CPSC 433: Artificial Intelligence – Project Problem Description

(Version 1)

We (you!) oversee the academic semester scheduling group at the University of Calgary. This is a rather challenging problem that's obviously even more complex than we will define here. We will simplify this problem to the challenge of assigning ***Lectures*** and ***Tutorials*** to weekly room time ***Slots***.

Instead of this being a year-long problem, or even a semester-long problem, the fundamental idea of our problem can be simplified to the challenge that we (as the university scheduling group) want to make a weekly schedule (re-usable for 13 weeks) for the semester for all our classes.

We will have already contacted all the interested faculties/departments/professors to ask them what time locations in a 5-day week MTWRF they'd like to have access to room slots the university has available. Those groups will have given us some (hard/soft) constraints that described how they want their lectures/tutorials to fit into the available room slots. (We are going to save Sat/Sun for Continuing Education)

We are going to help ourselves out further by treating our room availabilities ***Slots*** as having two types: (1) room slots that lectures can fit in (larger and designed for lecture delivery) and (2) rooms slots that tutorials can fit in (smaller and designed for tutorial delivery).

There are many more properties that rooms can have in real life, but the only other property we're going to consider is if a room is designed for 'active learning' (AL) [generally this means moveable layout and interactive technical systems]. Some lecture rooms and tutorials rooms available at certain times will have this AL state be either true or false. We'll address how to track AL later as a hard constraint.

Note, for each time a **Slot** represents for a week we have different quantities of rooms available. We aren't going to bother with variations in rooms sizes for lectures/tutorials and will consider all lectures to be equal in size and similar for all tutorials.

(*There are a lot of things both you and I can think of to make this problem even more challenging or simply different that we will be ignoring!)*

General Problem Description

There are many departmental programs at the university. For example, Computer Science, Data Science, and Software Engineering.

Each program offers numerous courses under course codes.

Each semester a department will offer one ore more lecture sections for these courses.

Each program wants to book time slots for their weekly lecture offerings. For now, to simplify things instead of worrying about all 3 components for this description (since each triple of 3 is uniquely needing to be scheduled), we will reduce each down to one lecture symbol l.

This reduces the problem down to the generic challenge of needed to schedule a set of m lectures $\boldsymbol{Lectures} = \{\boldsymbol{l_1}, \dots, \boldsymbol{l_m}\}$ to schedule, where one l_i is one lecture section that needs to be scheduled.

Tutorials are inconsistently named across campus but stand in for the idea of smaller teaching environments that the students from a lecture are divided into.

As a result, we also have a set $\boldsymbol{Tutorials} = \{\boldsymbol{t_{11}}, \dots, \boldsymbol{t_{1k_1}}, \dots, \boldsymbol{t_{m1}}, \dots, \boldsymbol{t_{mk_m}}\}$ of tutorials associated with lectures.

Tutorials $t_{i1}, \dots, t_{ik_i}$ are associated with lecture section i. Note, if $k_i = 0$ this would mean the lecture does not have any tutorials to be scheduled.

Finally, we have a set

$\boldsymbol{Slots} = \{\boldsymbol{s_1}, \dots, \boldsymbol{s_n}\}$ of time slots into which lectures and tutorials must be assigned. These slots will have at least a time start and end point.

Since you are a student at the UofC a comparison to creating a weekly schedule for your own weekly semester is fair (that's where this problem came from!). Lectures need to be booked to times in a week. Some lectures have one or more tutorials that also need to be booked to times in a week. There will be limitations on the available time periods to be booked and other limitations like limiting overlap between lectures/tutorials, etc. Like your class schedule this problem can be thought of as creating a weekly schedule that is re-used every week.

Now for a formalization of the constraints on the problem. We'll start with the most obvious example of hard constraints.

For each slot s_j we have a limit $\boldsymbol{lecturemax}(\boldsymbol{s_j})$ (a natural number) of lectures that can be put into the slot. This is because we have only that many lecture-sized rooms available at that time.

For each slot s_j we have a limit $\boldsymbol{tutorialmax}(\boldsymbol{s_j})$ (a natural number) of tutorials that can be put into the slot. (Lectures and tutorials are independent from each other in this regard, i.e., a slot can take $\boldsymbol{lecturemax}(\boldsymbol{s_j})$ lectures and $\boldsymbol{tutorialmax}(\boldsymbol{s_j})$ tutorials.).

$\boldsymbol{lecturemax}$ and $\boldsymbol{tutorialmax}$ are one example of a ***hard constraint*** for this problem that must be fulfilled for a valid solution to be found.

