CPSC 433: Artificial Intelligence - Assignment Search Problem - Input and Output Definition

(Version 1)

Your system should be able to take as input a text file and positive integer inputs. The simplest way to do this is through the command line, but a GUI is an option as well if your group so chooses. The text file will contain a description of the specific problem being solved, and the positive integer inputs will be used for soft constraint penalty values and as additional multiplier weightings to soft constraint categories.

Example command line execution

java CPSC433F22Main filename $w_{minfilled}$ w_{pref} w_{pair} $w_{secdiff}$ $pen_{gamemin}$ $pen_{practicemin}$ $pen_{notpaired}$ $pen_{section}$

java CPSC433F22Main input.txt 1 0 1 0 10 10 10 10

The definition and usage of these specific weights and penalties will be addressed later in this document. The four penalties should already be familiar from the prior document discussing the project problem description. The four weights were not in that document.

You will generate a search instance (your starting state) based on the text file, which contains a few key words and then describes the information as tables.

Every key word header (Ex. **Game slots**:) will be in every input file, although sometimes there will be no data under a header. For example, if there are no paired games/practice, or no pre-assigned games/practices. The quantity of empty lines between each table is not guaranteed. Each line in the input file (that is not a key word header line or empty line) is a row in one of the key word tables. You should use keyword headers as table breaks. The fields within a row are separated by commas! Please note, that extra spaces, or missing spaces around commas may occur in valid input files.

The general scheme of an input text file is as follows:

Name:
Example-name

Game slots:
Day, Start time, gamemax, gamemin

Practice slots:
Day, Start time, practicemax, practicemin

Games:
Game Identifier

Practices:
Practice Identifier

Not compatible:
Game Identifier, Game Identifier
Game Identifier, Practice Identifier
Practice Identifier, Practice Identifier

Unwanted:
Game Identifier, Slot day, Slot time
Practice Identifier, Slot day, Slot time

Preferences:
Slot day, Slot time, Game Identifier, Preference value
Slot day, Slot time, Practice Identifier, Preference value

Pair:
Game Identifier, Game Identifier
Game Identifier, Practice Identifier
Practice Identifier, Practice Identifier

Partial assignments:
Game Identifier, Slot day, Slot time
Practice Identifier, Slot day, Slot time

Naturally, this requires some additional explanations:

- Possible days for all references to game slots are **MO** and **TU** only (due to the additional hard constraint of the City of Calgary that forces games played on monday to be played on wednesday and friday at the same time, resp. games played on tuesday to be played on thursday at the same time).
- Possible days for all references to practice slots are **MO**, **TU**, and **FR**.
- The possible start times for all available slots are stated in the problem description given prior.
- Every slot that is available must have an entry in the input file. If one of the possible slots does not occur in the input file, we assume that $\boldsymbol{gamemax}$ and $\boldsymbol{gamemin}$ (resp. $\boldsymbol{practicemax}$ and $\boldsymbol{practicemin}$) is 0, which means that no games (practices) can be scheduled into this slot!
- We schedule only games that occur in the list after **Games**:
  *Exception: if we have under **Games**: games starting with CMSA U12T1 or CMSA U13T1, we have to schedule CMSA U12T1S (resp. CMSA U13T1S) in the indicated slots (note, they are scheduled into practice slots) and have to add all the constraints regarding them that are given in the problem description.*
- We schedule only practices that occur in the list after **Practices**:
- A Game Identifier has the form described in the problem description (see also the example below).
- A Practice Identifier has the form described in the problem description (see also the example below).
- The entries after "**Not compatible**" identify all games and practices that are not compatible with each other. But note that there have to be also some additional built-in incompatibilities, as described in the problem description! Note also that incompatibilities are symmetric!
  Finally, there is no particular order of the 3 types of statements, they can be mixed in this section!
- The entries under "**Unwanted**" provide a list of games/practices and slots in which we do not want the games/practices to be scheduled. Game statements and practice statements can occur in any possible order.
- "**Preferences**" can be expressed for games and practices, and we do not require a particular order here. i.e. it is not necessary to list all games first! If a preference entry does not refer to a valid time slot, you can ignore this entry (but you might want to print out a warning)! Note that we can have preferences for games/practices and slots that also appear in the **"Unwanted**" list.

- The entries under "**Pair**" identify games that we want to be scheduled at the same time, if possible.
  Again, there is no particular order of the 3 types of statements, they can be mixed in this section!
- The order of the entries under "**Partial assignments**" is not fixed, again. If an entry is not valid (game with a slot that is not a game slot or practice with a slot that is not a practice slot, except for CMSA U12T1S and CMSA U13T1S) terminate with an error message.

