBC-PD.dd.OTHER > | OP.OTHER > | OP-PD.OTHER > | ST.OTHER > | ST-PD.OTHER |

Figure 8.2: Virtual preference list for candidates for seats in the NITs. with de-reservation. See Section 4.5.2 together with Section 8.3 on how to read the entries.

| Category | Which State | Preference List | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GEN | HOME | OBC.d.AI > | OP-PD.d.AI > | OBC-PD.dd.AI > | OP.AI > | OBC.d.HS > | OP-PD.d.HS > | OBC-PD.dd.HS > | OP.HS | | | | |
| OBC | HOME | OP-PD.d.AI > | OP.AI > | OBC-PD.d.AI > | OBC.AI > | OP-PD.d.HS > | OP.HS > | OBC-PD.d.HS > | OBC.HS | | | | |
| SC | HOME | OBC.d.AI > | OP-PD.d.AI > | OBC-PD.dd.AI > | OP.AI > | SC-PD.d.AI > | SC.AI > | OBC.d.HS > | OP-PD.d.HS > | OBC-PD.dd.HS > | OP.HS > | SC-PD.d.HS > | SC.HS |
| ST | HOME | OBC.d.AI > | OP-PD.d.AI > | OBC-PD.dd.AI > | OP.AI > | ST-PD.d.AI > | ST.AI > | OBC.d.HS > | OP-PD.d.HS > | OBC-PD.dd.HS > | OP.HS > | ST-PD.d.HS > | ST.HS |
| GEN_PwD | HOME | OBC.d.AI > | OBC-PD.dd.AI > | OP.AI > | OP-PD.AI > | OBC.d.HS > | OBC-PD.dd.HS > | OP.HS > | OP-PD.HS | | | | |
| OBC_PwD | HOME | OP.AI > | OP-PD.AI > | OBC.AI > | OBC-PD.AI > | OP.HS > | OP-PD.HS > | OBC.HS > | OBC-PD.HS | | | | |
| SC_PwD | HOME | OBC.d.AI > | OBC-PD.dd.AI > | OP.AI > | OP-PD.AI > | SC.AI > | SC-PD.AI > | OBC.d.HS > | OBC-PD.dd.HS > | OP.HS > | OP-PD.HS > | SC.HS > | SC-PD.HS |
| ST_PwD | HOME | OBC.d.AI > | OBC-PD.dd.AI > | OP.AI > | OP-PD.AI > | ST.AI > | ST-PD.AI > | OBC.d.HS > | OBC-PD.dd.HS > | OP.HS > | OP-PD.HS > | ST.HS > | ST-PD.HS |
| GEN | OTHER | OBC.d.AI > | OP-PD.d.AI > | OBC-PD.dd.AI > | OP.AI | | | | | | | | |
| OBC | OTHER | OP-PD.d.AI > | OP.AI > | OBC-PD.d.AI > | OBC.AI | | | | | | | | |
| SC | OTHER | OBC.d.AI > | OP-PD.d.AI > | OBC-PD.dd.AI > | OP.AI > | SC-PD.d.AI > | SC.AI | | | | | | |
| ST | OTHER | OBC.d.AI > | OP-PD.d.AI > | OBC-PD.dd.AI > | OP.AI > | ST-PD.d.AI > | ST.AI | | | | | | |
| GEN_PwD | OTHER | OBC.d.AI > | OBC-PD.dd.AI > | OP.AI > | OP-PD.AI | | | | | | | | |
| OBC_PwD | OTHER | OP.AI > | OP-PD.AI > | OBC.AI > | OBC-PD.AI | | | | | | | | |
| SC_PwD | OTHER | OBC.d.AI > | OBC-PD.dd.AI > | OP.AI > | OP-PD.AI > | SC.AI > | SC-PD.AI | | | | | | |
| ST_PwD | OTHER | OBC.d.AI > | OBC-PD.dd.AI > | OP.AI > | OP-PD.AI > | ST.AI > | ST-PD.AI | | | | | | |

Figure 8.3: Virtual preference list for candidates for seats in certain other GFTIs with de-reservation. See Section 4.5.3 together with Section 8.3 on how to read the entries.

- The output of the two algorithms is closely related. Ignore ties for this discussion (ties can lead to very small deviations from what we describe below). Then the two algorithms give the same program level allocation, with possible differences in the split of different candidates across different virtual programs. One obvious difference is that a candidate may occupy a de-reserved seat marked with (d) in the single run version (technically a seat in the original category virtual program), whereas in the multirun version, the seat would first be de-reserved and then the same candidate would get a seat in the parent virtual program. In addition, there can be a difference in examples like the following: Two OPEN seats, one OPEN-PwD seat, two OPEN candidates, CRL rank 1 and 2, and one OPEN-PwD candidate CRL 259 and OPEN-PwD 1. The single pass version finds the true candidate optimal fair allocation in the constructed "market" with virtual preferences that include de-reserved versions of programs: i.e., CRL 1 ends up with an OPEN-PwD (d) seat, whereas CRL 2 and 259 end up with OPEN seats. The multi-run version finds a slightly different fair allocation, with CRL 1 and 2 getting OPEN seats and CRL 259 getting an OPEN-PwD seat. **The program level allocation, i.e., what program a candidate is allotted a seat to, remains the same.**

Since ties can theoretically cause small differences in the allocation, and to avoid complications with computing opening and closing ranks, etc., it may be prudent to employ the multi-run implementation of de-reservations.

## 8.4 Computing Min-Cutoff and setting Cat-Change

**Setting** Min-Cutoff: Candidates who are to be penalized (the so-called Cat-Change penalty) have their Cat-Change flag set to 1. They are also added to the Cat-Change list. Candidates whose seats were cancelled, but are not to be penalized still need to have their credentials updated. In the mining example of Section 5.5, the candidate should not get the same mining seat. This situation is recorded by setting their Cat-Change flag to 4. (By default, recall that the Cat-Change flag is 2.) Further, it is set to 3 if a candidate's credentials change, but the seat doesn't get cancelled.[3]. Figure 8.4 depicts how the Cat-Change flag is set in various situations.

**Computing** Min-Cutoff**:** While the notion of Min-Cutoff is very candidate friendly, there are complications due to the intertwined change in credentials, and the change in the merit order. The word minimum in min-cut-off is used to indicate that in future rounds, the cutoff (or closing rank) can actually become worse (i.e., larger) than the min-cut-off. We consider the output WaitList($p$), and set Min-Cutoff($p$) for the next round as follows. First note that it is entirely possible that programs may have unfilled seats. These unfilled seats, may or may not be de-reserved. In the sequel, we assume that merit lists have been updated

[3]A candidate with Cat-Change set to 1 candidate cannot get the freeze assurance even in future rounds with the way "freeze" is implemented (see Section 5.4)

