# Assignments 2016-2017

# Constraint Programming (2IMI25)

The CP assignments are to be done in pairs of people. If you do not have a partner for the assignments, you should simply contact me. I will collect the names of the people that do not yet have a partner and will take care of forming pairs. I do not want three (or more) people in one group as that represents too much of an advantage.

The assignments will be finalized by an oral exam around the *second* assignment. This oral exam will take about 15 minutes. Potential discussion topics are the OPL model you developed for the assignment, the way you got to this model, the report you wrote, and the results you obtained. You will need to have the running OPL model on your PC with you. We'll be sitting together at the same table, so I'll be able to look at the screen of your laptop. If I have any doubts on whether the developed code is indeed developed by you, I may contact you before the oral exam and/or discuss this at the oral exam. Both people that hand in the assignments are responsible for all material they hand in, i.e., one cannot answer a question by saying “I don’t know, as that part was developed by my partner.”. Well, at least not without it having a negative effect on your grade. After the oral exam, I will determine a final grade. The final grade is composed of the grade for the first assignment (25%) and the grade of the second assignment including the oral exam (75%).

The first assignment will not be about scheduling where the second assignment will be about scheduling. As the non-scheduling material is treated in earlier lectures, you will be able to do the first assignment sooner. The first assignment will also be easier than the second and will not require writing a report, which is reflected in the grading explained above.

For both assignments I will test your model on a set of small test instances (STI) and a set of larger test instances (LTI). As will be explained in class, small test instances are not aimed at testing performance. They are in fact so small that failure to get the, often optimal, result I believe is easily reachable within the given time limit, is indeed considered a failure of your model. Other simple things like your model not running correctly due to criterion weights that are ignored or criterion weights that are assumed integer (second assignment), will also be considered a failure of your model. To prevent this and other failures to happen, you are well advised to develop your own set of small test instances, so as to convince yourself your model is indeed correct. You should build a well-designed set of small test instances, showing that your model correctly finds good/optimal solutions. Doing tests on instances you generate yourself is important both to check validity of your model (mostly through small test instances) as well as to assess the performance behavior of your model (by devising your own larger test instances, see below). To motivate you, help you, and provide you feedback on the model you are developing, I will send some instances in STI that I devised as we go, together with a description of what the result should be. Note that these instances and their expected results are a valuable source of feedback. I advise you to look at them closely and amongst others make sure your final model correctly solves them. I will check that and failure of your final model on any instance in STI counts severely. As explained though you should start devising small test instances yourself as soon as possible as that will increase the probability that you will develop a correct and well-performing model before the deadline. As to LTI: for the first assignment I will provide you with this set of larger test instances when communicating the assignment, for the second assignment I will do so within two weeks after communicating the assignment. Note that you can already make progress without LTI, and in fact it is my experience that turning away too soon from testing on small test instances to test on LTI leads to a less efficient way to develop a good model.

For the first assignment, you will *just send me your final model*. For the second assignment you will *send me your final model together with the report discussing the model and the results you obtained* (see below and the assignment description).

* The first assignment (final model thus) is due 7 October 2016 end-of-day CET.
* The second assignment (final model and report) is due 6 November 2016 end-of-day CET.
* The oral exams will be on 10-11 November 2016.

Missing a submission deadline disqualifies you from being graded. **Submission needs to be done via mail to** [**w.p.m.nuijten@tue.nl**](mailto:w.p.m.nuijten@tue.nl) **and the subject of the mail needs to start with “2IMI25 Assignment Submission” *followed by both your names*.**

At the final deadline you need to hand in the working OPL model plus the report in case of the second assignment, in one archive/zip file. Both the models and the report should clearly mention name, student number, and email address of the authors. The models need to be in one .mod file, so include statements of other files are not allowed. All settings and settings alternatives should also be in the .mod files, i.e., not in settings (.ops) files, so that I have all information in one place. The version of the model that will run when I test the model without any changes should be your overall best model, i.e., the one with which you have obtained the best results on LTI for the described time limit.

You should not repeat the problem description in the report (second assignment) as I know that description already. As such you can quickly move to describing the problem model: which decision variables did you use, what constraints did you identify, which objective function did you define? You need to explain what OPL code you developed, motivate the choices made in the model, and describe the relevant output and results obtained. Use Courier New or equivalent for OPL code in the report.

