CPSC 433: Artificial Intelligence - Assignment Search Problem Description

(Version 1)

We (you!) oversee access to soccer (football) fields (pitches) around the city 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 ***Games*** and ***Practices*** to weekly field time ***Slots***. Note, for each time slot we have different quantities of fields available, and we won't be bothering with concerns about where those fields are in the city or other properties about them. (*There's 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 several leagues across the city. Each league has a desire to book time slots for weekly games for each of its divisions. As a result, we have a set

$\boldsymbol{Games} = \{\boldsymbol{g_1}, \ldots, \boldsymbol{g_m}\}$ of weekly games slots to book where one g_i is the weekly games slots to book for a division i in a league which may have multiple divisions.

We also have a set

$\boldsymbol{Practices} = \{\boldsymbol{p_{11}}, \ldots, \boldsymbol{p_{1k_1}}, \ldots, \boldsymbol{p_{m1}}, \ldots, \boldsymbol{p_{mk_m}}\}$ of practices associated with those divisional games. Practices $p_{i1}, \ldots, p_{ik_i}$ are associated with teams involved in the games g_i of a particular division i. Note, if $k_i = 0$ this would mean the division does not have any practices slots to be scheduled.

Finally, we have a set

$\boldsymbol{Slots} = \{\boldsymbol{s_1}, \ldots, \boldsymbol{s_n}\}$ of time slots into which games and practices must be assigned.

It is a fair comparison to think of games as lectures, practices as tutorials, and slots as time periods in the day. Lectures need to be booked to times in a week, some lectures have one or more tutorials that also need to be book 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.

For each slot s_j we have a limit $\boldsymbol{gamemax}(\boldsymbol{s_j})$ (a natural number) of games that can be put into the slot. This is because we have only that many fields available at that time to be booked.

For each slot s_j we have a limit $\boldsymbol{practicemax}(\boldsymbol{s_j})$ (a natural number) of practices that can be put into the slot. (Games and practices are independent from each other in this regard, i.e., a slot can take $\boldsymbol{gamemax}(\boldsymbol{s_j})$ games and $\boldsymbol{practicemax}(\boldsymbol{s_j})$ practices. This is a result of our practice fields being independent from our game fields which simplifies the problem.).

$\boldsymbol{gamemax}$ and $\boldsymbol{practicemax}$ are one example of ***hard constraints*** in this problem that must be fulfilled for a valid solution to be found. Another obvious ***hard constraint*** for this problem is that the practices for a particular division can never be in the slot in which the games of the division are so that players are available. (*On the other hand, we won't care if a practice for a division is booked immediately before/after games for a division 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 games and practices to slots that fulfills the ***hard constraints*** and optimizes the ***soft constraints***.

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

$\boldsymbol{assign}: \boldsymbol{Games} + \boldsymbol{Practices} \rightarrow \boldsymbol{Slots}$ 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 3 are restatements of prior constraints

1. Not more than $\boldsymbol{gamemax(s)}$ games can be assigned to each $s \in \boldsymbol{Slots}$.
2. Not more than $\boldsymbol{practicemax(s)}$ practices can be assigned to each $s \in \boldsymbol{Slots}$.
3. $\boldsymbol{assign(g_i) \neq assign(p_{ik_i})}$ for all i and k_i.

These are new ***hard constraints***

4. Some divisions allow players to play in both divisions and therefore want to make their games/practices not happen at the same time. The input for your system will contain a list of $\boldsymbol{notcompatible(a,b)}$ statements, with $a, b \in \boldsymbol{Games + Practices}$. For each of those, $\boldsymbol{assign(a) \neq assign(b)}$.
5. Sometimes there are certain divisions that already have pre-arrangements for certain slots. The input for your system can contain a partial assignment $\boldsymbol{partassign}$: $\boldsymbol{Games + Practices \rightarrow Slots + \{\$\}}$. ($ is a placeholder for no pre-assignment.) The assignment $\boldsymbol{assign}$ your system produces has to fulfill the condition:
   $\boldsymbol{assign(a) = partassign(a)}$ for all $a \in$
   $\boldsymbol{Games + Practices}$ with $\boldsymbol{partassign(a) \neq \$}$.
6. The input for your system can contain a list of $\boldsymbol{unwanted(a,s)}$ statements, with $a \in \boldsymbol{Games + Practices}$ and $s \in \boldsymbol{Slots}$. For each of those $\boldsymbol{assign(a) \neq s}$ must be true.
7. There will be additional hard constraints specific to the City 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 usage we'd like a slot to achieve. To facilitate this, we have for each slot s a $\boldsymbol{gamemin(s)}$ and $\boldsymbol{practicemin(s)}$ that indicate how many games, resp. practices, should at least be scheduled into the slot s. Your system should be able to accept as input penalty values pen_{gamemin} and $pen_{practicemin}$ (as command line arguments) and for each games below $\boldsymbol{gamemin}$ we will get pen_{gamemin} and for each practice $pen_{practicemin}$ added to the **Eval**-value of an assignment. Example. If $\boldsymbol{gamemin(s)} = 5$ and we have assigned 3 we would have $2 * pen_{\text{gamemin}}$ as a penalty.
- Certain leagues have certain preferences regarding in which time slots their games and practices should be scheduled. Naturally, we see this as something that should be treated as ***soft constraint***. Higher age groups and tiers are awarded a certain number of ranking points and lower ones fewer ranking points. Each league can distribute these points over pairs of (game/practice,