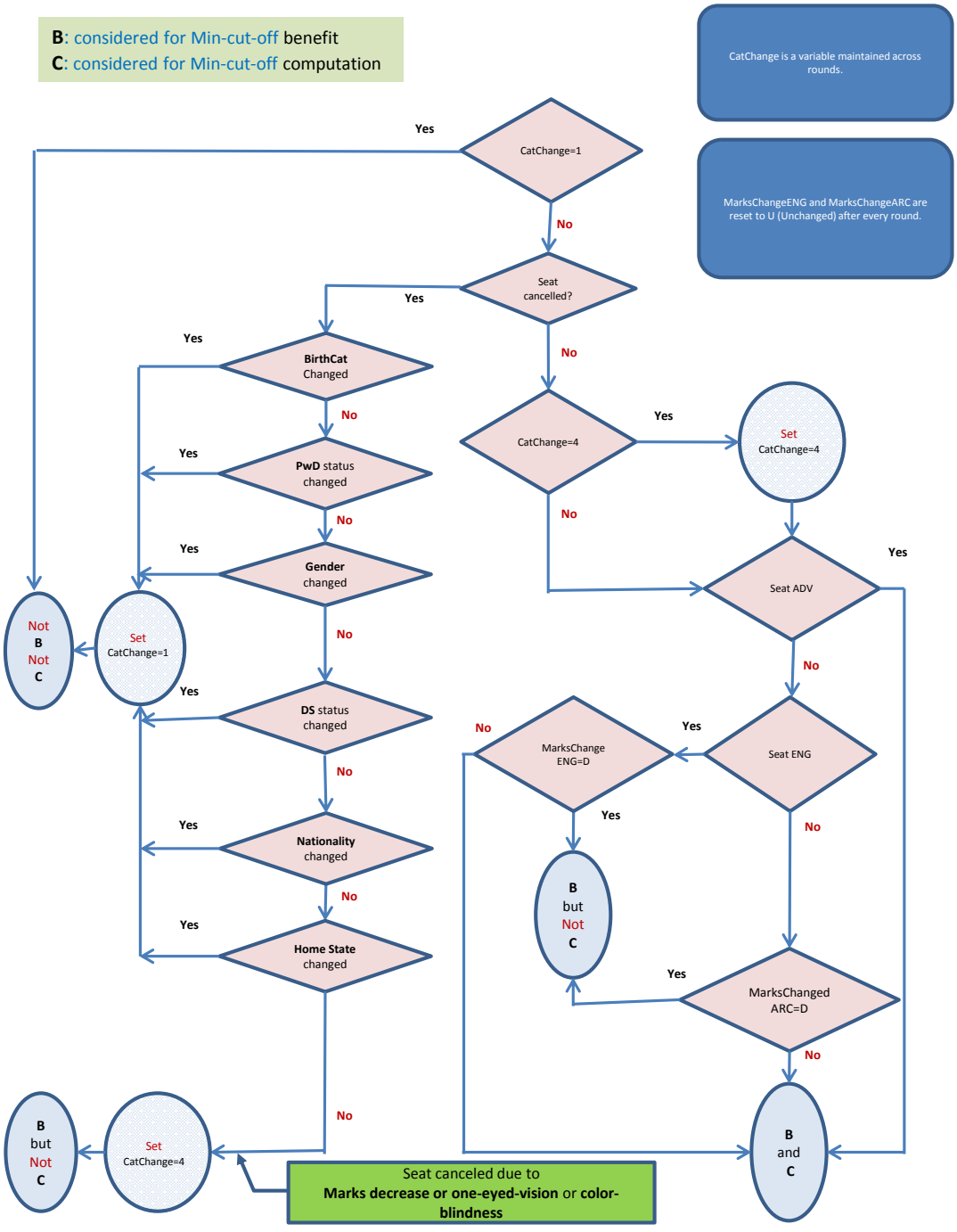



Figure 8.4: Setting and maintaining CAT-CHANGE flag across rounds

based on marks revision. We temporarily construct a reduced waitlist for a virtual program $p$ by removing the following kinds of candidates:

1. Those who opted for reject
2. Those who were awarded a seat through the Defense Services Priority Allocation (DS) (these candidates should be considered for min-cut-off of the DS virtual program and not for other virtual programs)
3. Those with CatChange=1
4. Those with CatChange=4
5. If $p$ is an engineering program then those with a decrease in the marks in the engineering exam. Conversely, if $p$ is an architecture program then those with decrease in subjects related to the architecture program.

We then set MIN-CUTOFF($p$) for the next round as follows:

- If the reduced waitlist is non-empty, MIN-CUTOFF($p$) is set to RANK($y, p$), where $y$ is the last candidate in the reduced wait list. Note again that, RANK here stores the extended merit list rank after revision. [4]
- If the reduced waitlist is empty, MIN-CUTOFF($p$) is set to 0. Thus, nobody applying to $p$ in the next round will get any min-cut-off benefit.

Table 8.1 states the conditions under which a candidate can be considered for MIN-CUTOFF benefit as well as the computation of MIN-CUTOFF.

## 8.5 Choice Filling

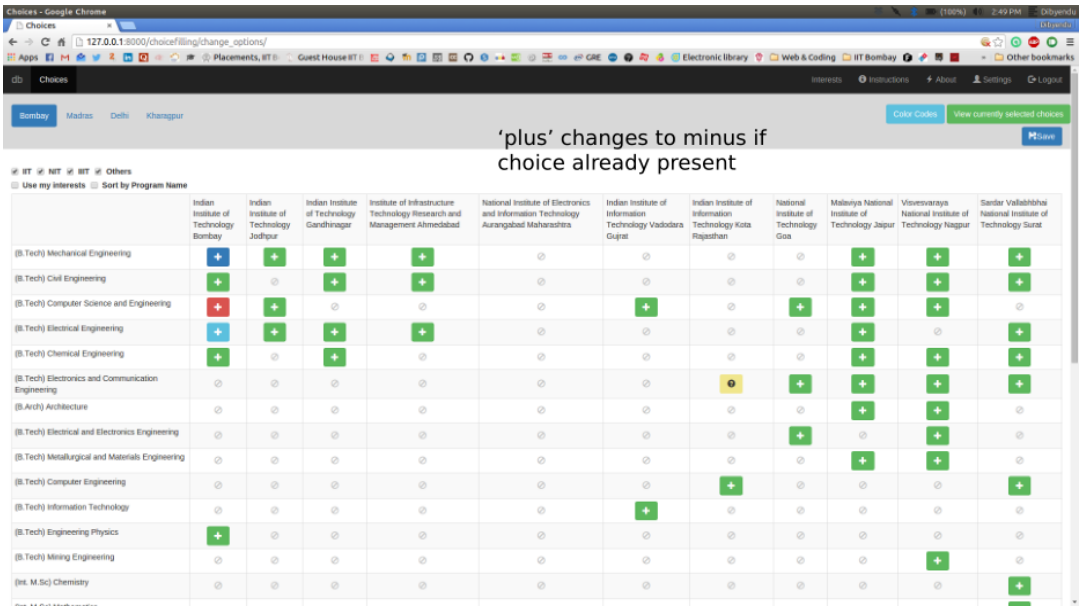
The current process of choice filling requires a candidate to fill choice one time. But how does one make 300 choices out of 600, especially when it can impact one's future life prospects? Choice filling ended up being (relatively speaking) one of the few weak links in the allotment process of 2015.

One way out is design an interface that makes this task simpler, giving information of each and every candidate. An example of the interface designed as a prototype in 2015 is shown below.

[4]Recall Definition 1. **Example**: If $p$ is an OBC-NCL virtual program then we use the last rank in the standard OBC-NCL merit list. On the other hand if $p$ is an SC virtual program then we use the last rank in the extended merit list, which, to recall is constructed by taking the SC rank list followed by the PC-SC rank list.

| MarksChange<br><br>(consider ENG if the allotted program is ENG, consider ARC if allotted program is ARC) | CatChange value | Min-cut-off benefit<br><br>(If given, a supernumerary seat may be created for a candidate if his/her rank is better than min-cut-off) | Consideration for Min-cut-off computation<br>("considered" implies that the candidate will be included in the reduced waitlist) |
|---|---|---|---|
| I/U | 1 | Not given | Not considered |
| I/U | 2 | Given | considered |
| I/U | 3 | Given | considered |
| I/U | 4 | Given | Not considered |
| D | 1 | Not given | Not considered |
| D | 2 | Given | Not considered |
| D | 3 | Given | Not considered |
| D | 4 | Given | Not considered |

Table 8.1: The conditions under which a candidate is considered for MIN-CUTOFF($p$) computation and MIN-CUTOFF($p$) benefits for a virtual program.




There were about 60,000 views with an average of 5 min per person per day in the 4 days before choices were accepted. There were absolutely no advertisements from JoSAA, IIT Bombay JEE Advanced Cell, or CSAB, and information spread purely by word of mouth.

## 8.6 Survey Questions

Candidates who wish to exit from the system after showing interest should be asked to fill out a survey before their fees are refunded. Here is a possible mini-survey design:

This survey is intended to obtain a detailed understanding of the performance of JoSAA 2015, and possibly suggest areas for potential further improvements in efficiency of seat allocation. Complete privacy of your data is guaranteed. There will be no penalties of any kind based on information entered here. We appreciate your help in providing us this vital information.

Why did you choose to not take-up your allocation of program ??? in round ??? of JoSAA 2015? Please select the appropriate option:

__ Want to write JEE again[5].
__ Admitted to other Institute
(i) Institute name _____ (ii) Date of admission notification _____
__ Other. (i) Please specify _____ (ii) Date of decision _____(select on a calendar)

Thank you for your assistance!

