Modelling for Combinatorial Optimisation (1DL451) and Constraint Programming (1DL442) Uppsala University – Autumn 2024 Assignment 1 (pass/fail)

Prepared by Pierre Flener, Gustav Björdal, and Jean-Noël Monette

— Deadline: 13:00 on Friday 13 September 2024 —

The scope of this assignment is Topics 1 to 2: you need ***not*** show any knowledge of subsequent topics. The source code of the demo report has problem-independent indications on how to proceed. Read the *Submission Instructions* and *Grading Rules* at the end of this document. It is strongly recommended to prepare and attend the help sessions, as huge time savings may ensue.

Combinatorial optimisation is intended for complex real-life problems, but we warm up slowly, with simple puzzles and simplified real-life problems. Model the following problems[1] (by starting from copies of the problem-independent model skeleton.mzn) and evaluate the performance of backends of ***all*** the solving technologies considered in the course (see Problem 1). If you make assumptions that are not part of a problem formulation, then state them clearly in the report and model.

Some of the problems have more than one solution, or more than one optimal solution. Also, some of the problems can actually be solved without search, either by paper-and-pencil analysis or by polynomial-time algorithms. Models can however be written without much knowledge of such analytic or algorithmic processes. The objective of this assignment is to assess whether the MiniZinc toolchain is of help even for such computationally simple problems. Speed is not an issue for Problems 2 and 3, and you should finish *all* problems and the report *before* revisiting Problems 4 and 5 towards tuning those models until your self-allocated time budget is up.

1 Solving Technologies (pass/fail)

Give the acronym and full name of the solving technology of each backend that is used by our experiment-running script `run_backends.sh` explained in our cheatsheet.

2 The Nine Children (pass/fail)

There is a person with nine children of different ages, expressed as integers. There are equally long gaps between the ages of any two consecutively born children. The squared age of the parent is the sum of the squares of the ages of the children. Perform the following sequence of tasks, where you can assume that no person gets older than 150 years:

[1] Solo teams may skip Problem 2, but are highly encouraged to try it nevertheless.

A. Write a MiniZinc model called `nineChildren.mzn` in order to find the ages of the children and their parent. Follow the instructions of Section B of the demo report for how to document your model; this applies to ***all*** the problems of ***all*** the assignments.

B. Evaluate your model, writing the age of the parent in the result table. Follow the instructions of Section C of the demo report for how to evaluate your model; this applies to ***all*** the problems of ***all*** the assignments.

C. How old are the children and their parent? Describe how to use the MiniZinc toolchain in order to determine whether this solution is unique and, if not, to determine a solution with the youngest parent.

3 Halmos and his Wife (from Paul Halmos) (pass/fail)

Paul Halmos, the mathematician, and his wife attended a dinner party attended by four other couples. During the cocktail hour, some of those present shook hands, but in an unsystematic way, with no attempt to shake everyone's hand. Nobody shook their own hand, nobody shook hands with their spouse, and nobody shook hands with the same person more than once. During dinner, Halmos asked each of the nine other people present, including his wife, how many hands they had shaken. Under the given conditions, the possible answers range from 0 to 8. However, it turned out that each person gave a different answer: one person had not shaken any hand, one person had shaken exactly one hand, one person had shaken exactly two hands, and so on, up to one person who had shaken hands with all the others present, except their spouse, that is eight hands. Perform the following sequence of tasks:

A. Write a MiniZinc model `wifeHalmos.mzn` to find how many hands Halmos' wife shook.

B. Evaluate your model, writing her number of handshakes in the result table.

C. Add a constraint to the model in order to determine whether the number of handshakes by Halmos' wife in Task B is unique. If it is not unique, then report under Task B the first solution of each backend.

4 A Largest Permutation (pass/fail)

Find a permutation $[v_1, \ldots, v_n]$ of the integers 1 to n such that the sum of the products of all pairs of successive values v_i and v_{i+1} is maximal. Perform the following sequence of tasks:

A. Write a MiniZinc model called `largestPerm.mzn` in order to find such a permutation.

B. Evaluate your model from $n = 5$ to at most $n = 15$, by steps of 2, using any time-out of 30 to 60 CPU seconds, and giving the ***best-found*** solution upon a time-out.

   This problem can be solved in time polynomial in n but is a difficult benchmark for general-purpose solvers: you are ***not*** expected to find a model that scales up to $n = 25$ before timing out with such a short time-out, and our purpose ***only*** is to observe the different scalability of the various solving technologies and backends.

   In general, we usually do ***not*** expect proven optimality and are interested in comparing best-found solutions.

Hint: The maximum sum for $n = 5$ is 46, and you can gauge the progress by using the MiniZinc integrated development environment (IDE) or the `-a` option at the command-line interface (CLI) for displaying the intermediate feasible solutions to an optimisation problem.