We will also handle Active Learning room needs in a similar way. Each slot will have a $\boldsymbol{ALlecturemax}$ and $\boldsymbol{ALtutorialmax}$ to indicate the maximum active learning rooms of each type available at that slot time. These will always be less than the general max values for each slot (i.e. represent the subset size of rooms for AL)

Another obvious ***hard constraint*** for this problem is that the tutorials for a particular lecture section can never be in the slot in which the lectures are so that students can register in that tutorial. (*On the other hand, we won't care if a tutorial for a lecture is booked immediately before/after lectures which simplifies things.*)

There will be more ***hard constraints*** listed later as well as ***soft constraints*** which can be left unfilled in a valid solution, but ***better*** solutions will satisfy them if possible.

The task of the system you will develop/implement is to find an assignment $\boldsymbol{assign}$ of lectures and tutorials to slots that fulfills the ***hard constraints*** and optimizes the ***soft constraints***.

More formally, $\boldsymbol{assign}$ is a function

$\boldsymbol{assign}: \boldsymbol{Lectures} + \boldsymbol{Tutorials} \rightarrow \boldsymbol{Slots}$ that fulfills two conditions, namely

1. $\boldsymbol{Constr}(\boldsymbol{assign}) = \text{true}$
   $\boldsymbol{Constr}$ is a function testing the fulfillment of all ***hard constraints*** (and being **true** if and only if every single ***hard constraint*** is fulfilled)

2. $\boldsymbol{Eval}(\boldsymbol{assign})$ is **minimized**
   $\boldsymbol{Eval}$ is an evaluation function that measures how well an assignment fulfills the $\boldsymbol{Eval}$ ***soft constraints***.

General Problem Constraints

First, lest us examine the general ***hard*** and ***soft*** constraints.

Hard constraints:

First 5 are restatements of prior constraints

1. Not more than $\boldsymbol{lecturemax(s)}$ lectures can be assigned to each $s \in \boldsymbol{Slots}$.
2. Not more than $\boldsymbol{tutorialmax(s)}$ tutorials can be assigned to each $s \in \boldsymbol{Slots}$.
3. Not more than $\boldsymbol{ALlecturemax(s)}$ lectures that need $\boldsymbol{activelearning}$ (described later) can be assigned to each $s \in \boldsymbol{Slots}$.
4. Not more than $\boldsymbol{ALtutorialmax(s)}$ tutorials that need $\boldsymbol{activelearning}$ (described later) can be assigned to each $s \in \boldsymbol{Slots}$.
5. $\boldsymbol{assign(l_i)} \neq \boldsymbol{assign(t_{ik_i})}$ for all i and k_i.

These are new ***hard constraints***

1. Some courses should not be scheduled at the same time. For example, first year students are directed to take course like CPSC 231 and CPSC 251 in the same semester. To inform the system of this, the input for your system will contain a list of $\boldsymbol{notcompatible(a,b)}$ statements, with $a, b \in \boldsymbol{Lectures + Tutorials}$.

   For each of those, $\boldsymbol{assign}(a) \neq \boldsymbol{assign}(b)$ This mean the a and b slots the lectures/tutorials are in can't overlap in time.

2. Sometimes there are certain offerings that already have pre-arrangements for certain slots. The input for your system can contain a partial assignment $\boldsymbol{partassign}$: $\boldsymbol{Lectures + Tutorials \rightarrow Slots + \{\$\}}$. ($ is a placeholder for no pre-assignment.)

   The assignment $\boldsymbol{assign}$ your system produces has to fulfill the condition: $\boldsymbol{assign}(a) = \boldsymbol{partassign}(a)$ for all $a \in$ $\boldsymbol{Lectures + Tutorials}$ with $\boldsymbol{partassign}(a) \neq \$$.

3. The input for your system can contain a list of $\boldsymbol{unwanted(a,s)}$ statements, with $a \in \boldsymbol{Lectures + Tutorials}$ and $s \in \boldsymbol{Slots}$.

   For each of those $\boldsymbol{assign}(a) \neq s$ must be true.

4. Sometimes there are certain offerings that should be delivery in an active learning environment. To inform the system of this, the input for your system will contain a list of $\boldsymbol{activelearning(a)}$ statements, with $a \in \boldsymbol{Lectures} + \boldsymbol{Tutorials}$.

   For each of those, $\boldsymbol{assign}(a)$ should be such that $\boldsymbol{ALlecturemax(a)}$ (or $\boldsymbol{ALtutorialmax(a)}$ as necessary) is greater than 0 (and the prior hard constraint for the slot AL availability is not exceeded).

5. There will be additional hard constraints specific to the University of Calgary that will be explained later.

Soft constraints:

- There are certain times of the day that nobody prefers but we'd like our system to produce a result that attempts to spread out the usage of our resources. To accomplish this, we'll define a ***soft constraint*** which is a minimum level of room usage we'd like a slot to achieve.

