A desktop application suited to schools, colleges, institutions of higher learning or any other institution with an operational structure akin to a tutors-for-learners-across classes paradigm.

**A SYNOPSIS (OF SORTS)**

**OLU’TOMI I. OWOEYE**

**CREATOR, DIRECTOR, DEVELOPER.**

**1.0**

**TIMETABLE GENERATOR/SCHEDULER**

**SHELVA**

# BACKGROUND: THE AUTHOR/CREATOR AND THE PROJECT

**Oluwatomilayo Inioluwa OWOEYE (Olu’tomi** for short**)** is a graduate of Mechanical Engineering from the University of Ibadan, Nigeria. With a fascination for numbers (and all things *Math*) that borders on awe, he has a natural predisposition toward anything which – even in the least – attempts to conceptualize, visualize and annotate numbers and the relationships between quantities. His flair for programming is a natural consequence of this.

The first wave of inspiration that birthed the **SHELVA-project** hit during his **NYSC** (National Youth Service Corps) assignment within which he served as a Mathematics and Further Mathematics teacher at **Federal Government college, Ogbomosho, Nigeria (2021-2022)**. At some point, due an overall readjustment of the school’s timetable structure, he got cornered into a position where he had to teach two different arms at the same time; something he found challenging to work around. As someone with programming skills, he took it as a challenge. His flow of thought went from *“isn’t there a quick and efficient way to handle a problem like this?”* to *“there must be an efficient way to handle a problem like this”* and finally to *“I just might know a bunch of efficient ways to handle a problem like this”.*

**SHELVA** thus became a thing.

Reach Olu’tomi via:

|  |  |
| --- | --- |
| Email | [oluwatomilayoowoeye.oio@gmail.com](mailto:oluwatomilayoowoeye.oio@gmail.com) |
| Phone | 09012481174, 08106754829 |
| LinkedIn | https://www.linkedin.com/in/olu-tomi-ini-owoeye- |

# ACKNOWLEDGEMENTS

I owe a debt of gratitude to Federal Government College, Ogbomosho. Teaching in this school gave me all the grounding and perspective I needed to pull off this project.

For their support and attention, I will always be especially thankful to:

* Mr. A.J. Ajagbe (The Director/Principal, Federal Government college, Ogbomosho, 2022)
* Mr. A.O Arowolo (The Vice-principal, Federal Government College, Ogbomosho, 2022)
* Mr. H.G. Ogunbiyi and Mr. O.M. Ajuwon (Heads of the timetable committee, Federal Government College, 2022)
* Mr. E.O. Osasona (The HOD, Mathematics and ICT, Federal Government College, Ogbomosho, 2022)
* Mrs. O.B. Alabi (The HOS, Mathematics, Federal Government College, Ogbomosho, 2022)

I am so thankful for all the data and files and loads of information I was provided, I wanted for nothing in that regard.

You have my deepest gratitude. Thanks a million, from the depth of my heart.

I had a lot of support from family, especially from:

* Mr. and Mrs. B. Ilelaboye. (My brother-in-law and dear sister)
* Mr. and Mrs. Owoeye (Dad and Mum)

This project took *a whole lot* out of me. Had I not received such warmth from these loving people, I probably would have bailed out on the idea. At times, it almost seemed as though they cared more about the project than I did. I’m grateful.

In a hundred years, I *still* would be grateful.

**And of course, I could never forget…**

I am thankful for the very existence of open-source code projects (especially Python) and its contributors worldwide. *What would we ever do without you people*?

Contents

[BACKGROUND: THE AUTHOR/CREATOR AND THE PROJECT 1](#_Toc130239745)

[ACKNOWLEDGEMENTS 2](#_Toc130239746)

[ABSTRACT: THE TIMETABLE PROBLEM 4](#_Toc130239747)

[DEFINITION OF TERMS AS USED IN THIS APP 5](#_Toc130239748)

[THE APP IN THE RUNNING 8](#_Toc130239749)

[MODEL HIERARCHIES 8](#_Toc130239750)

[The Class-hierarchy 8](#_Toc130239751)

[The Course hierarchy 8](#_Toc130239752)

[DYNAMICS OF THE TEACHER MODEL 10](#_Toc130239753)

[PERIOD GENERATION 12](#_Toc130239754)

[ARRANGEMENT AND PACKETING 15](#_Toc130239755)

[THE ATPG SCALE 15](#_Toc130239756)

