Grundlagen der Künstlichen Intelligenz

Programming Exercise 2: Constraint Satisfaction Problem 4th December 2020

Matthias Mayer,
Jiaying Huang and Xiangyu Zhao

(last updated 1st December 2020)

Problem 2: Welcome Party

The student union is planning to hold a welcome party for the new students. 8 Students (Abigail, Brian, Caroline, Daniel, Edith, Frank, Grace, Harold) volunteer to perform in several shows. The following four types of shows are allowed to be performed on the party. (Each type of show is performed no more than once):

- Singing (costume for each student costs 50)
- Dancing (costume for each student costs 80)
- Comedy (no costs)
- Piano performance (costume for each student costs 100)

Consider the following constraints:

1. Everyone should participate in exactly one show
2. No one performs alone
3. Every type of show is performed (each show must be performed at least by one student)
4. Total budget is less than or equal to 400
5. Singing requires at least 3 students if it is performed
6. Dancing requires a maximum of 2 students if it is performed
7. Comedy requires 2 or 3 students if performed
8. Piano is played by a single student if it is performed
9. Piano is played by two student if it is performed
10. Abigail performs together with Harold (other students could also perform with them)
11. Frank, Grace and Harold don't perform in the same show because they don't get along (any two of them don't want to perform together.)
12. Brian and Daniel don't want to dance
13. Abigail and Caroline don't want to perform the comedy
14. Edith and Grace want to sing (other students could also sing with them)
15. No one plays the piano

Model the constraint satisfaction problem in Python. For each of the following subsets of constraints, find the solution, if it exists:

Problem 2.1: { 1 – 2, 5 – 7, 9 – 14 }

Problem 2.2: { 1, 3 – 8, 10 – 12, 14 }

Problem 2.3: { 1, 3 – 7, 9 – 13 }

Problem 2.4: { 1 – 3, 6 – 7, 11 – 14 }

Problem 2.5: { 1 – 2, 4 – 7, 9 – 14 }

Problem 2.6: { 1 – 2, 4 – 7, 10 – 15 }

Note that problems `2.3` can not be satisfied.

Programming Framework

For this programming exercise a *Jupyter Notebook* will be used. The template for the exercise can be found in ARTEMIS[1]. To model the constraint satisfaction problem, you should know or look up Python's lambdas, lists and dictionaries. The following steps are required to correctly set up the environment for the programming exercise:

1. **Installation of Anaconda and Download of the AIMA python code** If you do not already have the *Jupyter Notebook* environment installed on your machine, the installation is the first step you have to perform. We recommend to install *Anaconda*, since this will set up the whole environment for you. The template for the programming exercise is based on the code from the *AIMA python*[2] project. Therefore, you first have to download the code from this project before the template can be used. Instructions for installation of Anaconda and AIMA python code can be found in "Instructions on how to run the Jupyter Notebooks" on Moodle[3].
2. **Pull of the template:** Pull the repository with the template from ARTEMIS, which can be done similarly to AIMA python just with the repository link from ARTEMIS. To avoid issues with the relative file paths, we recommend to copy all files contained in the template into the root-directory of the *AIMAcode* project that you downloaded in the previous step.

After completing the above steps, you are all set up to start with the exercise. The main function of the template is the *Jupyter Notebook* **csp.ipynb**, which is also the only file you have to work on. Your task is to model the Welcome Party problem. An example, on how to model a constraint satisfaction problem using the *AIMAcode*, is provided in the notebook. This is the same as given in Exercise 3.4.

Submission

For submission, you have to upload the following files in ARTEMIS:

1. Copy **csp.ipynb** (notebook containing your solution for modelling the Welcome Party problem) to the pulled repository.
2. Add and commit the altered notebook and push it to ARTEMIS with

   ```
   git add csp.ipynb
   git commit -m "A commit message"
   git push
   ```

   (all within the ARTEMIS repository)

A pass will be awarded only if:

1. you submitted the **correct file** with the **correct name**, as shown above.
2. you **did not zip** your file.
3. you **pushed your files to your ARTEMIS branch**.
4. you **did not change** the **variable names** provided by us within the template.
5. your submitted files can be run in an Anaconda environment (Python 3.7) with the packages provided by the *requirements.txt* in the *aima repository*, the utils.py, the search.py and the csp_programming_exercise.py provided by us **within a reasonable time (under 5 minutes)**.
6. the problem has been modelled correctly using the NaryCSP class from the module csp_programming_exercise.
7. like the rest of the programming exercises, this is an individual project and work **must** be your own. (We will use a plagiarism detection tool and any copied code will annul all bonus exercises from both the copier and the copied person!)

Submission will close on **Friday, 1.1.2021 at 23:59**. Your solution will be marked by ARTEMIS. There will be feedback on formatting errors and rightly solved CSP. Nonetheless, it is very important to follow the instructions exactly!

[1] https://artemis.ase.in.tum.de/#/courses/85/exercises

[2] https://github.com/aimacode/aima-python

[3] https://www.moodle.tum.de/pluginfile.php/2589427/mod_resource/content/1/AIMAinstallation.pdf

We offer preliminary checks of your solution and ARTEMIS will show your progress. You can submit your solution multiple times and get feedback for each submission. Your final submission will be checked. We award 1 point if all checks including plagiarism pass.