Modelling for Combinatorial Optimisation (1DL451)
Uppsala University – Autumn 2025
Assignment 1 (pass/fail)

— Deadline: 13:00 on Friday 14 November 2025 —

The scope of this assignment is Topics 1 to 2: you need ***not*** show any knowledge of subsequent topics. Read the **Report Instructions** and **Grading Rules** at the end of this document ***before*** tackling the following problem(s) upon reading them a first time. 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] 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 below 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 speed-tuning those models until your self-allocated time budget is up: this assignment is only pass/fail and we will grade only for correctness.

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:

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 a model; this applies to ***all*** the models of ***all*** the assignments.

B. Evaluate your model, writing the age of ***only*** the parent in the result table (see Section 3.2 of the cheatsheet for how to achieve that). Follow the instructions of Section C of the demo report for how to evaluate a model; this applies to ***all*** the models 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.

[1]Solo teams, except PhD students, may skip Problems 2 and 5.C, but are encouraged to try them nevertheless.

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 ***only*** the number of handshakes of Halmos' wife in the result table (see Section 3.2 of the cheatsheet for how to achieve that).

C. State which constraint to add 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 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 $n = 15$, 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 $n = 15$ 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, $\ldots$, $n \times n$ 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 decision 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 $w, h, w \cdot h$ instead of just the objective value $w \cdot h$ in the result table (see Sections 3.2 and 3.3 of the cheatsheet for how to achieve that).

C. A reflection or rotation of any (partial) solution results in a symmetrical (partial) solution. Likewise, a reflection or rotation of any (partial) non-solution results in a symmetrical (partial) non-solution. One way to break these symmetries is to constrain the lower-left corner of the $n \times n$ tile to be somewhere within the lower-left quarter of the $w \times h$ bounding rectangle (assuming the origin of its coordinate system is in its lower-left corner). 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 only for Gecode, and discuss the performance impact in terms of solution quality and solution time. Import this extended model into the report not for this Task C, but for Task A, with the symmetry-breaking constraint commented away for Tasks A and B.
Hint: Avoid using the `div` function on decision variables by rewriting a constraint such as `a < b div c`, where `a` and `b` are decision 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.

Report Instructions

1. You ***must*** fill in the provided skeleton report and skeleton model(s). You ***must*** inspect your model(s) using our checklist. See the demo report (LaTeX source) for generic instructions and for the expected quality of a report. The specific writing instructions are in comments that start with '`%%`' in the LaTeX source of the provided skeleton report and should be followed even when not using LaTeX. All running text should be black: before submitting, comment away line 15 (which typesets the placeholders in blue) and uncomment line 16 (which typesets them in black). In the provided MiniZinc skeleton model(s), the placeholders are '`...`' and the main model items are delimited by comments that start with '`%%`'.

2. Spellcheck the report and the comments in ***all*** models, in order to show both self-respect and respect for your readers. Thoroughly proofread the report, at least once per teammate, and ideally grammar-check it. In case you are curious about technical writing: see the English Style Guide of UU, the technical-writing Style Manual of the Optimisation research group, a list of common errors in English usage, and a list of common errors in English usage by native Swedish speakers.

3. 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.

4. Submit (by only ***one*** of the teammates) by the given ***hard*** deadline two files via *Studium*: your report as a ***PDF*** file (all other formats will be rejected) and a compressed folder in ***ZIP*** format with all source-code files (including all MiniZinc models, plus possibly a software pipeline for the project) and datafiles (only for the project).

Grading Rules (specific to Assignment 1)

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

- ***If*** *all* models ultimately produce *correct* solutions to *most* instances under backends of *all* considered solving technologies, ***and*** *all* models produce (*near-*)*optimal* solutions in *reasonable* time to *most* instances under backends of *at least two* technologies (under MiniZinc version 2.9.4 on a Linux computer of the IT department), ***and*** *all* models have *good enough* comments and explanations, ***and*** *all* task answers are *correct enough*, ***then*** you *pass* this assignment and are *not* invited to its grading session.

- ***If*** *some* models produce *incorrect* solutions to *too many* instances, ***or*** *some* models have *poor* comments or explanations, ***or*** *some* task answers have *severe errors*, ***then*** you *are* invited to the grading session, at the end of which you are informed whether you *pass* or *fail* this assignment. Each invited student who makes a no-show *fails* the assignment.