[CHUNK VALUES AND THE PACKETING OPERATION 17](#_Toc130239757)

[THE SORTING OPERATION AND SORTING ALGORITHMS 19](#_Toc130239758)

[THE SORTING OPERATION 19](#_Toc130239759)

[THE SORTING ALGORITHMS 19](#_Toc130239760)

[POST-PROCESSING (GENERATION OF REPORTS) 23](#_Toc130239761)

[DETAILS AND DIAGNOSTICS 23](#_Toc130239762)

# ABSTRACT: THE TIMETABLE PROBLEM

The rigor of fashioning a timetable for a large school with so many areas of compartmentalization could be, among other things, overwhelming. Teachers usually take on more than one subject across more than one class and are required to fulfil the weekly quota (frequency) of each subject they take for each of the classes in which they take it. The timetable for such a system is thus constructed such that for all the periods available for each arm of each class for a single day, each teacher’s teaching periods do not coincide or overlap for every subject and every class that they teach, thus ensuring that they be not required to be in more than one class at a time, which (if that were to be the case) would be grossly impractical, if not impossible.

Setting up the timetable, more often than not, becomes *very* complicated *very* quickly as other requirements (subjects occupying consecutive periods being one of many) often get thrown into the mix. The larger the school, the larger the teaching staff around which to pass subjects, or the larger the number of subjects, the more complicated the entire system tends to prove, especially when adjustments are required on the fly. The launch of such a task presents itself as hectic, and would require a lot of permutation.

This project solves the timetable problem by creating a software package that abstracts away the hassle of fashioning out a timetable, i.e. sorting the teachers and their classes into periods spread across each day for each day spread across each week such that:

1. no two (or more) teachers are assigned the same period for the same set of students on the same day. Except where two or more subjects/courses run in parallel for the same class arm
2. no two (or more) exact periods in different class arms are assigned to the same teacher.
3. every subject’s weekly period quota (frequencies) are met for each of its offering class arms.
4. students’ convenience, subject structure and teachers’ expertise are put squarely into perspective.

Many more requirements that take into consideration the efficiency of teaching (for the teacher) and learning optimization (for the students) have been factored right into the algorithm driving the code.

# DEFINITION OF TERMS AS USED IN THIS APP

The purpose of this section is to get the reader acquainted with the terms and processes involved in the use of this app. Indeed, these processes are not complicated, they just might require a little bit of getting used to, as some of the terms (with which we most likely have already been familiar) have been redefined. The redefined terms have undergone no radical change in definition, rest assured, they just have been tweaked in a few places for the purpose of the app. For the most part, they retain their conventional, outside-this-app meanings to a significant degree.

There exists the likelihood that some of the terms used in this application might be a little confusing to new users. These terms are thus defined for good measure:

1. **Day (as in School day).**

Day, as used in this app, does not necessarily refer to its dawn-till-dusk meaning. It simply refers to the time frame (usually a day, hence the name) within which all the periods for a certain class feature. It spans the first and last periods of school for an (actual) entire day in real-life. For a typical school in South-Western Nigeria (where this app was initially written), the school day starts by 8:00 a.m. and runs till 2:00 p.m. or thereabouts. unless there are arrangements for afternoon classes. It is the first period all the way till the last.

1. **Department:**

The Department, much like its real-life meaning, is a container for the subjects provided by a school to its students. It is a catalogue for organizing subjects which share some level of decided-upon similarity.

1. **Subject/Course**

Subject (also, Course), as with "Day" above, may not always refer to the conventional meaning of the word in an academic setting, though it does, more often than not. A subject (course), with regard to the app, is any engagement which occupies one or more periods of a given class arm. Subjects occupy whole number values of periods e.g. Music in the first period or Math in the first two periods. Two subjects can only exist for the same period(s) if they go concurrently and both run through till the end of the period, e.g. Yoruba/Igbo languages occupy the last period. Subjects do not always have to be academic; break (recess) or extracurricular activities also count as subjects inasmuch as they fall within "Day". A subject is termed “Academic” if it is handled by a teacher. Non-academic periods are termed "Special periods" or more often, “Non-acads”.

1. **In-parallel Subjects/Courses:**

In-parallel subjects are academic subjects which contain some constituent subjects that always run side-by-side for the same period, for the same offering class arm. E.g. French/Spanish, Chemistry/Literature-in-English.

