02285 AI and MAS, SP19
Guidelines for the Programming Project report

Thomas Bolander

The overall goal of your report is that it should look as much as possible like a conference publication. By aiming for this, you get a bit of training in writing academic papers, which is very different from e.g. writing software documentation reports. Below are a number of guidelines for how to achieve this, plus some specific requirements for formatting your report.

Report style and length

The report should be prepared in the AAAI style, which is also used for the major international AI conferences like IJCAI (International Joint Conference on Artificial Intelligence) and AAAI. The file `AuthorKit.zip` includes the instructions for typesetting a report/paper using this style. It is possible to typeset your report in Word, but we strongly recommend LaTeX (it is never to late to learn to use LaTeX even if you didn't so far). The detailed instructions for formatting your report using LaTeX is in the file:

`formatting-instructions-latex.pdf`

and the corresponding instructions for Word are in:

`formatting-instructions-word.docx`

The paper can be **at most 6 pages long** plus up to one additional page of references.

Overall guidelines

Your report should focus on your problem analysis, your solution, and an analysis of your findings and results. The reader of the report is expected to be familiar with the project description given in `assignment_prog_proj.pdf` and the entire curriculum of the course. Your report should describe your ideas, solutions, and original contributions. You should describe your implemented system in terms of its overall algorithmic functionality, not in terms of implementation details. Your report should make clear what methods within AI and MAS you have been implementing, using the relevant concepts and notions from the course curriculum whenever possible. Using the concepts and notions from the course curriculum often makes it possible to express yourself in a more concise way, e.g. by writing "goal decomposition" instead of "splitting the overall problem into subproblems". Another example is that heuristics are often based on relaxed problems,

and can be easier to describe in terms of problem relaxation than without using any such underlying concept. Finally, make to sure reflect on the strengths and weaknesses of your solution.

You should make sure to include screenshots of your submitted competition levels in the report, and briefly comment on their design (Why do they look like this? What makes them challenging? What are they testing?). If you can't include screenshots of a reasonable size in the report itself, put them after the references in an appendix.

Structure of the report

Below is a recommended structure of your report. You are not obliged to strictly follow this structure, but all questions below should be addressed somewhere, and the abstract is mandatory.

1. **Abstract**. A short summary of the paper and its main results/insights. Should preferably be less than 150 words.

2. **Introduction**. What problem are you trying to solve? Why is this important? What are the main results/accomplishments/highlights of the paper. Since the reader is assumed to be familiar with the details and goal of the programming project, this section will probably be fairly short. But you can use it to briefly describe the type of solution you have chosen and why.

3. **Background**. Briefly present the theories that you work relies on. Note that you are allowed to expect the reader to be familiar with everything presented in the course curriculum, but you should at least mention which of these theories you are using and make suitable references. Even for the theories in the course curriculum, it might also be necessary to settle the notational conventions, as these sometimes differ between references. If you employ theories and ideas from outside the course curriculum, you should explain them in a bit more detail, and of course also make suitable references. In general, any specific work you might have used in completing the project (books, articles, implementations, online resources, etc.) should be referenced.

4. **Related work**. Has this been done before? What is the closest related research? How does your work differ? Related work is sometimes integrated into the introduction or background, and sometimes it is made a separate section towards the end of the paper. To make the related work section, you will be required to do some literature search to see if you can find papers that use similar methods (or combinations of methods) on similar types of problems. A piece of related work could for instance be if someone has written a paper on using similar AI methods for the Sokoban game. Sometimes it can be hard to find related work, but you should do your best. Use e.g. Google and Google Scholar(if a certain relevant paper is licensed, try to download it via `findit.dtu.dk`).

5. **Methods**. How does your client work? Describe its overall algorithmic functionality as clearly and precisely as possible, without going into any implementation details. Describe it in terms of the theories you rely on (as presented in background). Include small, illustrative examples of how the client works. Aim to describe your client sufficiently precisely that a reader being an expert in the field would be able to implement a similar solution based on your description (note that this is not an encouragement to describe implementation *details*, but rather make sure that the overall structure and ideas of your implementation are sufficiently precise).

6. **Experiments/results**. Benchmarks (test results) and possibly competition results. Include explanations and analyses of your results. As mentioned under "Benchmarks and comparisons" below, it is usually a clear strength if you have some results to compare, e.g. two different versions of your client with different methods/features/parameters. You could also compare the behaviour of your client on different individual levels that illustrate a certain point, e.g. similar levels resulting in large differences in the client's behaviour.

7. **Discussion**. What are the strengths and weaknesses of your solution? Why did you choose your particular solution method, and did it work out as expected? If it did not work as expected, what should have been different? Here you can also include a more thorough comparative analysis if you have implemented different methods/clients. Which worked best? Could you have done even better? How?

8. **Future work**. What are the most immediate possible extensions or generalisations of your work? How would you develop it further if you had more time? Does your method generalise to other types of domain? If not, what would be required for it to do so? Sometimes it makes most sense to integrate discussion and future work into a single section. Future work should focus on immediate extensions that you more or less would know how to implement, it shouldn't be very speculative extensions that are not even quite clear to yourself.

9. **References**. Ideally you should have at least 6 references, of which at least 3 should be academic papers. These can e.g. papers you find when looking for related work. It is not enough to only cite Russell & Norvig, because there will surely be other work that is more directly relevant to your problem and solution. But you are of course allowed *also* to cite Russell & Norvig, e.g. saying "We use the conventions for STRIPS actions presented in [Russell & Norvig]".

The most important sections/parts of your paper are 5, 6 and 7 above (methods, experiments/results and discussion). These would usually take up the major part of the article.

More on benchmarks and comparisons

Make sure to include benchmarks of you client in the report. If you wish, you can refer to the competition results. When reporting benchmarks, make sure to mention the hardware used (e.g. "All experiments were performed on a computer with a single Intel i7-4510U CPU core and 8GB of RAM.").

It is often a very good idea to seek to benchmark different solutions/clients against each other, as this can lead to many interesting insights and discussions. It might be that you have several different versions of your client that you can benchmark against each other, or maybe you have some features that can either be turned on or off. You can also consider to team up with another group attacking the problem with different techniques, and then you can compare the respective strengths and weaknesses of your chosen techniques.

It is important that you do not only include the benchmark data, but *analyse* it thoroughly. Why do the results look like this? Why does this small change in level have this major impact on performance?

Final words

If after reading this you are still uncertain about what is expected from you in the report, please contact the teachers. A good way to get a feeling for the expected structure of a report like this is to look at the structure of the papers you find when looking for related work. This is particularly true for conference articles that tend to have the same format and structure as required here.