## 8.7 Detailed algorithm for handling DS candidates

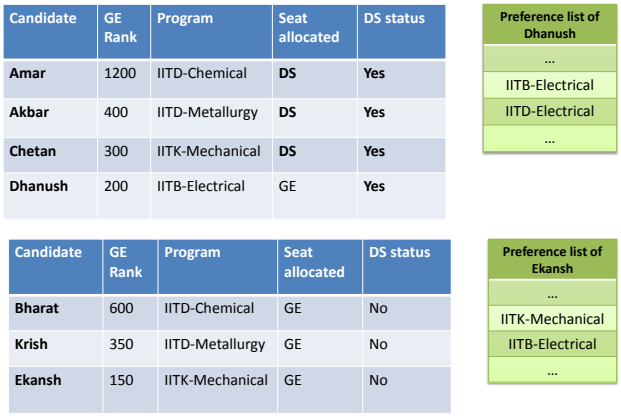
First we describe an example of a race condition that may arise while processing some DS candidates. Thereafter, we present a method for detecting a race condition. We conclude with a complete algorithm for handling DS candidates.

### 8.7.1 Example of Race Condition

Let Amar, Akbar, Chetan, and Dhanush be four DS candidates. At the end of the DA algorithm, Amar, Akbar, and Chetan get their program through a DS seat; But Dhanush gets his program IITB-Electrical through a GE seat because there were already two better ranked DS candidates who got DS seats in IIT Bombay. Moreover, let Dhanush happens to be the last ranked candidate getting IITB-Electrical. Let Bharat, Krish, and Ekansh be the last ranked GEN candidates in IITD-Chemical, IITD-Metallurgy, and IITK-Mechanical respectively. The details of all these seven candidates with the programs allocated by the DA algorithm is shown in Figure 8.5.

We now describe the processing of Amar, Akbar, and Chetan who got DS seat. These candidates have to be given OPEN seats. In order to accommodate Amar, we need to remove Bharat and this leads to Bharat getting some other less preferred program in the rejection chain. In a similar manner, Akbar gets IITD-Metallurgy after removal of Krish and Krish gets some other less preferred program in the rejection chain. Let us process Chetan now. Since Chetan got seat IITK-Mechanical through DS quota we need to remove Ekansh from IITK-Mechanical. Next preferred program for Ekansh is IITB-Electrical. Recall that Dhanush is the

[5]Keep this option only if this is allowed in the first place for people who accepted seat by paying fees and then changed their mind.



| Candidate | GE Rank | Program | Seat allocated | DS status |
|---|---|---|---|---|
| **Amar** | 1200 | IITD-Chemical | **DS** | **Yes** |
| **Akbar** | 400 | IITD-Metallurgy | **DS** | **Yes** |
| **Chetan** | 300 | IITK-Mechanical | **DS** | **Yes** |
| **Dhanush** | 200 | IITB-Electrical | GE | **Yes** |

| Preference list of Dhanush |
|---|
| ... |
| IITB-Electrical |
| IITD-Electrical |
| ... |

| Candidate | GE Rank | Program | Seat allocated | DS status |
|---|---|---|---|---|
| **Bharat** | 600 | IITD-Chemical | GE | No |
| **Krish** | 350 | IITD-Metallurgy | GE | No |
| **Ekansh** | 150 | IITK-Mechanical | GE | No |

| Preference list of Ekansh |
|---|
| ... |
| IITK-Mechanical |
| IITB-Electrical |
| ... |

Figure 8.5: Program allocation by the DA algorithm to 3 DS and 2 GE candidates.

last ranked candidate getting IITB-Electrical. Notice that though Dhanush is a DS candidate, he got OPEN seat in IITB-Electrical. So Ekansh will remove Dhanush from IITB-Electrical. So Dhanush will apply for his next preferred program which is IITD-Electrical. There are already two DS candidates Amar and Akbar for IITD. Since Dhanush has better rank than Akbar, and Akbar has better rank than Amar, so Dhanush will remove Amar from DS virtual program of IITD. So the processing of Amar in the past goes waste since he is not getting a DS seat in IITD (the seat vacated by Bharat for Amar goes waste, the seat remained vacant in the end and Bharat got a less preferred program). A similar example can be constructed wherein the DS candidate who initiates a rejection chain will have to be de-allocated. In any such situation, we should revert the rejection chain and allocate the DS candidate a supernumerary seat. In the current example, Chetan has to be given a supernumerary seat in IITK-Electrical program.

### 8.7.2 Detecting Race Condition

After processing each seat $s$, we need to check for race condition. Some key points to be noted about race condition are:

1. In the above example, if Dhanush had also applied for IITD-Chemical instead of IITD-Electrical, it would not have been a race condition. This is because, Dhanush will now simply occupy the space created by the processing of Amar in IITD-Chemical Open virtual program. Note that this doesn't result in any program being vacant and hence there is no race condition.
   Thus, for race-condition-free allocation, candidates in a processed seat may change but their programs should not.

2. The number of candidates in any DS virtual program can be variable due to the rule of supernumerary allocation to same ranked candidates.

We gather that for any candidate $x$ occupying a processed seat in old allocation (before processing of seat $s$), there must be a candidate $x'$ occupying a processed seat in new allocation such that $x$ and $x'$ have the same DS virtual program and $x.allotted\_program = x'.allotted\_program$. If this condition does not hold after processing of a seat $s$, we have a race condition and we will need to give the candidate occupying $s$ a supernumerary seat.

To summarize, we need to verify that the allotted programs to processed seats in Old allocation of a DS virtual program constitute a subset of allotted programs to processed seats in New allocation of the same DS virtual program. If this holds true for all DS virtual programs, then we can proceed with processing further seats, otherwise we need to revert to old allocation and give a supernumerary seat to the latest processed DS candidate.

To check the above condition, after we process a seat, we iterate over processed seats in old allocation (including the latest processed seat), for each allotted program of a processed seat, we find a matching processed seat in new allocation with the same allotted program. If at any point, no such match is found, we declare that there is a race condition and we need to revert to old allocation (before the processing of latest seat) and make one supernumerary seat.

### 8.7.3 Pseudocode

We now provide full details of the algorithm in this section.

#### Details of the algorithm

The algorithm maintains the set $D$ of seats in all DS programs. Since, due to ties in multiple DS candidates, number of seats in any program may change during the course of the algorithm, we keep $D$ as an instance of ALLOCATION object. The algorithm processes the seats from $D$ sequentially. At each stage, it also maintains a flag Is_processed (initialized to false) for each seat in $D$. After processing a seat, this flag is set to true. It picks an unprocessed seat $s$ from $D$ and does the following. Let $x$ be the candidate occupying the seat $s$ and $p$ be the program opted by him/her. Let $w$ be the worst rank candidate from OPEN category who has been assigned program $p$. $w$ is removed from program $p$ and the capacity of $p$ is reduced by 1. We now start DA algorithm with input $\{w\}$. Effectively, the candidate $w$ applies to the next program in his/her preference list. This generates a rejection chain resulting in a different allocation than before. Let the allocation before processing be OLD-ALLOCATION and allocation after processing be NEW-ALLOCATION. It is quite possible that the candidate corresponding to any processed seat in NEW-ALLOCATION.$D$ is different from that in OLD-ALLOCATION.$D$. However, if the set of programs associated to the processed seats of NEW-ALLOCATION.$D$ differs from that in OLD-ALLOCATION.$D$, we revert to NEW-ALLOCATION, create a supernumerary seat for $p$ and allocate it to $x$.

We need to consider each DS seat one-by-one, create their respective rejection chain and revert it if it causes race condition. To detect race condition after processing a seat $s$, all we need to do is to check if the multiset $S_1$ of allotted programs to processed seats in OLD-ALLOCATION is a subset of the multiset $S_2$ of allotted programs to processed seats in NEW-ALLOCATION. Note that, there cannot be a new processed seat created in NEW-ALLOCATION ($s$ is marked

processed in both OLD-ALLOCATION and NEW-ALLOCATION). Hence, $|S_2| = |S_1|$. Therefore, in order to detect race condition, all we need to do is to verify whether $S_1 = S_2$. If $S_1 \neq S_2$, then there is a race condition, else there is no race condition. Notice that $S_1$ and $S_2$ are multi-sets and so they should be matched element by element for equality. An element is a program (i.e. IITK_CS, IITK_CE etc. That is, consider branch names augmented with institute names).