time slots). Formally, we assume a function
$preference: (\boldsymbol{Games} + \boldsymbol{Practices}) \times \boldsymbol{Slots} \rightarrow \mathbb{N}$

that reports those preferences. (i.e. we have a function that indicates a numerical natural number score of the preference of a game or practice being assigned to a slot.)
For each assignment in $\boldsymbol{assign}$, we add up the preference-values for a games/practices that refer to a different slot as the penalty that is added to the **Eval**-value of $\boldsymbol{assign}$.

- For certain games and/or practices, 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 $a, b \in \boldsymbol{Games} + \boldsymbol{Practices}$ 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 game 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 games and practices.

In the following, I will describe the instantiation of the general problem for the City 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

Games/Practice Naming

In Calgary, weekly games to be booked games are identified by an organizing body, an age group (and often a tier within that age group), and division in the age group. For example, the main soccer body in Calgary is CMSA who has several different age groups/tiers it operates, but there are also leagues like CUSA and CSSC that operate in the city. A league will have an indicator that includes the division in that sub-league that looks like:

CMSA U12T1 DIV 01

For Calgary Minor Soccer Association - Under 12 - Tier 1 - Division 1.

Practices add on the end of this indicator

CMSA U12T1 DIV 01 **PRC 01**

is practice 1 for CMSA U12T1 DIV 01. If a practice is used by all divisions of the age/tier level, then we drop the division

CMSA U12T1 PRC 01

Sometimes a league doesn't book practices but instead generally creates open field access for their division. Instead of PRC we will see indicator OPN. You will be able to treat OPN as synonymous with PRC for this project as field bookings operate identically.

CMSA U12T1 DIV 01 **OPN 01**

Time Slots

The available time slots depend on the day of the week and whether we look at ***games*** or ***practices***.

Mondays and **Wednesdays**, the slots available for ***games*** and ***practices*** 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 slots are available for ***games*** also on **Fridays** (but see the ***hard constraints*** for connections between slots on these three days of the week for ***games***).

The available time slots for **Tuesdays** and **Thursdays** for ***games*** 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 ***practices***, the available time slots are the same on **Tuesdays** and **Thursdays** as previously given for **Mondays** and **Wednesdays** practices.

The slots available for ***practices*** 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 ***games*** and ***practices*** 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 City of Calgary problem has the following ***hard constraints***:

- If a ***division's games*** (Ex. CMSA U12T1 DIV 01) are put into a slot on **Mondays**, then they must be put into the corresponding time slots on **Wednesdays** and **Fridays**. So, these three time slots are treated as one abstract slot, which allows us to see our Calgary problem as an instantiation of the general problem!
- Similarly, if a ***division's games*** are put into a slot on **Tuesdays**, then they must be put into the corresponding time slots on **Thursdays**.
- If a ***practice*** (ex. CMSA U12T1 DIV 01 PRC 01) is put into a slot on **Mondays**, it must be put into the corresponding time slots on **Wednesdays**.
- If a ***practice*** is put into a slot on **Tuesdays**, it must be put into the corresponding time slots on **Thursdays**.
- **Fridays** are single ***practice*** slots and not linked with other days.
- All ***divisions*** with a division number starting DIV 9 are ***evening*** divisions and must be scheduled into ***evening*** slots.
- All ***games*** in all tiers of the U15/U16/U17/U19 level must be scheduled into non-overlapping time slots.
- No ***games*** can be scheduled on **Tuesdays** 11:00-12:30 as a league wide meeting occurs weekly at this time for admin.
- There are two special "game bookings" CMSA U12T1S and CMSA U13T1S that must be scheduled **Tuesdays** / **Thursdays** 18:00-19:00. CMSA U12T1S is not allowed to overlap with any ***practices***/***games*** of CMSA U12T1 and CMSA U13T1S is not allowed to overlap with any ***practices/games*** of CMSA U13T1. These two "practice bookings" are a way of schedule special showcase tryouts series for these divisions' players for selection to special provincial teams for Alberta Games.

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

- Different ***divisional games*** within a single age/tier group should be scheduled at different times. For each pair of ***divisions*** that is scheduled into the same slot, we add a penalty $pen_{section}$ to the **Eval**-value of an assignment $\boldsymbol{assign}$. Ex. U12T1 DIV 01 and U12T1 DIV 02 should not be scheduled at same time slot, if possible, for a better **Eval** score.

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!