Noteworthy, however, is the fact that each constituent subject has a different set of teachers which handle them. In the app, when said subjects are displayed, they are suffixed with ‘-P’, thus referring to the fact that it contains child subjects running in “Parallel”. Even each of those constituents are suffixed by a ‘-P’ when displayed.

1. **Period:**

A period is the set duration within which a subject occurs. How long it takes is institution-specified. It is the very measure by which "Day" is calibrated. Academic periods tend to have the same period duration, whereas, non-academic periods (e.g. Break/Recess) might take varying lengths.

1. **School Class Category:**

Also termed "class category" or "class group", this is simply a collection of all the classes across different levels or grades with a common property for ease of reference. This in real life is what we would call "Senior School", or "Junior High" in a school situation especially where more than one school class category might exist. For instance, the typical Nigerian secondary school has two class categories: Junior secondary school and Senior secondary school.

1. **School Class (or simply, Class):**

The school class is a combination of all the arms that share one level of the academic hierarchy in the school setting. In the typical Nigerian school system, its arms are prefixed with the its name and alphabetized, e.g. JSS 1A, SSS 2B. This application is able to render class arms in said format. Also, each arm of the class in question can be rendered numerically, e.g. JSS 1\_1, SSS2\_2.

1. **School Class Arm:**

Sometimes, plainly referred to as "Arm", the school class arm is a sub-section of the school class, which in turn is a subset of the school class category. The arm is the compartment into which the school class divided, primarily to accommodate a number of students that exceeds the standard number of students studying together in the same class room and also to allow for different specializations between students of the same class, e.g. Sciences and The Arts. The school class arm is the literal classroom in which schoolwork is done.

1. **Teacher**:

Fairly self-explanatory. A teacher handles the subject for a class at a given "Period" of the "Day". It is quite possible for a teacher to take on more than one subject, across more than one class arm.

1. **Teacher’s Sequence**

The teacher’s sequence is the list of class arms for which the teacher is responsible for a certain subject. It could also refer to the list of class arms a teacher is required to teach on a given day.

1. **Chunk (or chunk value):**

A subject’s chunk value refers to the number of times a subject features in the periods of a particular class arm on a certain day. Subjects that feature more than once often do so contiguously (except where certain other circumstances exist). E.g. class arm JSS1A has Mathematics for two periods on Monday, we would say “the chunk value for Mathematics for JSS1A on Monday is 2”.

1. **Frequency (Weekly):**

Also referred to as subject/course quota. This is the total number of periods a subject must occupy for the total span of the school week by the user’s design; it is almost always a multiple of the subject’s chunk value.

1. **Packeting:**Packeting refers to the allocation of the different subjects offered by a class arm into different days of the week for which the arm is required to take classes.
2. **Sorting:**Sorting is the major work of this project. The sorting operation ensures that for a given day, every teacher which features in it (regardless of the class arms they teach) has their periods spread in such a way that no overlaps occur. As much as possible, the teacher’s periods are arranged such that (except for intentionally consecutive periods) there is some spacing between the periods to help them rest and freshen up for their next lesson.

# THE APP IN THE RUNNING

## MODEL HIERARCHIES

This application models the operational structure of a full-fledged (typical Nigerian) school. However, this model structure is in no way limited to schools with such an operational structure. It can be modified to suit institutions whose structures are not as involved. As earlier stated, this application models both the school's operational structure and the (likely) interactions between them. These models present themselves in hierarchical formats.

There are two models, conveniently termed The Class-hierarchy and The Course-hierarchy. At its core, the timetable, in this context, is little more than a black and white representation of the relationship between subjects (courses) and classrooms at certain parts of the day, both collectively (i.e. in whatever groups these might have been sorted into) and individually. Both models would have to be created.

### The Class-hierarchy

The class hierarchy represents the superset-set-subset relationship of the between the items in the class hierarchy.

The hierarchy can be simply represented as:

Class categories (groups) contain all the classes which in turn contain all the arms for said classes. The class category in the real-life school setting is what we would call 'Junior School' or 'Senior High' or 'Middle School' (US). It is merely a convenient way to categorize classes, especially in a setting that has more than one class category running.

## The Course hierarchy

The Course hierarchy, just like the class hierarchy above, is a superset-set-subset relationship that describes the key components of the structure that handles and executes teaching, course structure and the various instances of specializations of class arms within a class, e.g. class arms designated for The Arts or Sciences or other.