Algorithm 5 presents the complete pseudocode of the algorithm for handling the rule for admission of DS candidates.

Notations

CANDIDATE($s$): candidate to whom DS seat $s$ is allocated. Set to $\phi$ if the seat is unoccupied.

PROGRAM($s$): OPEN virtual program corresponding to program allocated to candidate occupying DS seat $s$. Set to $\phi$ if the seat is unoccupied.

**Algorithm 5** Algorithm to incorporate the rule for admission under DS category

INPUTS:

The regular inputs for DA.

The Boolean SUPERNUMERARY-OK, which indicates whether a supernumerary seat may be created if a DS related problem occurs. Initially we run with SUPERNUMERARY-OK=False

Each seat has a Boolean flag IS-PROCESSED. We initialize it as IS-PROCESSED=False for all DS seats.

OUTPUTS:

The regular output of DA, as well as the waitlists for the DS virtual programs (these include the name of the actual program allotted to each candidate).

The list of stained programs $S$. If non-empty, this would be a trigger to get permission from the JAB Chairman and rerun with SUPERNUMERARY-OK = True.

1: **for all** Institutes **do**
2: Create a DS virtual program $I$.
3: $c(I) \leftarrow 2$
4: MERIT$(I) \leftarrow$ DS candidates in CML in CML order
5: WAITLIST$(I)$ will be an augmented waitlist, such that each entry will contain both a candidate as well as a program name.
6: **end for**
7: **for all** candidates $x$ with DS tag **do**
8: Create PREF$(x)$. For each program, first list virtual programs as usual as per birth category of $x$ based on Figure 8.1, then list corresponding institute DS virtual program (tag it with the relevant program).
9: **end for**
10: Run the DA algorithm
11: ALLOCATION $\leftarrow$ WAITLIST$(p)$ $\forall$ $p \in \mathcal{P}$ and $i(x)$ $\forall$ $x \in \mathcal{A}$ as per the output of the DA algorithm
12: ALLOCATION.$D \leftarrow$ List of seats in all DS programs in ALLOCATION. A seat data-structure includes a candidate, a program, and a Boolean flag IS-PROCESSED.
13: **for all** $s \in$ ALLOCATION.$D$ **do**
14: $s$.IS-PROCESSED $\leftarrow$ false
15: **end for**
16: $S \leftarrow$ Empty list ▷ List of stained programs $S$ is needed only if SUPERNUMERARY-OK = False
17: **while** $\exists s \in$ ALLOCATION.$D$ with ([$s$.IS-PROCESSED]= false and CANDIDATE$(s) \neq \phi$) **do**
18: OLD-ALLOCATION $\leftarrow$ ALLOCATION;
19: $L \leftarrow$PROCESSSEAT$(s)$
20: $Q \leftarrow$ Empty queue

21: **for all** $y \in L$ **do**
22: Remove $y$ from WaitList($p$)
23: Reject($y$) ▷ This adds $y$ to $Q$ as well
24: **end for**
25: Run DA with current $i(x')$, WaitList($p$) and $Q$ as inputs (other inputs as usual). During the run, if a DS candidate $x''$ is pushed out of a DS seat and the set of programs associated with the processed seats in that DS virtual program changes, add the corresponding OPEN virtual program to a list $S$ of stained virtual programs.
26: allocation ← Output of DA ▷ DA may maintain a list of affected DS virtual programs to make the loop below more efficient
27: to-revert ← IsRaceCondition(allocation,old-allocation)
28: **if** to-revert **then**
29: **if** supernumerary-ok **then**
30: allocation ← old-allocation
31: Increment $c(p)$ by 1;
32: **else**
33: Do nothing ▷ $S$ now includes both programs that lost a DS candidate after making room, as well as those that gained a DS candidate and may have a supernumerary seat (but will not be processed by the algorithm since that seat has been already processed).
34: **end if**
35: **end if**
36: allocation.$D$ ← List of seats in all DS programs according to allocation
37: **end while**
38: return allocation and $S$; ▷ Note that if a DS candidate has gotten a seat via DS in program $p$, he is not included in WaitList($p$). This must be read separately by reading the augmented waitlist WaitList($I$) (which specifies both the candidate and the program) where $I$ corresponds to the institute that hosts $p$.

39: **function** processSeat($s$)
40: $s$.Is-Processed $\leftarrow$ true;
41: $x \leftarrow$ candidate($s$);
42: $p \leftarrow$ program($s$);
43: Decrement $c(p)$ by 1;
44: $L \leftarrow$ List of all candidates who must be rejected from WaitList($p$) based on updated capacity, Min-Cutoff($p$) etc. ▷ In case of no ties, $L$ will contain one candidate if the last candidate does not clear Min-Cutoff($p$), and zero candidates otherwise.
45: Return $L$
46: **end function**
47:
48: **function** IsRaceCondition(new-allocation,old-allocation)
49: $S_1 \leftarrow$ multi set of allotted programs to processed seats in new-allocation.$D$ ▷ a program is institute+branch
50: $S_2 \leftarrow$ multi set of allotted programs to processed seats in old-allocation.$D$ ▷ a program is institute+branch
51: **if** $S_1 = S_2$ **then**
52: return False;
53: **else**
54: return True;
55: **end if**
56: **end function**

## 8.8 Validation Modules

An important challenge is to verify whether the output of the DA algorithm for a large test case (nearly 13 lakh candidates) satisfies various business rules. Validation modules provide effective ways to achieve this goal. Each validation module will take the input and output files of the DA algorithm for a round and verify whether the output is indeed correct for the given input. There are two types of validation modules.

### 8.8.1 Candidate specific validation modules

There are 6 validation modules required to ensure that allotment meets the business rules for the candidates.

#### Fairness

Suppose a candidate $c$ with CatChange = 2, CatChange = 3 or CatChange = 4 is allotted a program p which is her ith option in the choice list. Let $p_1,\cdots,p_{i-1}$ be the $i-1$ programs that rejected her. Then the rank of each candidate getting any one of the these programs must be superior to the rank of $c$. This module can be implemented efficiently with the help of the closing rank as follows. The closing rank of each of the virtual programs that rejected $c$ must be better than the rank of $c$.

#### Quota Eligibility

NIT program has home state (HS) and other state (OS) quota. Similarly for other GFTIs, each program has home state (HS) and all India (AI) quota. Quota eligibility requires the following condition to be guaranteed for HS quota (similarly for OS and AI). A candidate can be allotted a seat from HS quota of a program only if she is eligible for the HS quota of that program; conversely, if the candidate is not eligible in the OS quota, she should not be awarded the seat. (This description is different for AI).

#### Category Eligibility

The aim of this module is to verify that a candidate must be assigned seat from the category for which she is eligible. For example, a SC candidate who does not appear in CRL cannot be assigned a seat from any OPEN virtual program. In a similar manner, a GEN candidate cannot be assigned a seat from an SC virtual program.

#### Candidate willingness

After a given round, the candidates who are allotted programs may opt for any of the following options: Freeze, Float, Slide, or Reject. These options must be respected while allocating programs to them in the following rounds.

**Seat guarantee in later rounds (Min-Cut-off)**

Let c be a candidate that had CatChange = 2 or CatChange = 3. Let c be allotted a seat in program p in round i. In round i+1, c must get a seat in program p or a program that she prefers to p, if she is still eligible to get p and any one of the following holds:

- p is an ENG program and MarksChangeEng of c is not "D"
- p is an ARC program and MarksChangeArc of c is not "D"

**Restricted fairness for category change**

Let c be a candidate whose CatChange = 1. Both of the following must be true

- If rank of c is better than min-cut-off for program p but she does not get program p even after applying to p, then no candidate can get program p with rank worse than min-cut-off of p.
- No candidate with CatChange = 1 can get p if his/her rank is worse than that of c.

### 8.8.2 Program specific validation modules

There are two broad validation modules to verify whether the allotment meets the business rules of the programs.