  We have for each slot s a $\boldsymbol{lecturemin}(s)$ and $\boldsymbol{tutorialmin}(s)$ that indicate how many lectures, resp. tutorials, should at least be scheduled into the slot s.

  Your system should be able to accept as input penalty values $pen_{\text{lecturemin}}$ and $pen_{tutorialmin}$ and for each lecture below $\boldsymbol{lecturemin}$ we will get $pen_{\text{lecturemin}}$ and for each tutorial $pen_{tutorialmin}$ added to the **Eval**-value of an assignment.

  Example. If $\boldsymbol{lecturemin}(s) = 5$ and we have assigned 3 we would have $2 * pen_{\text{lecturemin}}$ as a penalty.

- Certain departments have certain preferences regarding in which time slots their lectures and tutorials should be scheduled. Naturally, we see this as something that should be treated as ***soft constraint***. Each department can distribute points over pairs of (lecture/tutorial, time slots).

  Formally, we assume a function $preference\colon (\boldsymbol{Lectures} + \boldsymbol{Tutorials}) \times \boldsymbol{Slots} \to \mathbb{N}$ that reports those preferences. (i.e. we have a function that indicates a numerical natural number score of the preference of a lecture or tutorial being assigned to a slot.)

For each assignment in $\boldsymbol{assign}$, we add up the preference-values for a lectures/tutorials that refer to a different slot as the penalty that is added to the **Eval**-value of $\boldsymbol{assign}$. The penalty will accumulated be the specific number attached to each preference.

- For certain lectures and/or tutorials, we might prefer these to be scheduled at the same time.

  To facilitate this, there will be a list of $\boldsymbol{pair}(\boldsymbol{a}, \boldsymbol{b})$ statements in the input for your system, with $a, b \in \boldsymbol{Lectures} + \boldsymbol{Tutorials}$ and a parameter $pen_{notpaired}$ for your system.

  For every $\boldsymbol{pair}(\boldsymbol{a}, \boldsymbol{b})$ statement, for which $\boldsymbol{assign}$(a) is not equal to $\boldsymbol{assign}$(b), you have to add $pen_{notpaired}$ to the **Eval**-value of $\boldsymbol{assign}$.

The description of the basic version of our problem was aimed to be very general, so that this description can match the requirements of many different lecture scheduling situations that are possible. However, usually an organizing body doing this job will have additional ***hard***/***soft*** ***constraints*** (that might be realized using the ones we already described) and naturally they will also have concrete time slots and some organization regarding how they name and describe lectures and tutorials.

In the following, I will describe the instantiation of the general problem for the University of Calgary, and your task will be to write a system that solves the instantiated problem (Note that this is a completely fabricated artificial instantiation of the problem. As well this is a partial instantiation, and there are still many different specific search instances that can occur out of it).

Instantiation Specifics

Lectures/Tutorial Naming

Programs are generally communicated generally in a four-letter shorthand. For example,

$\boldsymbol{CPSC, DATA, SENG}$

are used for Computer Science, Data Science, and Software Engineering respectively.

Each program offers numerous courses. Course offerings for a semester are encoded by a 3-digit number.

$$\textbf{\textit{CPSC 383}}, \textbf{\textit{CPSC 433}}, \textbf{\textit{DATA 201}}$$

Course offerings sometimes require one or more lecture sections within those courses.

$$\textbf{\textit{CPSC 383 LEC 01}}, \textbf{\textit{CPSC 433 LEC 01}}, \textbf{\textit{DATA 201 LEC 01}}, \textbf{\textit{DATA 201 LEC 02}}$$

Each program wants to book time slots for their weekly lecture offerings. (Each one of the Program/CourseNo/Section combinations need a weekly scheduled time)

$$\textbf{\textit{Ex.}} \quad \textbf{\textit{Lectures}} = \begin{Bmatrix} \textbf{\textit{CPSC 383 LEC 01}}, \\ \textbf{\textit{CPSC 433 LEC 01}}, \\ \textbf{\textit{DATA 201 LEC 01}}, \\ \textbf{\textit{DATA 201 LEC 02}} \end{Bmatrix}$$

Remember, the larger set of tutorials would look more like the following if shown in more detail

$$\textbf{\textit{Ex.}} \quad \textbf{\textit{Tutorials}} = \begin{Bmatrix} \textbf{\textit{CPSC 383 LEC 01 TUT 01}}, \\ \textbf{\textit{CPSC 433 LEC 01 TUT 01}}, \\ \textbf{\textit{CPSC 433 LEC 01 TUT 02}}, \\ \textbf{\textit{DATA 201 LEC 01 TUT 01}}, \\ \textbf{\textit{DATA 201 LEC 02 TUT 01}} \end{Bmatrix}$$

If a tutorial is used by all lectures, then we drop the lecture code

$$\textbf{\textit{CPSC 433 TUT 01}}$$

Sometimes lectures have labs instead of tutorials. We will ignore any differences between the two and treat all LAB designations the same way we treat TUT designations. For example, Data Science often uses labs like

$$\textbf{\textit{DATA 201 LEC 01 LAB 01}}$$

Time Slots

The available time slots depend on the day of the week and whether we look at ***lectures*** or ***labs/tutorials***.