The teachers primarily handle the teaching and what they teach is the subject or course. The department is merely the container that houses like subjects, a neat way of grouping subjects in the event that similar-structured subjects are taken as a whole. However, for ease of design (as regards the efficiency of coding), among other things, the subject is said to "have" a teacher and "hail from" a department.

It should be plainly stated that there are certain subjects (Non-acads, as they are referred to here, as discussed in the definition of terms section) which do not require handling by a teacher. These are subjects like break-time (recess) or extracurricular activities. (The definition of the term “subject” used here and throughout the manual is consistent with the definition – or redefinition, if you will – in the [*Definition of terms*](#_THE_MEANING_OF) section).

The Course hierarchy for an academic subject goes:

In-parallel subjects

An In-Parallel subject, however, is a little more intricate. Its structure looks like this:

A Non-acad (Non-academic subject/course), however,stands alone, without a department and nor teachers

## DYNAMICS OF THE TEACHER MODEL

Apparently, the Teacher model (as is in the app), is an "in binary" rendering of the teacher, who, much more often than not, is an actual human. There exist some intricacies with regard to the teacher model that can neither be ignored nor compromised; not even a little.

Some immediately come to mind.

1. **No period overlap**

The teacher’s periods for a particular day must not overlap. This is a major constraint in the setup of any timetable. Although, teachers (might) teach across different class arms, probably across different classes, for the day in which they teach, there should be no intersection (or clash) whatsoever between periods in the teacher's sequence. Otherwise, that would imply that the teacher is required to be in more than one class arm at a time.

1. **Part-time members of staff**

The presence of part-time staff connotes that the teachers who fall under this category may only be required to teach on certain days of the week, but not every day.

This is especially true for the National Youth Service Corps Members (A Nigerian one-year assignment scheme for tertiary institution graduates, especially for those under thirty) who during their term of service often teach in schools. They are required by the scheme to gather on a certain day of the week to effect their community-development strategies, and as such do not attend school on that day.

Indeed, this is circumstantial, nevertheless, the possibility that teachers would, for some legitimate reason, be consistently absent on a certain day or days of the week is totally plausible.

1. **Spacing**

Teachers/Handlers/Tutors rarely dispense their duties efficiently and energetically if they always have to shuffle from class arm to class arm with little or no breaks in between. The software handles this concern by spreading each teacher’s periods for a single day in such a way that they (the Teacher) have room in between the periods to rest and prepare, except in the case where they really cannot get any rest period because they teach enough periods to span the entire day.

This application puts these factors squarely into perspective and would fulfill these requirements for every teacher as much as it can, even if recalibrations and readjustments would have to be carried out.

As a bonus, the app even calculates the minimum number of teachers that are required to take a subject (or subjects) such that the number of their teaching periods do not exceed the number of periods available for teaching for that particular day.

## PERIOD GENERATION

Unlike the registration of departments, days and class names, periods are not registered by hand. Rather, they are generated for as many class arms have been selected (in the user interface) for as many days have been selected (also in the user interface). Note that at this point, all the Non-academic periods needed must have already been created. Periods are generated in three different ways:

1. **Given duration**

This method generates periods for the already selected arms and days based on:

* When the day is supposed to start, e.g. 8:00 a.m.
* The duration of each academic period in **hh:mm:ss** format (e.g. 00:45:00 for 45 minutes).

Note that all academic periods (periods reserved for academic subjects) have the same duration. Only Non-academic periods (i.e. the periods associated with non-academic subjects) can take varying lengths.

* Duration of each non-academic period and its position(s).

Positions are comma-separated whole numbers. E.g. A position if 4,7 means that the Non-academic period is present in two places during the day, at both the 4th and 7th periods respectively.

* Frequency of the academic periods; i.e. how many academic periods the user wants
* Inter-period interval.

This is how much time there should be in between periods, in **hh:mm:ss** format. Enter 00:00:00 if there are no time intervals between periods.

1. **Given day-start, day-end boundaries for Acads.**

Peradventure the user has a total duration for all of the academic periods without factoring the period intervals or non-academic periods.

A plausible example would be “We want the total amount of *academic* work to take as long as 8:00 a.m. to 1:00 p.m. only exceeding 1:00 p.m. when intervals and extra-curricular (Non-academic) activities are added.”

This method generates periods based on:

* When the day starts
* The limit (total duration, without intervals and non-academic periods considered; the “1:00 p.m.” in the above example)
* Duration of each non-academic period and its position(s). Same as in point 1 above.
* Frequency of the academic periods; i.e. how many academic periods the user wants
* Inter-period interval.