**De-reservation**

During multiple runs of DA in a round, the capacity of a virtual program may change due to de-reservation. Thus within a round, the seat matrix may change. However, the capacities of the sum of all virtual programs must remain unchanged. The checks we can do here are:

- Sum of capacities of all virtual programs is the same before and after de-reservation. In fact, the following equation should hold for each virtual program p

  $$\text{NewCap} = \text{InitCap} + \text{No. of seats de-reserved to p} - \text{No. of seats de-reserved from p}$$

- A virtual program from which de-reservation happened cannot have any supernumerary seat.
- A virtual program from which de-reservation happened must not be denied to any candidate eligible for that virtual program. That is, there should be no eligible candidate who applied to this program and didn't get it.

**Supernumerary**

The number of candidates getting a virtual program may be more than the capacity of the virtual program. The surplus candidates are given supernumerary seats. The cause of supernumerary seats may be any one of following:

- Multiple candidates at closing rank (EQ)
- Min cut-off criteria (MC)
- Foreign nationals (FR)
- DS race condition (RC)

In order to verify the cause of supernumerary seats, additional information has been provided in the revised allotment table. This verification can be done in two steps:

1. Verify the total supernumerary seats: That is, verify that

$$\text{waitlist size} - \text{NewCap} = \text{number of supernumerary seats}$$

2. Verify whether the supernumerary seats themselves are justified. The second check is done on a case-by-case basis as follows:

   EQ: Check if the candidates marked EQ all have equal rank that is also the closing rank of the virtual program

   MC: If any candidate in a virtual program is getting marked MC then all of the following must hold:
   - All candidates in that program must be better than min cut off
   - There can be no CatChange=1 candidate in that program
   - There can be no RC or EQ candidates in that virtual program [6]. [7]

   FR: Two checks need to be performed:
   - The candidate is a foreign national
   - One of the following must hold:
     * The candidate has a rank better than or equal to closing rank
     * The candidate has a rank better than min-cut-off

   Note that FR should only be marked if the foreign national is actually getting a supernumerary seat. That is, if a foreign national gets a seat in a program due presence of unfilled seats, then such a seat is not considered supernumerary and hence we must have SupNumReason = NA. Also, even if a foreign national is getting a seat due to any other reason (like EQ or MC), he/she must be marked FR.

[6] If closing rank of a virtual program is better than min-cut-off, the program is written in a way such that no candidate at closing rank is marked EQ. Such candidate(s) should be marked MC if applicable

[7] This follows from the way we process DS seats. A DS candidate won't be able to displace any GEN candidate since all of their ranks are better than min-cut-off of that GEN virtual program.

RC: The number of candidates marked RC is bounded from above by the number of candidates getting that program with DS = Y

### 8.8.3 Test Assisting modules

In order to assist the people engaged in testing the DA algorithm, the Allotment table will be augmented with 2 additional columns (Flag, SupNumReason). These columns will be used to compute min-cut-off of virtual program for the next round, provide and verify the reason for the creation of supernumerary seats in a virtual program. In addition to the augmented allotment table, the following table will also be output by the DA implementation.

#### Program Statistics

For each virtual program, the following information will be mentioned.

- Opening rank of the virtual program.
- Closing rank of the virtual program.
- MinCutOff of the virtual program.
- Total candidates allotted to the virtual program.
- Initial and final capacity of the virtual program
- The number of seats that got dereserved to and from the virtual program[8]
- The number of supernumerary seats created in the virtual program.

#### Allotment comparing module

In order to compare two allotments (produced by two different implementations), a module will be required that will highlight the candidates whose allotment differs in the two allocations. This module will help testing teams and may help fixing of a bug, if any. Such a module was a part of the IITK implementation of the DA algorithm and it was very helpful.

## 8.9 Implementation Details of MRDA

In this section we present the implementation details of the algorithm. In particular, we elaborate on the input, output, and the interface of the DA algorithm along with its interaction with the reporting center during any round.

[8]These are two different columns DereserveTo and DereserveFrom

### 8.9.1 Algorithm: interface and interactions

Seat allocations happen in multiple stages (or rounds). The key difference between the first round and subsequent rounds is that some candidates have seats that the Joint Seat Allocation Authority (JoSAA) agrees to a guarantee - seats offered in prior rounds (e.g. "freeze") will continue to be available. Having a single implementation of the DA algorithm (with updated inputs) that works for every round is better than having multiple implementations. We have to validate only one implementation. We call such an implementation a "generic" implementation. Figure 8.6 shows the interface (input/output) of the generic DA algorithm.

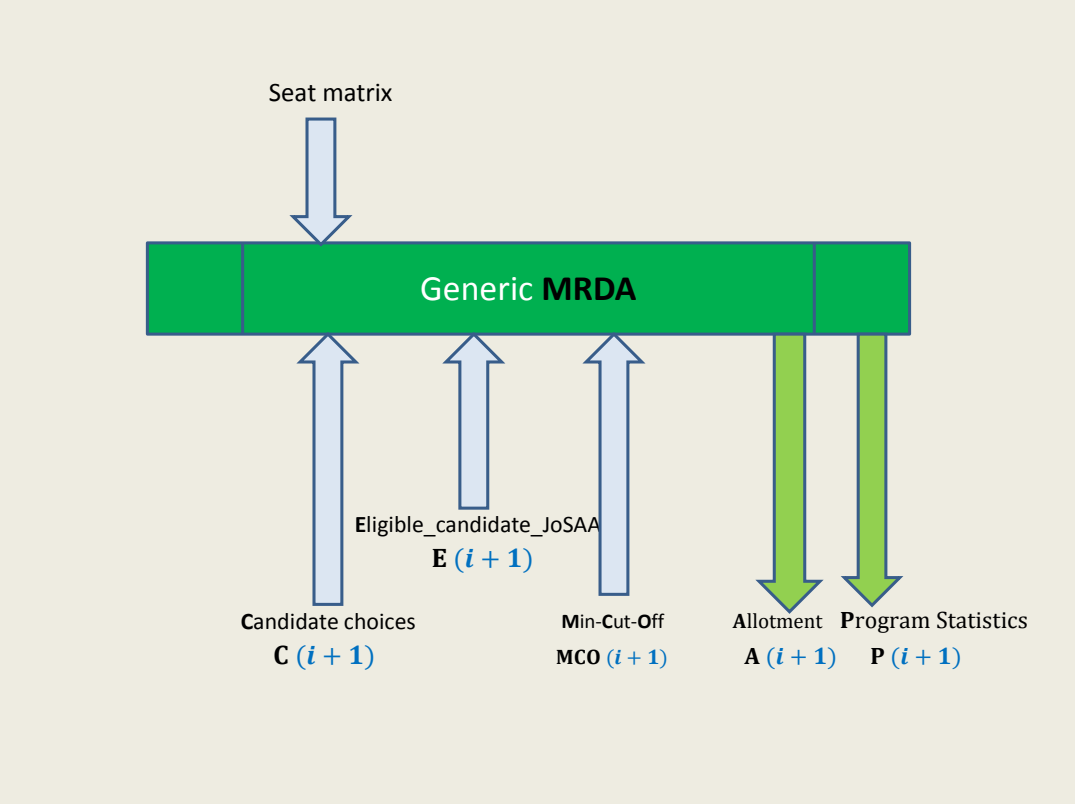



Figure 8.6: The interface (input/output) of the generic DA algorithm

We provide below a summary of four tables that will serve as input to the generic implementation of DA algorithm, and also two output tables.

- **Seat_Matrix**: The table stores the list of programmes of various institutes and their capacities. Seat matrix will remain the same for each round. (However, internally, during multiple runs, this matrix may change due to de-reservation.) Seat matrix is obtained from the institutes.

- **Candidate_Choices**: This table stores the choices of each candidate. The table for the (N+1)th round is defined by the table for the (N)th round, the candidate willingness option (freeze/float/slide/reject). The Validity field can be updated at the reporting center. The willingness information is not a part of the candidate_choices table, but is available in the 63rd column of the Eligible_Candidate_JoSAA table mentioned below. The initial table is obtained after JEE Mains Rank List and JEE Advanced Rank List are declared. N is indexed starting from 1.

- **Eligible_Candidate_JoSAA**: This table stores personal and other eligibility information of candidates. The information is collected at the time of the registration for the exam. For JEE Mains, this information can be quite dated, being collected almost six months in advance. The information in this table is vulnerable, because, for example, the OBC-NCL status of candidates can change, possibly due to change in income. This table will be provided externally at the beginning of every round. For round (N+1), N $>$ 1, this table will have meaningful information in the following fields.

