**Marshall Course Scheduling Optimization: Interim 1**

**Group 4**:

Srivijay Chaparala

Yu (Phoebe) Guo

Alberto Longoria

Keying (Claire) Que

Jiatong Zhong

**Introduction**

The following report presents one metric to evaluate the “goodness” of the course scheduling process at Marshall in phase I. We identify why this metric is needed, describe computation process for the metric, go through an analysis of the results based on the metric in the current process, justify the appropriateness of the metric, as well as outline further steps to optimize the scheduling process using the metric.

**Definition of the Metric**

In the current course scheduling process, 60% of the courses are scheduled during Phase I; this is where department coordinators place courses based on initial room allocations provided by Shannon and her team. Later in the process, some of the scheduled courses might be moved or modified in order to satisfy the remaining unscheduled courses. This second correction implies that there is room for improvement in the initial allocation; more courses could be arranged with greater insight during Phase I as the final scheduling results. Therefore, we have decided to create the Allocation Match Index (AMI) as a metric to measure how well actual course schedules matches with the slots given in the initial department allocation. AMI is calculated as the number of hours actually scheduled by each allocated department in the final schedule, divided by the total number of hours initially assigned to departments or programs (excludes unassigned slots in allocation step) in a given term. To achieve what is mentioned to compute AMI in Python/R, we follow the steps below:

* Data source
  1. Historic allocation data - *Department\_Allocations\_20171.xlsx*
  2. Historic scheduled classes - *Marshall\_Course\_Enrollment\_1516\_1617.xlsx*
* Data cleaning
  1. Filter the enrollment information by same term (20171) with allocation data
  2. Extract *room*, *days*, *department*, *start time* and *end time* from both files into two separate tables
  3. Recode the room numbers and department abbreviation so that they match in the two tables
  4. Remove courses with missing start / end time
* Convert start and end time to a numeric vector
  1. Split the time by space, put time in column 1, and put AM/PM to column 2
  2. Further split column 1 into hours, minutes and seconds by “:”
  3. Time = hours + minutes / 60
  4. Use AM/PM info in column 2 to further convert time in to 0-24 hours a day
* Separate time by days
  1. For the slots that are in multiple days, we need to separate them by each individual day. First, generate variable corresponds to Mon through Sun.
  2. For each slot, if it’s indicated by the Days variable to have slot in a particular day, then assign a value to the variable of that day
  3. Use the melt function to melt the table by the Mon through Sun. For example, room ACC201 has slot on Mon & Wed from 8-9:50 AM, so the Mon & Wed variable would have value ‘M’ and ‘W’ respectively and then melted accordingly.

|  |  |  |  |
| --- | --- | --- | --- |
| Room | Days | Start Time | End Time |
| ACC201 | MW | 8:00 AM | 9:50 AM |

↓

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Room | Days | Start Time | End Time | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
| ACC201 | MW | 8:00 AM | 9:50 AM | M |  | W |  |  |  |  |

↓

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Room | Days | Start Time | End Time | Day |
| ACC201 | MW | 8:00 AM | 9:50 AM | M |
| ACC201 | MW | 8:00 AM | 9:50 AM | W |

* 1. Remove rows with missing value after melt
* Calculate match between allocation and final schedule
  1. Construct slot ID by paste room, day and department together
  2. For each ID that match in both tables, calculated the intersected hours in start / end time intervals
  3. Sum up the intersect hours
  4. Divide the number in step 3 by total hours assigned during allocation

**Analysis result**

Initially, over 1700 hours of slots from 38 classrooms were assign to 13 departments / programs. However, in the final scheduled the classes, only 730 hours actually matched their original assignments. The AMI is only around 43% and required additional human labor to re-assign the rest of the slots.

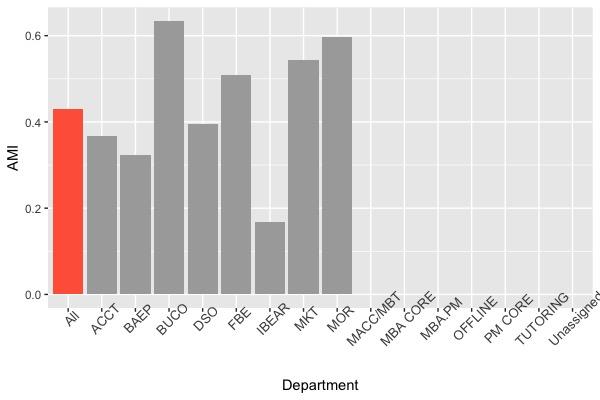


Figure 1. Allocation Match Index by department

**Justification (CASE)**

We believe the Allocation Match Index (AMI) to be an appropriate metric for the following reasons:

* It is computable: the metric can be calculated from the available data (historic allocation data, historic scheduled classes) after cleaning and modifying it, as described above. We have done so successfully and arrived at the result of 43% for AMI.
* It is actionable. Currently, AMI is only at 43%. With more insight through data analytics, the scheduling teams will be able to improve allocation through optimization methods that incorporate more relevant information and priorities. This will lead to a better Phase I schedule which more properly accommodates both continuing classes and new classes, and foresees scheduling issues that would otherwise be changed ad-hoc in Phase II. Through appropriate, strategic action, AMI may be certainly be increased.
* It is simple to understand and interpret. AMI, simply put, measures how similar the final and original schedule allocation are. Interpreting the result is simple as well: a higher AMI means that the schedule allocation is a better one, which requires less corrections in Phase II.
* Finally, the metric is also enlightening. It has an extremely close relationship with our measure of goodness, which is how close the Phase I allocation is to the final allocation. In terms of AMI (number hours actually scheduled by each department divided by number of hours initially assigned to departments or programs), the closer this value is to 1.0, the closer Phase I allocation is to the final one.