This is how much time there should be in between periods, in **hh:mm:ss** format. Enter 00:00:00 if there are no time intervals between periods.

1. **Given day start, end boundaries as absolute constraints.**

This method generates all the periods, non-academic periods (with all the intervals) and fits everything between two particular time inputs (the day-start and the day-end), without spilling over the time constraints.

An example would be: “We want to generate periods such that we have 10 academic periods, one non-academic period for break time 45 minutes long at position 4 (the fourth period), an interval of 2 minutes 30 seconds between each period, and all of *these* are to fall between 8:00 a.m. and 4:30 p.m., no extensions!”

This method generates periods based on:

* The absolute start time for the day (Abs. start)
* The absolute end time for the day (Abs. end)
* Duration of each non-academic period and its position(s). Same as in point 1 above.
* Frequency of the academic periods; i.e. how many academic periods the user wants
* Inter-period interval

**Noteworthy:**

All time input (intervals, durations, start and end times), regardless of the generation method employed, is of the format hh:mm:ss (24-hr format)

## ARRANGEMENT AND PACKETING

### THE ATPG SCALE

*An objective way to arrange subjects based on their structural attributes*

ATPG stands for **Arithmetic, Theoretical, Practical, Grammatical**. This is a rating system that evaluates subjects based on four structural attributes by assigning said attributes numerical weights. These four structural qualities are: arithmetic, theoretical, practical, grammatical by default, even though these can be changed if the overall description of the structure of the user’s registered courses deviate from this. The weights that can be assigned range from 1 to 7, and the ratings assigned to said weights range from 1 to 10.

The mathematical weighting system used, borrows into the idea that every positive integer (with the exception of 1) is a product of primes.

The weighting system runs such that if the weight associated with any attribute (i.e. any ) one-one maps to a prime number , such that, :

Where:

prime number weight of the “Analytical” property of the subject/course

prime number weight of the “Theoretical” property of the subject/course

prime number weight of the “Practical” property of the subject/course

prime number weight of the “Grammatical” property of the subject/course

rating of the “Analytical” property of the subject by the user from the user interface.

rating of the “Theoretical” property of the subject by the user from the user interface.

rating of the “Practical” property of the subject by the user from the user interface.

rating of the “Grammatical” property of the subject by the user from the user interface.

a scaling constant.

The subjects are then sorted according to their ATPG values in descending order. This sort is maintained throughout except in cases where an overlap would occur in a teacher’s periods.

**Noteworthy:**

The score gotten by a subject in the ATPG analysis is **in no way** a measure of the importance of said subject in reality, whether singly or with respect to the others. The ATPG analysis merely yields a numerical value with which to arrange and position subjects. Academic subjects, that is.

Non-academic (special) subjects have no ATPG value.

The values (not the weights) of the values are set from a group of sliders on the app interface. However, if the user deems it preferable, other attribute names and its corresponding weight can be set in place of the arithmetic, theoretical, practical and grammatical attributes.

### CHUNK VALUES AND THE PACKETING OPERATION

#### THE CHUNK VALUE

To begin with, every subject has its weekly frequency for every class arm that takes it; for instance, Drama could take up 4 periods weekly for JSS 1A. Moreover, every time it features (on the day in which it features), it could take up two consecutive periods at a time (double-period). Thus, it only features on 2 days of the week, for 2 consecutive periods on each of these days. The number of (consecutive) periods the subject takes for each feature is called its *chunk value*, or simply, its *chunk.* Thus a double-period simply means a chunk value of 2. Single period means a chunk value of 1.

The chunk value of each subject is specified by the user right from the user interface.

#### PACKETING

Packeting refers to the process of distributing out subjects into different days of the week for a particular class arm. This “distribution” is based on the chunk value of said subject. For instance, as in the scenario of *Drama* earlier stated, Drama would be assigned to say, Monday (twice, as its chunk value is 2) and to another day, say Thursday, for its remaining two periods, since (as earlier stated) its weekly frequency for JSS 1A is 4.

Also, the Packeting process takes into account the maximum number of periods allowed for a class arm during the day(s) in question.