  - CatChange: This field will take value from {1,2,3,4}. For the 1st round, the value will be 2.
  - MarksChangeENG:
    These fields will be one of "U/D/I" depending upon whether the board-marks-eng of the candidate underwent no-change/decrease/increase after the (N)th round. For the first round, both of these fields will be U for all candidates. These fields are present in the 61st and 62nd columns.
  - MarksChangeARC:
    These fields will be one of "U/D/I" depending upon whether the board-marks-arc of the candidate underwent no-change/decrease/increase after the (N)th round. For the first round, both of these fields will be U for all candidates. These fields are present in the 61st and 62nd columns.
  - Decision:
    This field will be one of Freeze/Float/Slide/Reject depending upon the option exercised by the candidate after a seat is allocated. This field will be used for computing the choice list of the candidate for the (N+1)th round.

- **Min-cut-off**

  This table will store min-cut-off for each virtual programme. For the first round, it will be set to 0 for all virtual programmes. Min-cut-off for a virtual programme prior to the execution of round (N+1) will be computed using the allotment table of the (N)th round and the four additional fields (column 60, 61, 62 and 63) of the Eligible_Candidate_JoSAA file mentioned above.

Note: For rounds other than the first round, Eligibile_Candidate_JoSAA table and Candidate_Choices table will be the only external input.
The output of the algorithm for each round will be the following two tables.

- **Allotment**

  This table will store the details of the programme, if any, allotted to each candidate. The allotment table has two significant columns for purposes of multiround DA. Details appear below.

- **Program Statistics**

  This table will store the details of the program statistics (opening rank, closing rank, supernumerary information, and de-reservation information).

Please refer to Figure 8.6 for the interface of the generic DA algorithm and refer to Figure 8.7 for various activities that take place during any round.

Figure 8.7 depicts the execution of the algorithm and activities that take place during $(i+1)$th round.

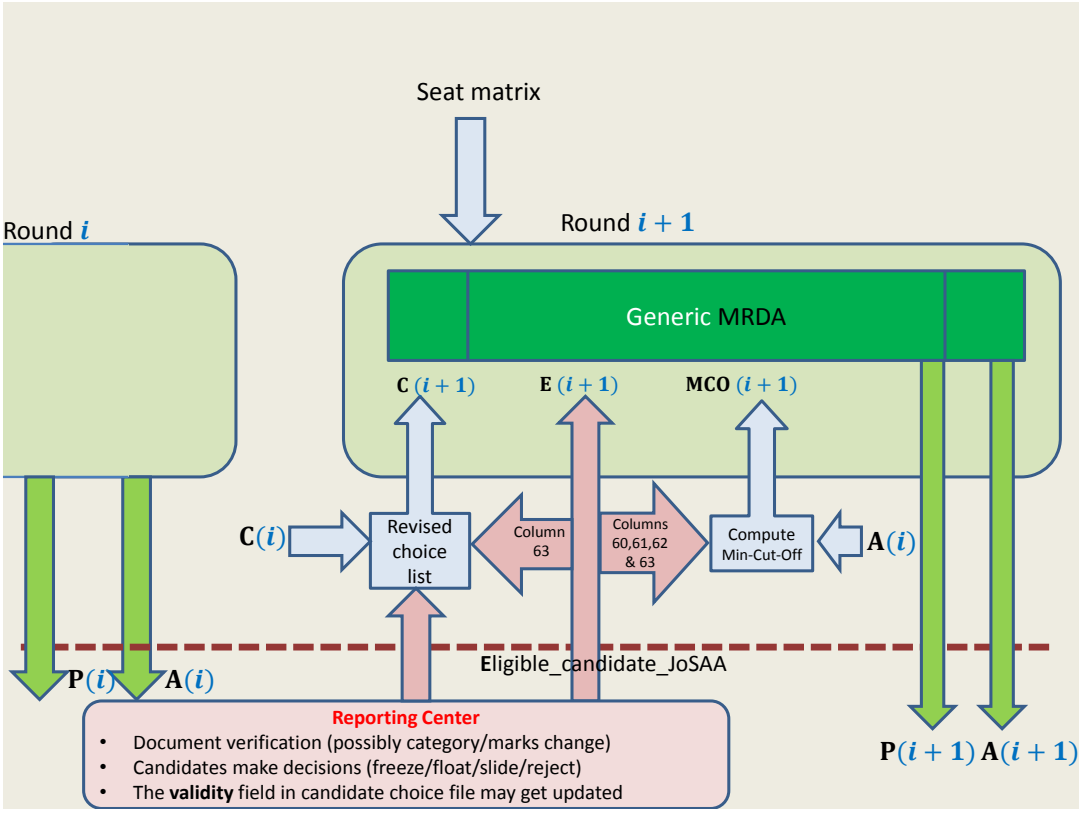



Figure 8.7: The activities during $(i+1)$th round

### 8.9.2 Input format

The complete details of the three input tables are given below.

**Seat Matrix**

| S. No. | Column Name | Type | Description |
|---|---|---|---|
| 1. | Quota | char(2) | AI/HS/OS |
| 2. | InstCd | char(3) | Institute Code |
| 3. | BrCd | char(4) | Branch Code |
| 4. | OP | int | Open Seats |
| 5. | OP_PwD | int | Open-PwD Seats |
| 6. | SC | int | SC Seats |
| 7. | SC_PwD | int | SC-PwD Seats |
| 8. | ST | int | ST Seats |
| 9. | ST_PwD | int | ST-PwD Seats |
| 10. | OBC_NCL | int | OBC-NCL Seats |
| 11. | OBC_NCL_PwD | int | OBC-NCL-PwD Seats |
| 12. | Total | int | Total Seats |
| 13. | StCd1 | char(2) | State code of Eligibility |
| 14. | StCd2 | char(2) | State code of Eligibility |
| 15. | StCd3 | char(2) | State code of Eligibility |
| 16. | StCd4 | char(2) | State code of Eligibility |

**Candidate Choice**

| S. No. | Column Name | Type | Description | Remarks |
|---|---|---|---|---|
| 1 | RollNo | char(8) | JEE (Main) Roll number | |
| 2 | OptNo | int | Option No | |
| 3 | Instcd | char(3) | Institute Code | |
| 4 | BrCd | char(4) | Branch Code | |
| 5 | Validity | char(1) | NULL (default)- if choice is valid<br>N - Choice is not valid | This field will be updated at the time of Document verification |

**Candidate Table**

| S. No. | Column name | Type and Length | Description | Remark |
|---|---|---|---|---|
| 1. | RollNo | char(8) | JEE(Main) Roll Number | |
| 2. | AppNo | char(8) | JEE(Main) Application No | |
| 3. | NAME | varchar | CandidateâĂŹs Name | |
| 4. | MNAME | varchar | Mother Name | |
| 5. | FNAME | varchar | Father Name | |
| 6. | GName | varchar | GuardianâĂŹs Name | |
| 7. | SCode | char(2) | State Code of Eligibility | |
| 8. | Gender | char(1) | 1-Male, 2-Female,3-Transgender | |
| 9. | DOB | char(10) | Date of Birth (DD/MM/YYYY) | |
| 10. | CAT | char(2) | GN/SC/ST/BC | |
| 11. | PwD | char(1) | 1-YES, 2-NO | |
| 12. | Nationality | char(1) | 1-Indian, 2- Foreign | |
| 13. | AI_Eng_CRL_Rank | float | JEE(Main) All India Eng CRL Rank | |
| 14. | AI_Eng_OBC_NCL_Rank | float | JEE(Main) All India Eng OBC-NCL Rank | |
| 15. | AI_Eng_SC_Rank | float | JEE(Main) All India Eng SC Rank | |
| 16. | AI_Eng_ST_Rank | float | JEE(Main) All India Eng ST Rank | |
| 17. | AI_Eng_CRL_PD_Rank | float | JEE(Main) All India Eng CRL-PwD Rank | |
| 18. | AI_Eng_OBC_NCL_PD_Rank | float | JEE(Main) All India Eng OBC-PwD Rank | |
| 19. | AI_Eng_SC_PD_Rank | float | JEE(Main) All India Eng SC-PwD Rank | |
| 20. | AI_Eng_ST_PD_Rank | float | JEE(Main) ) All India Eng ST-PwD Rank | |