Final remark before turning to how your work will be graded. The “execute” block with parameters that you will surely have has to go before the “subject to” block and the “execute” block which does solution handling has to go after the “subject to” block. This makes sense as the parameters are required to be known before the model is run and a solution is only known after the model is run. Note by the way that it is good practice is to keep all parameters together in one execute block as that makes you have an easy overview.

Guidelines for grading the first and second assignment:

* To qualify for final grading, you need to have a working and efficient OPL model generating correct results on the test instances in LTI. The model needs to be clear and well-written with clear comments.
* You should have a generic, extensible approach where possible.
* I will test your final OPL model using different instances also outside LTI. The largest or hardest instances in LTI do represent the size and the difficulty your model should be able to handle.

Additional guidelines for grading the second assignment (seriously more extensive):

* You need to provide a clear, well-written report. The quality of the report will have a serious influence on the grade, i.e., just having a working OPL model is not sufficient.
* Your report needs to have page numbers (and yes, people do hand in reports without them, making it hard to refer to a certain page when commenting and discussing).
* As described below you will be doing a set of tests. It is then absolutely crucial that you explain in the report what your *conclusions* of these tests are. Simply describing the tests and the computational results is insufficient.
* As to presenting the computational results in the report. I at least want to see one table where you report the results obtained with your best model for all instances in LTI within the described time bound, together with whatever settings you used. Note thus that these settings for these results will be common to all instances in LTI.
* The table should include a row for each instance in LTI and have columns giving:
  + the found objective function value
  + the time spent in solve (to be found in the "Engine log" tab),
  + the number of constraints,
  + the number of variables,
  + the number of fails,
  + (in case of scheduling) the number of intervals,
  + (in case of scheduling) the number of sequences,
  + the used propagation (inference) level.
* I furthermore want to at least see a table, again having rows and columns as described above, where for each instance in LTI you give the best objective function value you ever found in the computational study you did, together with whatever settings you used. I will need to be able to reproduce these results, so you should explain what should be changed in the .mod file to get these results. This table thus can include results where you spent more time, where you used a different inference level, where you did some specific search that happened to work well for this instance, etc.
* I also want to see a study of how the inference level influences performance as well as a study of how available computation time influences performance.
* Note that you ***cannot change anything*** in the .dat files or the .xls files that I will send you. Your model should run with these files as I sent them to you. If it doesn’t, I can’t grade your work.
* Failing any of the following points leads to a lower grade. As such failing or underperforming on several of these will make that you will not pass the course.
  + You should do performance testing by any combination of the following:
    - have several alternative ways of modeling, for example by having redundant constraints, symmetry breaking constraints, dominance rule constraints,
    - try different inference levels,
    - try with and without specific search.

You should compare the results of the different alternatives and report on that. Tables that for example show the effect of your smart symmetry breaking constraints or of the specific search that you came up with, will have a positive influence on your grade. Once again, all alternative ways of modeling should appear in the one .mod file that you will deliver, where obviously the code that did not make it in the final best model will be commented out.

* + Using your best model, you should test on additional non-trivial instances that you construct yourself showing the strengths and limitations of your model and report on that. An analysis of variations of the data that makes instances more challenging or actually easier to solve will again have a positive influence on your grade. Also, if you found that certain good ideas did not pay off on the instances provided, you could think to construct instances where they do. For example, if you found a good symmetry breaking constraint, but that symmetry is hardly present in the provided instances, you could construct an instance where that symmetry is present and show that your constraint indeed improves performance.
  + Note that if you feel the performance of your model is not great, for example because longer runs give considerably better results, you need to show what you tried to improve performance (and thus show that what you tried did not improve performance).
* Optionally you can report on negative results, i.e., changes to the model that looked like a good idea, but in the end proved to make no difference or even deteriorate performance. It is of importance then that the changes indeed look promising, i.e., a list of silly proposals will have a negative rather than a positive influence on your grade.

Finally, some general guidelines to follow:

* Do not include the complete code listing in your report. I can read that in the .mod file you will include.
* Use parts of the code listing to explain how you modelled the problem.
* Always use sequential search (number of workers = 1) in all tests.
* If you have asserts, name them so as to get meaningful messages.

Have fun!

Wim Nuijten.