##### PACKETING PATTERNS

Subjects, for each class arm, are not “packeted” into days of the week willy-nilly. The subjects are imputed into days of the week based on an array of mathematical patterns. These patterns make up what is referred to here as the *Packeting algorithms* or *Packeting patterns*. They are as follows:

1. **The *Direct* Pattern**

The *direct* pattern distributes subjects into every successive day of the week, starting from the beginning of the week. As with every other pattern that will be mentioned here, subjects are distributed according their chunk values. Only when there isn’t enough room to accommodate the chunk value of a subject will it seek vacancy elsewhere, i.e. on another day of the week.

1. **The *Rev-direct* Pattern**

Identical to the direct pattern it its operation, the only difference is that the distribution of subjects into weekdays based on their chunk values is done in reverse (i.e. from the end of the week), hence the prefix “*Rev-*”.

1. **The *Leapfrog* Pattern**

The leapfrog pattern distributes subjects into days of the week based on their chunk values in a pattern that follows the arrangement of odd (or even) numbers on the real number line. This means that subjects are distributed into days which do not lie side-by-side, but have a one-day interval between them, e.g. subjects are packeted into day1 and day3 and day5.

Only when these odd-number days have been traversed (and there still are subjects yet to be packeted) are the even-number days considered for the packeting process. The one-day interval between the days could be thought of as “leaping over” (as in the leapfrog game) one day on to the next. That explains the name *Leapfrog* thus given to the pattern.

1. **The *Rev-leapfrog* Pattern**

Identical to the Leapfrog pattern in its operation, it only works in reverse, i.e. from the end of the week before spreading to the start of the week and then (if occasion calls for it) over the even-number days of the week (also in reverse). In every other regard it is completely similar.

1. **The *xlx-reflection* pattern**

This pattern, in its layout is a little more involved than the other patterns before it. This pattern distributes subjects into the first day, then the last, then the second day, then the second last, and so on till the entire week has been traversed. Mathematically, if the length of the week were denoted and any first day was taken, the next chosen day would be the )th day, the prefix “xlx-” is as a result of the dynamic between .

6. **The *rev-xlx-reflection* pattern**

As with all the other rev-prefixed patterns, this pattern is also identical to the xlx-reflection pattern, but works in reverse. This pattern distributes subjects into the last day, then the first, the second last, the second and so on, till the entire week has been traversed.

The Packeting patterns all work together automatically underneath the hood across all of the offering class arms, unlike the sorting algorithms in the subsequent section, the user will not have to specify an algorithm to run.

## THE SORTING OPERATION AND SORTING ALGORITHMS

### THE SORTING OPERATION

The sorting operation is really the “big-boy” function of this software package. It handles all the calculation and permutation required to set subjects into the periods of their offering class arms whilst making sure that the teacher of said subject does not get cornered into having two of his (gender identity implied) periods coincide with the other. This task seems simple enough if the teacher only taught one subject to only one class arm, however, this is rarely the case (if ever). The sorting operation is so powerful that it handles all the complexities that come with that task; however wild.

As with the Packeting operation, the sorting operation is completely math-methodical and does not in any way shuffle subjects randomly. In its ability to yield results, it is completely deterministic. This attribute was set by its developer on purpose to ensure consistency of results when parameters are identical.

The sorting operation works in concert with some algorithms specially designed for the sorting process.

### THE SORTING ALGORITHMS

The sorting algorithms are a group of mathematical patterns that have been designed with the aim of arranging subjects for a particular day in such a way that none of the teachers of said subjects get cornered. Due to its mathematical nature, it handles the case for every , i.e. it is capable of “expanding” to any level (however complex) without compromising its sorting integrity. The sorting algorithms direct the sorting process.

These algorithms run for every teacher that teaches for every day of the week. First, the teacher’s subjects are sorted by their ATPG values, such that subjects with similar structures tend to follow similar patterns as given by the algorithm. If the initial standard imposed by the algorithm cannot be met without the teacher getting cornered, the algorithm compromises and checks through all of the possible combinations of positions that can exist in turn without getting the teacher cornered. If it finds one, it sticks with it and sorts the teacher’s subjects accordingly. If it cannot, it bails, registers the teacher as “displaced” and stores the teacher (alongside his details, i.e. class arm, subject and the day of the week etc.) in a data structure that will later be called upon later on.

Although each of these algorithms offer a unique flavor of direction to the sorting process, they share some degree of similarity with each other and even with the Packeting algorithms. The algorithms are:

1. **The leapfrog algorithm**

The leapfrog algorithm takes the teacher’s sequence for a day and spreads his teaching periods in such a way that (except when absolutely impossible) between each of his teaching sessions (periods) for a class arm, there is at least one period of space for rest and recuperation. Ergo there seems to be a skip (or *leap*) over one period from one of his teaching sessions to the next. This entire arrangement begins from the first period until the last period of the day.