| S. No. | Column name | Type and Length | Description | Remark |
|---|---|---|---|---|
| 21. | Eng_Rem_Symb | char(1) | JEE (Main) - BE/B.Tech Eligibility<br>Remark symbol<br>'*' - Eligible under CRL<br>'=' - Eligible under OBC-NCL seats only<br>'+' - Eligible under SC/ST/PwD seats only<br>'$' - Eligible for OBC-NCL-PwD seat only<br>'%' - Eligible for GEN-PwD seat only<br>'N' - Non Eligible for Seat Allocation | |
| 22. | AI_Arc_CRL_Rank | float | JEE(Main) All India Arc CRL Rank | |
| 23. | AI_Arc_OBC_NCL_Rank | float | JEE(Main) All India Arc OBC-NCL Rank | |
| 24. | AI_Arc_SC_Rank | float | JEE(Main) All India Arc SC Rank | |
| 25. | AI_Arc_ST_Rank | float | JEE(Main) All India Arc ST Rank | |
| 26. | AI_Arc_CRL_PD_Rank | float | JEE(Main) All India Arc CRL-PwD Rank | |
| 27. | AI_Arc_OBC_NCL_PD_Rank | float | JEE(Main) All India Arc OBC-PwD Rank | |
| 28. | AI_Arc_SC_PD_Rank | float | JEE(Main) All India Arc SC-PwD Rank | |
| 29. | AI_Arc_ST_PD_Rank | float | JEE(Main) All India Arc ST-PwD Rank | |

| S. No. | Column name | Type and Length | Description | Remark |
|---|---|---|---|---|
| 30. | Arc_Rem_Symb | char(1) | JEE (Main) - B Arch/B.Planning Eligibility<br>Remark symbol<br>'* - Eligible under CRL<br>'=' - Eligible under OBC-NCL seats only<br>'+' - Eligible under SC/ST/PwD seats only<br>'$' - Eligible for OBC-NCL-PwD seat only<br>'%' - Eligible for GEN-PwD seat only<br>'N' - Non Eligible for Seat Allocation | |
| 31. | Adv_RollNo | char(8) | JEE(Advanced) Roll Number | |
| 32. | Adv_RegNo | char(10) | JEE(Advanced) Registration Number | |
| 33. | Adv_CRL_Rank | float | JEE(Advanced) Common Rank List | |
| 34. | Adv_OBC_NCL_Rank | float | JEE(Advanced) OBC-NCL Rank | |
| 35. | Adv_SC_Rank | float | JEE(Advanced) SC Rank | |
| 36. | Adv_ST_Rank | float | JEE(Advanced) ST Rank | |
| 37. | Adv_CRL_PD_Rank | float | JEE(Advanced) CRL-PwD Rank | |
| 38. | Adv_OBC_NCL_PD_Rank | float | JEE(Advanced) OBC-NCL-PwD Rank | |
| 39. | Adv_SC_PD_Rank | float | JEE(Advanced) SC-PwD Rank | |
| 40. | Adv_ST_PD_Rank | float | JEE(Advanced) ST-PwD Rank | |

| S. No. | Column name | Type and Length | Description | Remark |
|---|---|---|---|---|
| 41. | Adv_Rem_Symb | char (1) | JEE (Advanced) - Eligibility<br>Remark symbol<br>'*' - Eligible under CRL<br>'=' - Eligible under OBC-NCL seats only<br>'+' - Eligible under SC/ST/PwD seats only<br>'$' - Eligible for OBC-NCL-PwD seat only<br>'%' - Eligible for GEN-PwD seat only<br>'P' - Eligible for Preparatory seat<br>'N' - Non Eligible for Seat Allocation | |
| 42. | Adv_IsPrep | char(1) | Is eligible for Preparatory (1-Yes, 2-No) | |
| 43. | Adv_Prep_SC_Rank | float | Preparatory Rank for SC candidates | |
| 44. | Adv_Prep_ST_Rank | float | Preparatory Rank for ST candidates | |
| 45. | Adv_Prep_CRL_PD_Rank | float | Preparatory Rank for all PwD candidates | |
| 46. | Adv_Prep_OBC-NCL_PD_Rank | float | Preparatory Rank for OBC-NCL<br>PwD candidates | |
| 47. | Adv_Prep_SC_PD_Rank | float | Preparatory Rank for SC-PwD candidates | |
| 48. | Adv_Prep_ST_PD_Rank | float | Preparatory Rank for ST-PwD candidates | |
| 49. | Adv_AAT_Status | char(1) | 1-Qualified, 2-Not Qualified | |
| 50. | Adv_DS | char(1) | DS Status (1-Yes, 2-No) | |
| 51. | Adv_colour blind | char(1) | 1-Yes, 2-No | |
| 52. | Adv_OneEyedVision | char(1) | 1-Yes, 2-No | |
| 53. | Adv_Top_20 | char(1) | 1-Yes, 2-No | |

| S. No. | Column name | Type and Length | Description | Remark |
|---|---|---|---|---|
| 54. | Board_Mark_Eng | float | Aggregate % age of class 12th marks as per JEE(Main) Paper-1 criteria | |
| 55. | Board_Mark_Arc | float | Aggregate % age of class 12th marks as per JEE(Main) Paper-2 criteria | |
| 56. | Board_Mark_Adv | float | Aggregate % age of class 12th marks | |
| | | | as per JEE(Advanced) criteria | |
| 57. | Board_RollNo | varchar | Board Roll No | |
| 58. | Board_Year_Passing | varchar | Year of passing of class 12th or equivalent | |
| 59. | Board_Name | varchar | School Board Name | |
| 60. | CatChange | char(1) | Category Change<br>1-Yes<br>2-No (Default)<br>3-Yes but no penalty | |
| 61. | MarksChangeENG | char(1) | Change in Marks<br>I: Increase,<br>D: Decrease,<br>U:Unchanged(Default) | Satus of marks change in Board_marks_ENG (from previous round) |
| 62. | MarksChangeARC | char(1) | Change in Marks<br>I: Increase,<br>D: Decrease,<br>U:Unchanged(Default) | Satus of marks change in Board_marks_ARC (from previous round) |
| 63. | Decision | char(2) | FR: Freeze<br>FL: float<br>SL: Slide<br>RJ: Reject | |

### 8.9.3 Output format

**Allotment Table**

| S. No. | Column name | Type and Length | Description | Remark |
|---|---|---|---|---|
| 1. | RoundNo | int | Round No | |
| 2. | RollNo | char(8) | JEE(Main) Roll Number | |
| 3. | Birth_Cat | char(2) | CandidateâĂŹs birth category | |
| 4. | Optno | int | Option No | |
| 5. | InstCd | char (3) | Institute Code | |
| 6. | BrCd | char(4) | Branch Code | |
| 7. | Rank | float | Rank used for seat allocation | |
| 8. | AllottedCat | char(4) | Allotted Category. 8 possibilities:<br>OPNO, OPPH, BCNO,<br>BCPH, SCNO, SCPH,<br>STNO, STPH | First two characters<br>correspond to the birth category<br>and the second two correspond<br>to the PwD status of the candidates<br>for which this program is reserved |
| 9. | AllottedQuota | char(2) | Allotted Quota (AI/HS/OS) | |
| 10. | Flag | char(1) | Four possibilities:<br>N: Normal<br>D: DS<br>F: Foreign<br>P: Preparatory<br>(Default: N) | D: DS seat is given to the candidate.<br>A DS candidate can also receive<br>a "N" normal seat. |
| 11. | SupNumReason | char(2) | NA- Not Applicable (Default)<br>EQ- Closing rank equality<br>FR- Foreign national<br>RC- DS consideration<br>MC- Min cut off | If 5 candidates are at the<br>same closing rank, any 4<br>of them can be marked as EQ.<br>Similarly, if 5 supernumerary seats<br>are created due to min-cut-off,<br>any 5 candidates clearing<br>min-cut-off and whose<br>category didn't change<br>can be marked as MC |

