Modelling for Combinatorial Optimisation (1DL451) and Part 1 of Constraint Programming (1DL442)
Uppsala University – Autumn 2023
Report for Assignment n by Team t

Clara CLÄVER and Whiz KIDD

11th September 2023

This document shows the ingredients of a good assignment or project report for this (part of the) course. The LaTeX source code of this document exemplifies almost everything you need to know about LaTeX in order to typeset a professional-looking report (for this course). Start from that source code whether you use LaTeX or not, and delete everything irrelevant. The usage of LaTeX is *optional*, but highly recommended, for reasons that will soon become clear to those who have never used it before, for example upon enjoying the feature described at footnote 2. Any learning time of LaTeX is *outside* the time budget of this course, but will hugely pay off, if not in this course then in the next courses you take and when writing a thesis or other report.

Reading these blue instructions is an utter waste of time for the students who wish to waste time in writing a report or wish to lose points when we grade their report (or both). Address ***each*** task of ***each*** problem, using the numbering and ordering in which the problems and tasks appear in the assignment statement:

- For each task requiring a ***model***, follow the advice and structure of Section C.
- For each task requiring a model ***evaluation***, follow the structure of Section D.

Delete all unnecessary text in order to save trees (in case we print your report), suitably replace all other text in this document, and make all running text appear in black.

All experiments were run under Linux Ubuntu 18.04 (64 bit) on an Intel Xeon E5520 of 2.27 GHz, with 4 processors of 4 cores each, with a 70 GB RAM and an 8 MB L3 cache (a ThinLinc computer of the IT department).[1] (If different hardware was used for different problems, then justify this and replicate a paragraph like this one within each relevant report part or section.)

Problem 1
The Minimal Magic Square Problem

A Heading for Task A

(Write your answer for Task A here. Remember that this document is a guideline, so you should change its text when appropriate.)

[1] Hint: Under Linux, do `lscpu` to find this information. Under macOS, you find this information via “About This Mac” in the Apple menu.

B The Problem

(You need ***not*** describe a problem specified in an assignment statement.) The *Minimal Magic Square* problem is about finding a magic square (as defined in Topic 5), of a given order, with minimal sum of the corner elements.

C Model

Our model is given in Listing 1: it has the prescribed comments as per the scope of the assignment, it has the imposed name `minMagicSquare.mzn`, and it is uploaded to Studium. We understand that we will lose points if some of all this is not true (say because we just copied this sentence without paying attention to it). A MiniZinc [1] model ***must*** include comments that give each of the following explanations, provided the scope of the assignment goes at least until the indicated topic:

- the semantics of the parameters (which are in the possibly provided skeleton model);
- the semantics of derived parameters (that are not in the possibly provided skeleton model);
- the semantics of the decision variables;
- the semantics of the problem constraints enforced by the choice of decision variables;
- the semantics of the redundant decision variables (Topic 4), and specify (non-)mutuality;
- the semantics of the channelling constraints (Topic 4), and specify one-way or two-way;
- the semantics of the problem constraints not enforced by the choice of decision variables;
- the semantics of the implied constraints (Topic 4);
- the semantics of the symmetry-breaking constraints (Topic 5);
- a paraphrase of the objective and possibly of the objective function;
- the importance of the inference annotations (Topic 8);
- a paraphrase of the search annotation (Topic 8).

The quality of comments in the model is considered while grading. You ***must*** include in the report your model ***and*** upload it as a separate file to Studium (so that we can run it).

Listing 1: A MiniZinc model for the Minimal Magic Square problem

```
%% Parameters:
int: n; % width (and height) of the magic square

%% Derived parameters:
set of int: N = 1..n; % index range for the rows and columns
% The magic sum, for each row, column, and major diagonal:
int: magicSum = sum(1..n*n) div n;

%% Decision variables:
% Magic[r,c] = value at row r and column c of the magic square:
array[N,N] of var 1..n*n: Magic;
```

```
% No problem constraints are enforced by this choice of variables.
% (See the slides for examples)

%% Redundant decision variables:
% None
% (See the slides for examples, marked as mutual or non-mutual)

%% Channelling constraints:
% None
% (See the slides for examples, marked as 1-way or 2-way)

%% Problem constraints:
% All values in the magic square are distinct:
constraint all_different(Magic) :: domain_propagation; % crucial inference annotation!
% Each row sums up to the magic sum:
constraint forall(r in N)(sum(Magic[r,..]) = magicSum);
% Each column sums up to the magic sum:
constraint forall(c in N)(sum(Magic[..,c]) = magicSum);
% The major down-diagonal sums up to the magic sum:
constraint sum([Magic[    i,i] | i in N])  = magicSum;
% The major up-diagonal sums up to the magic sum:
constraint sum([Magic[n+1-i,i] | i in N])  = magicSum;

%% Implied constraints:
% None
% (See the slides for examples)

%% Symmetry-breaking problem constraints:
% Break the 3 rotation symmetries and 4 reflection symmetries:
constraint symmetry_breaking_constraint(Magic[1,1] < Magic[1,n] /\ Magic[1,n] < Magic[n,1] /\ Magic[1,1] < Magic[n,n]);

solve
  %% Search strategy:
  % Search (1) on the four corners -- which are subject to the
  % objective function (below) and more constraints (diagonals and
  % symmetry) than the other cells -- starting in the lower halves
  % of bisections of their domains, due to the minimisation and
  % the arithmetic in the constraints; and (2) similarly for the
  % remaining cells, starting with those on the major diagonals:
  :: seq_search([
       int_search([Magic[i,j] | i,j in {1,n}], input_order,indomain_split),
       int_search(Magic,occurrence,indomain_split)
     ])
  %% Objective and objective function:
  % Minimise the sum of the four corners of the magic square:
  minimize sum([Magic[i,j] | i,j in {1,n}]);
```

