Modelling for Combinatorial Optimisation (1DL451) and Constraint Programming (1DL441) Uppsala University – Autumn 2018 Assignment 1: Warm-Up (pass/fail)

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

— Deadline: 15:00 on Friday 14 September 2018 —

Read the source code of the relevant demo report at `http://user.it.uu.se/~pierref/courses/COCP/demoReport` ***before*** tackling any assignment: it contains problem-independent instructions on what to hand in, as well as useful problem-independent indications on how to proceed. It is also strongly recommended to read the *Submission Instructions* and *Grading Rules* at the end of this document ***before*** attempting to address its tasks.

Combinatorial optimisation and the MiniZinc toolchain are intended for complex real-life problems, but we warm up slowly, with simple puzzles and simplified real-life problems. Model the following problems[1] using MiniZinc and document the performance of backends of all considered technologies, namely CP, LCG, MIP, SAT, and CBLS. If you make assumptions that are not part of a problem formulation, then state them clearly in the report and in the 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.

1 The Nine Children (pass/fail)

There is a person with nine children, with equally long gaps between the ages, expressed as integers, of any two consecutive 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 `nineChildren.mzn` in order to find the ages of the children and their parent.

B. Evaluate your model in the report.

C. How old are the children and their parent? Show how to use MiniZinc in order to determine whether this solution is unique. If it is not unique, then give the first solution of each backend.

[1] Solo teams may skip Problems 1 and 3, but are highly encouraged to try them nevertheless.

2 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 his or her own hand, nobody shook hands with his or her 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 his or her spouse, that is eight hands. Perform the following sequence of tasks:

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

B. Evaluate your model in the report, writing the found number of hands instead of 'sat' (for 'satisfiable') in the performance table.

C. Show how to use MiniZinc in order to determine whether the solution is unique. If it is not unique, then report under Task B the first solution of each backend.

3 Five Men and a Monkey (from Martin Gardner) (pass/fail)

Five men and a monkey were shipwrecked on a desert island, and they spent the first day gathering coconuts for food, piled them all up, and went to sleep. When they were all asleep, one man woke up and thought there might be a row in the morning about dividing the coconuts, so he decided to take his share. He divided the coconuts into five equal-sized piles. He had one coconut left over, gave it to the monkey, hid his pile, and put the rest back together. The next man woke up and did the same thing; he also had one coconut left over and gave it to the monkey. Eventually, all five men had done the same thing, one after the other, each taking a fifth of the remaining coconuts in the pile when he woke up, and each having one coconut left over for the monkey. In the morning, the monkey had five coconuts and the men divided what coconuts were left: they came out in five equal shares. Of course, each man knew there were coconuts missing, but each man was as guilty as the others, so they did not say anything. Perform the following sequence of tasks:

A. Write a MiniZinc model `fiveMenMonkey.mzn` in order to find the number of coconuts at the beginning.

B. Evaluate your model in the report, writing the found number of coconuts instead of 'sat' (for 'satisfiable') in the performance table.

C. Show how to use MiniZinc in order to determine whether the solution is unique. If it is not unique, then report under Task B the first solution of each backend.

D. What is the minimal solution? Show how to use MiniZinc in order to find it.

4 The 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 `largestPerm.mzn` in order to find such a permutation.

B. Evaluate your model in the report, from $n = 5$ to at most $n = 100$, by steps of 5, 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 $n = 100$ 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: Use the `-a` option for displaying the intermediate solutions to optimisation problems.

5 Square Packing (pass/fail)

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

A. Write a MiniZinc model `squarePacking.mzn` to find such a square packing, using the `diffn` predicate (see `http://www.minizinc.org/doc-latest/en/lib-globals.html#packing-constraints`). **Hint:** Give the variables as tight domains as possible.

B. Evaluate your model in the report, 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 performance table.

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 largest square, of size $n \times n$, to be placed in the lower-left quadrant of the bounding rectangle, that is $x[n] < w/2$ and $y[n] < h/2$. Add such a symmetry-breaking constraint to your model, re-run the evaluation of Task B, and also give the performance impacts 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` into `a * c < b`, which is valid when `c` is positive.

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

All task answers, other than source code, ***must*** be in a ***single*** report in ***PDF*** format; all other formats are rejected. Furthermore:

- Identify the team members and state the team number inside the report.
- Address ***each*** task of ***each*** problem, using the numbering and the ordering in which they appear in this assignment statement.
- Take the instructions of the relevant demo report at `http://user.it.uu.se/~pierref/courses/COCP/demoReport` as a ***strict*** guideline for the structure and content of a model description, model evaluation, and task answer in the report, as well as an indication of the expected quality of the report.
- If a teammate has taken a course on CP, then for Assignments 2 and 3 you are encouraged to show experiments with annotations: self-study and apply the slides of *Topic 8: Inference and Search for CP and LCG* (also see Section 2.6 in the *MiniZinc Handbook*).
- Write clear task answers, source code, and comments.
- Justify all task answers, except where explicitly not required.
- State any assumptions you make that are not in this document.
- Thoroughly proofread, spellcheck, and grammar-check your report.
- Upload all model files.
- Match exactly the uppercase, lowercase, and layout conventions of any filenames and I/O texts imposed by the tasks, as we will process your models automatically.
- Write a paragraph, which will not be graded, describing your experience with this assignment: Which aspects were too difficult or too easy? Which aspects were interesting or boring? This will help us improve the course in the coming years.
- 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.

Only ***one*** of the teammates submits the solution files (one PDF report with answers to ***all*** the tasks, and all model files), ***without*** folder structure and ***without*** compression, via the Student Portal, whose clock may differ from yours, by the given ***hard*** deadline.

Grading Rules

If ***all*** tasks have been seriously attempted and ***all*** requested models exist in files with the imposed names, have the comments exemplified in the relevant demo report, and produce correct outputs for some of our grading instances in reasonable time under version 2.1.7 (*not* 2.2.0) of MiniZinc on the Linux computers of the IT department, within the given time bounds (if any), then you *might* pass this assignment (read on), otherwise you fail it. Furthermore:

- If your models ***pass most*** of our grading instances ***and*** your task answers are ***mostly correct***, then you *do* pass this assignment and are not invited to the grading session.
- If your models ***fail many*** of our grading instances ***or*** your 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.