1. **The reverse-leapfrog algorithm**

Identical to the leapfrog algorithm discussed above. The only difference is that its arrangement takes the reverse direction, i.e. from the end last period to the first.

**Real-world implication for the leapfrog and reverse-leapfrog algorithms**

Any sorting that follows this pattern would have subjects with the highest ATPG ratings (and as such, might share similar properties) congregate at the first periods of the day (or at the end, if it is the reverse-leapfrog algorithm).

**A fairly realistic example:**

If, for example, for a particular class arm, all the science-leaning subjects (say Math, Chemistry, Physics, etc.) were to take the earliest periods of the day, when the environment is still cool and the sun isn’t burning hot yet, because only few people can care enough to deal with ’s and ’s when the sun is blazing hot.

1. **The center-cluster algorithm**

The leapfrog algorithm takes the teacher’s sequence for a particular day and spreads his teaching periods such that (unless absolutely impossible) his subjects (especially those similar in structure) congregate at the middle periods of the day, before spreading to an earlier (before the middle) period, then a latter (after the middle) period, then a much earlier period, the a much latter one and so on, till all the periods are filled (if the teacher teaches that much). Spaces are still kept wherever possible for the teacher to rest and recuperate.

1. **The reverse-center-cluster algorithm**

Identical in operation to the center-cluster algorithm defined above. The only difference (which is fairly obvious at this point) is that works in reverse. That means, it congregates like subjects at then middle periods, then spreads to a latter (after the middle) period, then to an earlier (before the middle) period, to a much latter period, to a much earlier period, and so on, till all the periods are filled. paces are still kept wherever possible for the teacher to rest and recuperate.

**Real-world implication for the center-cluster and the reverse center-cluster algorithms**

In a real-world scenario, this algorithm gives preference to the middle periods of the day for any one class arm. Subjects/courses that are similar in structure (and thus have close ATPG values) will tend to be congregated at the middle of the day, before the starting or ending periods of the day. This would be helpful in scenarios where the middle of the day is considered the epochal point of the day.

**A fairly realistic example:**

“For a particular class arm, we set our Arts-leaning subjects to have the highest ATPG values, and we would very much like for them to hold (as much as possible) at the middle of the day, you know, not quite beginning, and not quite the end.”

1. **The xlx-reflection algorithm**

The leapfrog algorithm takes the teacher’s sequence for a particular day and spreads his teaching periods such that (unless absolutely impossible) his subjects (especially those similar in structure) take both ends of the day (the starting period first, and then the last period), before spreading inward toward the middle. This is similar in spirit to the xlx-reflection pattern for packeting subjects into weekdays (as earlier discussed), their names were also derived in a similar fashion. Also, as with the other algorithms, spacing between periods is included (except when the entire periods have been occupied, and no spaces can be afforded).

1. **The reverse xlx-reflection algorithm**

This is identical to the xlx-reflection algorithm in its operation only that it works in reverse (as you will most certainly have guessed). Periods begin to fill from both ends (first the last period, then the first period), before spreading inward toward the middle. Just like the xlx-reflection algorithm, spacing is implied.

**Real-world implication of the xlx-reflection and reverse xlx-reflection algorithms**

This algorithm sees to it that similar-structured subjects (having similar or closr ATPG values) congregate at both ends of the day before spreading to the middle, all things being equal. This might be preferable in circumstances where the students can get mentally disengaged if subjects with similar structures are taught back-to-back. It makes sure students only take said courses at the start of the day and only come back to similar-structured subjects at the end of the day.

**A fairly realistic example:**

“For *this* class arm that has both double periods of Math and double periods of Further Mathematics today, why don’t we have Mathematics in the morning and push Further Mathematics till afternoon, lest the students get bored and disengaged due to four periods of back-to-back calculations?”

## POST-PROCESSING (GENERATION OF REPORTS)

### DETAILS AND DIAGNOSTICS

The application has a details and diagnostics section which seeks to make sense of the entire timetable structure. It spots the dynamics between models (Departments, teachers, subjects, class categories, classes, class arms and days of the week) and their intricacies.

The details section provides needful information like the number of class arms generated, number of teachers generated, number of classes registered, number of class categories registered, number of departments, number of subjects registered, etc.