5 Tile Packing (pass/fail)

Place n tiles of sizes 1×1, 2×2, ..., $n \times n$ inside a bounding rectangle of width w and height h such that no tiles are overlapping and $w \cdot h$ is minimal. Perform the following sequence of tasks:

A. Write a MiniZinc model called `tilePacking.mzn` in order to find such a tile packing, using the `diffn` predicate, which is specified in the MiniZinc Handbook and has an example in Figure 5.118.1 of the Global Constraint Catalogue).
Hint: Give the variables as tight domains as possible.

B. Evaluate your model from $n = 4$ to $n = 15$, by steps of 1, using any time-out of 30 to 60 CPU seconds, giving the best-found solution upon a time-out, and writing the tuple $\langle w, h, w \cdot h \rangle$ instead of just the objective value $w \cdot h$ in the result table. **Hint:** Use the `add_to_output` annotation for the MiniZinc variables corresponding to w and h.

C. A reflection or rotation of any (partial) solution results in a symmetrical (partial) solution. Likewise, reflecting or rotating any (partial) non-solution results in a symmetrical (partial) non-solution. An easy way to break these symmetries is to constrain the lower-left corner of the largest tile, which is of size $n \times n$, to be placed somewhere within the lower-left quadrant of the bounding rectangle. Add such a symmetry-breaking constraint γ to your model via the syntax `constraint symmetry_breaking_constraint(`γ`)`, re-run the evaluation of Task B, and discuss the performance impact this has for each backend, in terms of both solution quality and solution time. **Hint:** Avoid using the `div` function by rewriting a constraint such as `a < b div c`, where `a` and `b` are variables and `c` is a positive parameter, into `c * a < b`.

Trivia: The largest known optimal solution is for $n = 32$: it has a bounding rectangle of size 85×135 and took about 33.5 CPU days to find and prove optimal. Finding this solution requires much more sophisticated symmetry breaking as well as search heuristics.

Submission Instructions

In order to protect yourself against an unnecessary loss of points, use the following to-do list before submitting:

- Tackle ***each*** task of ***each*** problem, using (in order to accelerate the grading) the numbering and the ordering in which they appear in this assignment statement.
- Take the instructions of the source code of the demo report as a ***strict*** guideline for the structure and content of a model description, model evaluation, and task answer, and as an indication of the expected quality of content: write with the precision that you would expect from a textbook.
- You ***must*** use the MiniZinc experiment script explained in our cheatsheet: it conducts the experiments and generates a result table that can be automatically imported (rather than manually copied) into a LaTeX report, so that each time you change a model, it suffices to re-run the script and re-compile your report, without any tedious number copying!
- If a MiniZinc model does not compile and run error-free under backends of ***all*** the considered solving technologies, then obtain a teacher's approval in due time ***before*** submitting your report.

- ***Thoroughly*** proofread, spellcheck, and grammar-check the report, at least once per teammate, including the comments in ***all*** code. In case you are curious about technical writing: see the English Style Guide of UU, the technical-writing Check List & Style Manual of the Optimisation group, common errors in English usage, and common errors in English usage by native Swedish speakers.
- Match ***exactly*** the uppercase, lowercase, and layout conventions of any filenames and I/O texts imposed by the tasks, as we will process submitted source code automatically. However, do not worry when *Studium* appends a version number to the filenames when you make multiple submission attempts until the deadline.
- Do ***not*** rename any provided problem-specific skeleton model, for the same reason.
- Import all the MiniZinc models ***also*** into the report: for brevity, it is allowed to import only the lines after the copyright notice.
- Produce the report as a ***single*** file in ***PDF*** format; all other formats will be rejected.
- Remember that when submitting you implicitly certify (a) that your report and all its uploaded attachments were produced solely by your team, except where explicitly stated otherwise and clearly referenced, (b) that each teammate can individually explain any part starting from the moment of submitting your report, and (c) that your report and attachments are not freely accessible on a public repository.
- Submit (by only ***one*** of the teammates) the solution files (one report and all MiniZinc source code) ***without*** folder structure and ***without*** compression via *Studium*, whose clock may differ from yours, by the given ***hard*** deadline.

Grading Rules

If *all* the tasks have been tackled ***and*** *all* the requested models are in files with the imposed names, comments, and explanations exemplified in the demo report, ***then*** you *might* pass this assignment (read on), ***else*** you fail it. Furthermore:

- ***If*** *all* models *pass most* of our grading instances (by producing *correct* outputs under backends of *all* the considered solving technologies and (*near-*)*optimal* outputs under backends of *at least two* technologies in *reasonable* time under MiniZinc version 2.8.5 on a Linux computer of the IT department), ***and*** *all* models are good (in terms of comments and against the checklist of Topic 2), ***and*** *all* task answers are *mostly correct*, ***then*** you *do* pass this assignment and are *not* invited to the grading session.
- ***If*** *some* models *fail many* of our grading instances, ***or*** *some* models are flawed (in terms of comments and against the checklist of Topic 2), ***or*** the task answers have *many errors*, ***then*** you *are* invited to the grading session, at the end of which you are informed whether you pass or fail this assignment, a no-show leading to failure.