**Program Statistics**

| S. No. | Column name | Type and Length | Description | Remark |
|---|---|---|---|---|
| 1. | Quota | char(2) | Quota | HS/OS/AI |
| 2. | InstCd | char(3) | Institute Code | |
| 3. | BrCd | char(4) | Branch Code | |
| 4. | VCategory | char(4) | Category of virtual programme. One among OPNO, OPPH, BCNO, BCPH, SCNO,SCPH, STNO, STPH, and DSNO | DSNO if the programme is DS virtual programme. BrCd should be 0000 then |
| 5. | OpeningRank | float | Opening Rank | |
| 6. | ClosingRank | float | Closing Rank | |
| 7. | MinCutOff | float | The min cut off used for allotment in the current round. | This field is computed in the beginning of the current round using the output of the previous round. |
| 8. | TotalAllotted | int | The number of candidates to whom the programme is allotted | |
| 9. | InitCap | int | Initial capacity | |
| 10. | NewCap | int | Capacity after de-reservation | Includes non-supernumerary seats given to DS candidates |
| 11. | DeReserveFrom | int | Number of seats that got de-reserved from this virtual programme | |
| 12. | DeReserveTo | int | Number of seats that got de-reserved to this virtual programme | |
| 13. | SuperNum | int | Number of supernumerary seats created in this virtual programme | Includes supernumerary seats created due to all possible reasons. |

## 8.10 Some statistics of Joint Allocation 2015

In this section, we present some of the outcome of the Joint Seat Allocation conducted in 2015.

### Decisions made by Candidates

The seat allocation for round 1 was announced on the morning of 7th July. There were only 112 seats vacant (all from the NITs). The candidates offered seats were supposed to report to the reporting center by the evening of 12th July. At that time the candidates were required to make their decision of freeze, float, or slide. (FR/FL/SL). Many candidates did not turn up and so their decision was recorded as a reject (RJ).

The following is the distribution of the decision of candidates who were offered seats in Round 1.

| Institute-Type | Freeze(FR) | Float(FL) | Slide(SL) | Reject(RJ) | Total |
|---|---|---|---|---|---|
| **IIT** | 2837 | 5203 | 1406 | 587 | 10033 |
| **NIT** | 3376 | 7823 | 2439 | 4170 | 17808 |
| **IIIT** | 203 | 1154 | 59 | 812 | 2228 |
| **Other GFTI** | 432 | 1692 | 193 | 1603 | 3920 |

**Note:** The number of candidates in round 1 rejecting the offers for IITs, NITs, IIITs, and other GFTIs are, respectively, 6%, 24%, 36%, 41%.

### Statistics at Reporting Centers

At the time of the reporting, it was discovered that the CAT-CHANGE flag had to be reset for several candidates. In the meanwhile, CBSE also provided the revised rank for certain candidates.

Following is the statistics at the end of round 1.

| | IIT | NIT | IIIT | Other GFTI |
|---|---|---|---|---|
| **Number of Candidates Allotted in Round 1** | 10033 | 17808 | 2228 | 3920 |
| **Number of Candidates Reported** | 9446 | 13638 | 1416 | 2317 |
| **Number of Seat Cancellations** | 31 | 168 | 14 | 25 |

### Statistics of candidates whose marks were changed by CBSE

The following is the statistics of candidates whose marks were changed by CBSE after the declaration of the allotment of round 1.

| Marks Change ENG | Marks Change ARC | No. of Candidates |
|---|---|---|
| Decreased | Unchanged | 4 |
| Increased | Decreased | 3 |
| Increased | Increased | 564 |
| Increased | Unchanged | 4847 |
| Unchanged | Increased | 34 |
| Unchanged | Unchanged | 1298765 |

**Note:** It was found that out of those whose marks increased, 119 got better preference in round 2 allotment. However, the reason of this may not be attributed solely to marks increase. It could also be possibly because of many vacancies that got created due to "no-show" of candidates in round 1.

### Statistics of candidates suffering a change in "CatChange" field

#### Statistics of candidates with CatChange=1

| **Reason** | **No. of Candidates** |
|---|---|
| OBC-NCL → GEN | 55 |
| SC → GEN | 1 |
| OBC-NCL-PwD → OBC-NCL | 24 |
| OBC-NCL-PwD → GEN | 1 |
| GEN-PwD → GEN | 22 |
| ST-PwD → ST | 1 |
| SC-PwD → SC | 7 |
| **Change of Home State** | 20 |
| **Total** | 131 |

**Note:** "CatChange=1" candidates may compete only for the vacant seats and hence suffer a huge penalty. However, 108 out of 131 Candidates got a seat in Round 2 allotment. Hence the penalty of Category Change=1 was not too severe for these candidates. This may also be due to huge vacancy created in round 1 due to "no-show" of many candidates.

#### Statistics of candidates with CatChange=3

| **Reason** | **No. of Candidates** |
|---|---|
| ColorBlind Status (No → Yes) | 26 |
| OneEyedVision (No → Yes) | 13 |
| OBC-NCL → GEN | 27 |
| ST → GEN | 1 |
| **Change of Home State** | 62 |
| **Total** | 129 |

#### Statistics of candidates with CatChange=4

| **Reason** | **No. of Candidates** | **No. of Candidates who could get a seat in round 2** |
|---|---|---|
| **ADV_TOP_20** (Yes → No) | 27 | 5 |
| **Colorblind Status** (No → Yes) | 1 | 1 |
| **Disqualified candidates** | 3 | 0 |

**Note:** Only 6 out of 28 "CatChange=4" candidates who were eligible to get a seat were offered a seat in Round 2 allotment. It may appear surprising, because it means the candidates who got rejected from IIT seats were not able to get any seats in other institutes. But it was found that majority of them were preparatory course candidates and their JEE mains rank were indeed very low.

### Statistics of candidates who got their ENG, ARC Ranks after Round 1

| **Description** | **JEE Advanced** qualified candidates | **JEE Advanced** not qualified candidates |
|---|---|---|
| Total no. of candidates | 5 | 611 |
| No. of candidates who filled up choices | 4 | 77 |
| No of candidates who got the seat | 2 (1 got IIT and 1 got NIT) | 1 |

### Statistics of supernumerary seats created in Round 2 allocation

There were only 18 supernumerary seats and that too due to foreign nationals in IITs. There were no supernumerary seats created due to Min-Cutoff. This could be explained at least partly by the huge vacancy created in round 1 due to "no-show" of many candidates.

### Statistics of vacant seats over 3 rounds

As follows from the following table, the vacancies reduced after each round.

| **Seat Type** | **Round 1** | | **Round 2** | | **Round 3** | |
|---|---|---|---|---|---|---|
| | Allotment | Vacancy | Allotment | Vacancy | Allotment | Vacancy |
| **IITs** | 10006 | 608 | 10006 | 130 | 10006 | 49 |
| **IITs supernumerary** | 27 | - | - | - | - | - |
| **NITs** | 17808 | 4338 | 17744 | 1841 | 17736 | 1153 |
| **IIITs** | 2228 | 826 | 2228 | 507 | 2228 | 318 |
| **Other GFTIs** | 3920 | 1628 | 3920 | 1257 | 3920 | 1156 |
| **Total** | 33989 | 7410 | 33916 | 3735 | 33908 | 2676 |

# Bibliography


[1] Atila Abdulkadiroğlu, Parag A Pathak, and Alvin E Roth. The New York city high school match. American Economic Review, pages 364–367, 2005.

[2] Péter Biró. Student admissions in Hungary as Gale and Shapley envisaged. University of Glasgow Technical Report TR-2008-291, 2008.

[3] L. E. Dubins and D. A. Freedman. Machiavelli and the Gale-Shapley algorithm. The American Mathematical Monthly, 88(7):485–494, 1981.

[4] David Gale and Lloyd Shapley. College admissions and the stability of marriage. The American Mathematical Monthly, 69(1):9–15, 1962.

[5] JAB and CSAB. Business rules for joint seat allocation for the academic programs offered by the IITs, ISM, NITs, IIITs and Other-GFTIs for the academic year 2015-16. Approved by all stakeholders, Jan 2015.

[6] Elliott Peranson and Alvin E. Roth. The Redesign of the Matching Market for American Physicians: Some engineering aspects of economic design. American Economic Review, 89(4):748–780, September 1999.

[7] Alvin E Roth and Marilda A Oliveira Sotomayor. Two-sided matching: A study in game-theoretic modeling and analysis. Number 18 in Econometric Society Monographs. Cambridge University Press, 1992.