Some of the questions regarding input-files from previous years were:

1. is the format static?, i.e. do the key words occur in the given sequence - > Yes
2. what to do with additional blanks in inputs? -> your parser should filter them
3. what kind of GUI do I expect? -> CMD line is sufficient (GUI optional), regarding your output, see below.

The positive integer inputs will mostly have to be concerned with the various parameters that deal with the soft constraints and essentially for the function $\boldsymbol{Eval}$. I would like you to define $\boldsymbol{Eval}$ using four subfunctions $\boldsymbol{Eval_{minfilled}}$, $\boldsymbol{Eval_{pref}}$, $\boldsymbol{Eval_{pair}}$ and $\boldsymbol{Eval_{secdiff}}$ that are producing a value for each of the 4 types of soft constraints.

Then we have

$$\boldsymbol{Eval}(assign) = \boldsymbol{Eval_{minfilled}}(assign) * w_{minfilled} +$$
$$\boldsymbol{Eval_{pref}}(assign) * w_{pref} +$$
$$\boldsymbol{Eval_{pair}}(assign) * w_{pair} +$$
$$\boldsymbol{Eval_{secdiff}}(assign) * w_{secdiff}$$

with $w_{minfilled}$, w_{pref}, w_{pair} and $w_{secdiff}$ being weights that tell the system how important the different soft constraints are. Note that I will have examples that require as values for these weights a 0! Reminder pen_{gamemin}, $pen_{practicemin}$, $pen_{notpaired}$ and $pen_{section}$ are being used by the individual **Eval** sub-functions. $\boldsymbol{Eval_{minfilled}}$ uses pen_{gamemin}, $pen_{practicemin}$. $\boldsymbol{Eval_{pair}}$ uses $pen_{notpaired}$. $\boldsymbol{Eval_{secdiff}}$ uses $pen_{section}$. $\boldsymbol{Eval_{pref}}$ uses the specific individual values in input file under "**Preferences**:".

The following is a short example input file that your parser should be able to parse without error (note there is some inconsistent spacing in this input):

Name:
ShortExample

Game slots:
MO, 8:00, 3, 2
MO, 9:00,3,2
TU, 9:30, 2, 1

Practice slots:
MO, 8:00, 4, 2
TU, 10:00,2,1
FR, 10:00, 2, 1

Games:
CSMA U13T3 DIV 01
CSMA U13T3 DIV 02
CUSA O18 DIV 01
CSMA U17T1 DIV 01

Practices:
CSMA U13T3 DIV 01 PRC 01
CSMA U13T3 DIV 02 OPN 02
CUSA O18 DIV 01 PRC 01
CSMA U17T1 PRC 01

Not compatible:
CSMA U13T3 DIV 01 PRC 01, CSMA U13T3 DIV 02 OPN 02
CSMA U17T1 DIV 01, CSMA U13T3 DIV 01
CSMA U17T1 DIV 01, CSMA U13T3 DIV 02
CSMA U17T1 PRC 01, CSMA U13T3 DIV 02
CSMA U13T3 DIV 01, CSMA U17T1 PRC 01

Unwanted:
CSMA U13T3 DIV 01, MO, 8:00

Preferences:

TU, 9:00, CSMA U13T3 DIV 01, 10
MO, 8:00, CSMA U13T3 DIV 01 PRC 01, 3
TU, 9:30, CSMA U13T3 DIV 02, 10
TU, 10:00, CSMA U13T3 DIV 01 OPN 01, 5
MO, 8:00, CSMA U13T3 DIV 02 OPN 02, 1
MO, 10:00, CSMA U13T3 DIV 02 OPN 02, 7

Pair:
CUSA O18 DIV 01, CSMA U17T1 DIV 01

Partial assignments:
CUSA O18 DIV 01, MO, 8:00
CUSA O18 DIV 01 PRC 01, FR, 10:00

As output from your system, I expect an assignment presented in the following form and the **Eval**-value that your system assigns to the assignment. There are several ways how we could output an assignment, but I prefer it ordered by games. This means that we order the games alphabetically. After each game, the practices for the game (also alphabetically). At the end of each game or practice, output the assigned slot using the assigned days and times that identified it in the input file.

Here is an example (that is not the solution to the above input example, please create your own test examples for the system functionality):

Eval-value: 30
CSMA U13T3 DIV 01 : MO, 10:00
CSMA U13T3 DIV 01 PRC 01 : TU, 10:00
CSMA U13T3 DIV 02 : MO, 14:00
CSMA U13T3 DIV 02 OPN 02 : MO, 8:00
CSMA U17T1 DIV 01 : TU, 9:30
CSMA U17T1 PRC 01 : MO, 8:00
CUSA O18 DIV 01 : MO, 8:00
CUSA O18 DIV 01 PRC 01 : FR, 10:00