Mondays and **Wednesdays**, the slots available for ***lectures*** and ***labs/tutorials*** are

8:00-9:00, 9:00-10:00, 10:00-11:00, 11:00-12:00, 12:00-13:00, 13:00-14:00, 14:00-15:00, 15:00-16:00, 16:00-17:00, 17:00-18:00, 18:00-19:00, 19:00-20:00 and 20:00-21:00.

The same above slots are available for ***lectures*** also on **Fridays** (but see the ***hard constraints*** for connections between slots on these three days of the week for ***lectures***).

The available time slots for **Tuesdays** and **Thursdays** for ***lectures*** are

8:00-9:30, 9:30-11:00, 11:00-12:30, 12:30-14:00, 14:00-15:30, 15:30-17:00, 17:00-18:30 and 18:30-20:00.

For ***labs/tutorials***, the available time slots are the same on **Tuesdays** and **Thursdays** as previously given for **Mondays** and **Wednesdays** labs/tutorials.

The slots available for ***labs/tutorials*** on **Fridays** are

8:00-10:00, 10:00-12:00, 12:00-14:00, 14:00-16:00, 16:00-18:00, 18:00-20:00.

All slots beginning at 18:00 or later are called ***evening*** slots.

Note that the fact that the slots for ***lectures*** and ***labs/tutorials*** on **Tuesdays** and **Thursdays** are not following the same time scheme requires you to deal with time and how time slots may overlap. It also seems that this contradicts the general problem scheme, but it can be reformulated in terms of the general problem.

Our University of Calgary problem has the following ***hard constraints***:

- If a ***lecture*** (Ex. CPSC 231 LEC 01) is put into a slot on **Mondays**, then it must be put into the corresponding time slots on **Wednesdays** and **Fridays**. So, these three time slots are often treated as one abstract slot, which allows us to see our UofC problem as an instantiation of the general problem!
- Similarly, if a ***lecture*** is put into a slot on **Tuesdays**, then it must be put into the corresponding time slots on **Thursdays**.
- If a ***tutorial/lab*** (ex. CPSC 231 LEC 01 TUT 01) is put into a slot on **Mondays**, it must be put into the corresponding time slots on **Wednesdays**.
- If a ***tutorial/lab*** is put into a slot on **Tuesdays**, it must be put into the corresponding time slots on **Thursdays**.
- **Fridays** are single ***tutorial/lab*** slots and not linked with other days.

- **I don't care that the UofC now has WF slots or 2-hour tutorials on other days of the week than Friday! Let me keep the problem a bit simpler for you.**
- All ***lectures*** with a division number starting with the prefix "LEC 9" are ***evening*** lectures and must be scheduled into ***evening*** slots.
- All ***lectures*** in all tiers with a course number 5XX level must be scheduled into non-overlapping time slots.
- No ***lectures*** can be scheduled on **Tuesdays** 11:00-12:30 as a department meeting happens at this time once a month.
- There are two special tutorial/lab bookings for CPSC 851 and CPSC 913. These must be scheduled **Tuesdays** / **Thursdays** 18:00-19:00. CPSC 851 is not allowed to overlap with any ***labs/tutorials***/***lectures*** of CPSC 351 and CPSC 913 is not allowed to overlap with any ***labs/tutorials/lectures*** of CPSC 413. These are two unique slots saved for evening assessments in these courses. These bookings are only triggered if the respective lecture booking for CPSC 351 or CPSC 413 is requested in the input.

The University of Calgary also has the following ***soft constraints***:

- Different sections of ***lectures*** within a single department/course number should be scheduled at different times. For each pair of ***sections*** that are scheduled into the same slot, we add a penalty $pen_{section}$ to the **Eval**-value of an assignment $\boldsymbol{assign}$.

  Ex. CPSC 231 LEC 01 and CPSC 231 LEC 02 should not be scheduled at same time slot, if possible, for a better **Eval** score. (*often this is so instructor can teach both, and so that class is now has two times for students to try and fit into their own schedule*)

The information provided so far is enough to write the paper describing two search models and processes. Please use in this paper the terminology and symbols introduced here. I will make a description of the input file available shortly after your paper is due, so that those groups that have finished the paper can start writing the parser for their system. But note that I will select which search model and process a group will be implementing, so that starting on these parts of your program should not be done before this decision is made!