**Opportunities for improvement**

The key problems with current the class scheduling system could be summarized as follows:

1. Based upon history data, only 43% of the courses could be successfully scheduled in Phase I. However, there is no guidance of which class should be scheduled first and which should be handled after first round of trial.
2. There is not much flexibility for changes, such as temporary unavailability of buildings or increasing number of students. These changes could lead to much headache and scrambling. Similarly, there is no flexibility for administrators to control scheduling globally.
3. Pure human-based resolution of conflicts results in late scheduling and last minute scrambling. Human is good at logic deduction but not processing large scale of data.
4. Scheduling Phase II is significantly harder than Phase I because slots are already taken.

To address each of the issues, we propose the following improvements that is able to resolve most of the above problems:

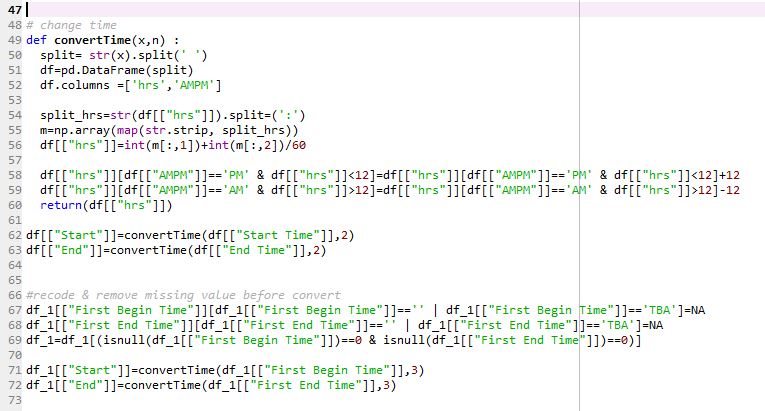
1. An automated system with a ranking for each class and instructor to guide the scheduling system to schedule classes with higher priorities. For instance, a mandatory class with big student count will be scheduled ahead of elective with less than 10 students. Based upon this, a fuzzier scheduling of classes helps to increase success rate. For example, scheduling a class two hours earlier or later than its history schedule will increase the chance of settling this class.
2. Introduce reserved time slots in which no class could be scheduled. This is to create tolerance for expected expansion of overall class scale. Also, we would like to consider introducing a coefficient to indicate the increase of students. For instance, if we expect a 10% increase of students and classes next semester compared against history average, we would like to allocate 1.1 times more classrooms for popular courses, this 1.1 number could be altered by administration people reflect a real expectation
3. Issue point 3 and 4 could be addressed by using this methodology. After first round of class scheduling, which is Phase I, results will go to Shannon’s department for Phase II, which is pure manual. Actually after Phase I, each department could try resolving unscheduled classes internally based upon its unscheduled classes and available spots assigned to it. All this could be done in a python script. Furthermore, if a course cannot be scheduled internally to a department, we can always do the same schlepping using Python a level up. In the end, Python will do the most of the jobs in minutes and leave fewer work for human beings to communicate with professors to swap classes and accommodate changes. With proper ranking and evaluation of a scheduling results, Python could give advice what courses are mostly suggested to be rescheduled.

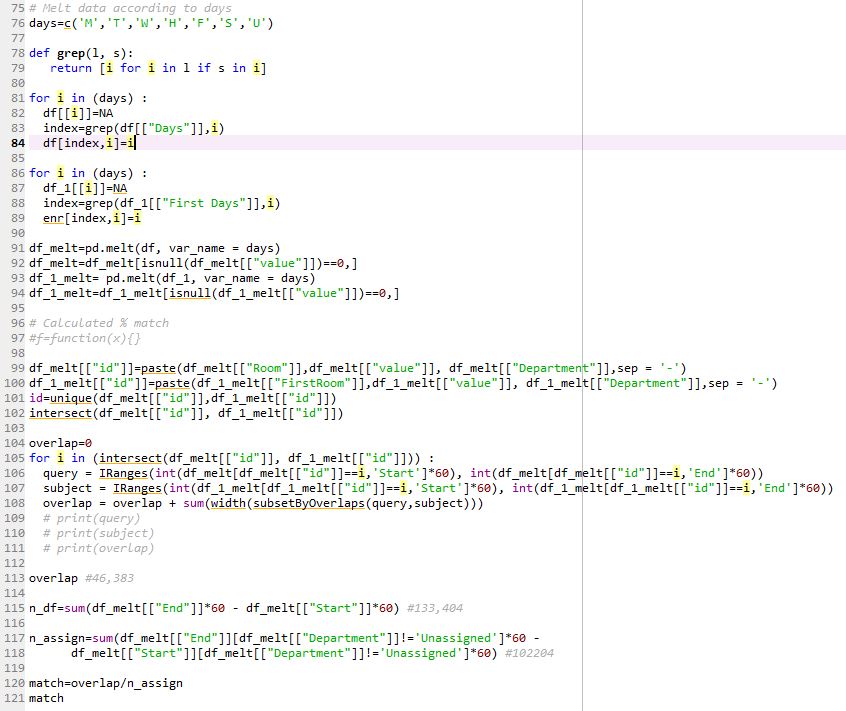
**Conclusion**

In summary, Allocation Match Index is an appropriate and efficient metric to measure goodness of the process. It is computable, actionable, simple, and enlightening. Implementing the methodologies such as creating class rankings and modify reserved time slots will help increase AMI, therefore, optimize the course scheduling process.

**Appendix:** Python code







R code