Backend	Gecode		CP-SAT		Gurobi		Yuck		PicatSAT	
`n`	`obj`	time	`obj`	time	`obj`	time	`obj`	time	`obj`	time
3	**20**	**422**	**20**	954	**20**	1268	**20**	t/o	**20**	1262
4	**34**	**372**	**34**	680	**34**	1210	**34**	t/o	**34**	8297
5	**26**	68100	**26**	t/o	**26**	**46645**	36	t/o	27	t/o
6	–	t/o	–	t/o	**26**	**65681**	–	t/o	39	t/o

Table 1: Results for our model of Minimal Magic Square, which is a minimisation problem. In the 'time' column: if the reported time is less than the time-out (600,000 milliseconds here), then the reported objective value in the '`obj`' column was *proven* optimal; else the timing out is indicated by 't/o' and the reported objective value is either the best one found but *not* proven optimal before timing out, or '–' indicating that no feasible solution was found before timing out. Boldface indicates the best performance (time or objective value) on each row.

```

%% Pretty-print solutions:
output ["Magic sum: \(magicSum) \nCorner sum: \(sum([Magic[i,j] |
  i,j in {1,n}])) \n"] ++ [show2d(Magic)];
```

Efficiency. (For each violation of a piece of advice in the checklists at `https://user.it.uu.se/~pierref/courses/COCP/demoReport/checklist.pdf` of Topics 2 and 3 (if Topic 3 is in the announced scope of the current assignment): how do you argue that it does not matter? For example, for a reification or a `where` clause involving variables, does a profiled compilation reveal numbers of generated variables and constraints that you argue to be acceptable; or is the solving time comparable to the one of a violation-free reformulation that you give?) The model features no violations of any pieces of advice in the checklists of the final slides of Topics 2 and 3, and we understand that we will lose points if this is not true (say because we just copied this sentence without paying attention to it).

D Evaluation

(Give a table of experiment results for the chosen backends and answer ***all*** the questions below.) Table 1 gives the results for various values of parameter `n` in our model.[2] The time-out was 600,000 milliseconds.

Which backends win overall, and how do you draw that conclusion? We observe that Gurobi wins overall, as it is the only one not to time out for the largest chosen instance, with $\mathtt{n} = 6$.

Which backends scale best, and how do you draw that conclusion? Gurobi and PicatSAT scale best, as they are the only ones to establish feasibility for $\mathtt{n} = 6$, even though the PicatSAT objective value of 39 at time-out is far above the minimum of 26 actually proven by Gurobi.

[2] You ***must*** use the script of `https://user.it.uu.se/~pierref/courses/COCP/assignments/cheatsheet.pdf`: it conducts the experiments and generates a result table (see the LaTeX source code of Table 1) that is automatically imported (rather than manually copied) into your report, so each time you change the model, it suffices to re-run that script and re-compile your report, without any tedious number copying!

How do the backends scale, and how do you draw that conclusion? On the small instances, with $\mathtt{n} \leq 4$, Gecode wins, narrowly defeating CP-SAT followed by Gurobi and PicatSAT. Starting from the medium instance, with $\mathtt{n} \geq 5$, Gurobi clearly wins, with only Gecode also not timing out for $\mathtt{n} = 5$, with CP-SAT finding but not proving the minimum for $\mathtt{n} = 5$, and with PicatSAT missing the minimum already by one unit for $\mathtt{n} = 5$.

Does the difficulty of instances monotonically increase with their size, and how do you draw that conclusion? PicatSAT is the only backend where $\mathtt{n} = 4$ is harder than the smaller $\mathtt{n} = 3$. Observe in the Gurobi column that the minimum objective value also does not monotonically increase with `n`.

How suitable is local search compared to systematic search, and how do you draw that conclusion? Yuck always times out, because local search can by construction not prove minima on problems, such as here, where the trivial lower bound (namely $10 = 1 + 2 + 3 + 4$ here) on the objective value is not feasible; it finds the minima for $\mathtt{n} \leq 4$, but is far above the minimum for $\mathtt{n} = 5$ and cannot even establish feasibility for $\mathtt{n} = 6$ before timing out.

Are there any contradictions between the results? No results are contradictory. All proven optima are the same.

Are there any occurrences of 'ERR' within the results generated by the experiment script? (If so, then first try and troubleshoot on your own by running the incriminated backend manually (within the IDE or at the command line by using the `--solver` flag of the `minizinc` command) and interpreting the error message. If you cannot resolve the error, then you ***must*** state here for each occurrence of 'ERR' when and how you received a teacher's ***prior*** approval to include it, and you ought to make an error report in the final section.) No occurrences of 'ERR' were generated by the experiment script.

Problem 2
The XYZ Problem

A ...

...

B ...

...

Feedback to the Teachers

(Please write a paragraph, which will ***not*** be graded, describing your experience with this assignment: which aspects were too difficult or too easy, and which aspects were interesting or boring? This will help us improve the course for the next year.)

Error Report

(Your model ***must*** compile and run error-free under backends of ***all*** the considered solving technologies, unless you have a teacher's ***prior*** approval to upload an error report here.)

largecumulative.mzn We secured the head teacher's oral permission on 7 September 2023 to upload this error report: for the larger instances, fzn-oscar-cbls [which is currently not used in this course, and which was fixed after this error report] crashes with the following exception, which seems to be caused by the JVM not being allocated enough heap space:

```
>./fzn-oscar-cbls -s -t 600 /tmp/tmp.fzn
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
  at scala.collection.mutable.FlatHashTable$class.growTable(FlatHashTable.scala:217)
  at scala.collection.mutable.FlatHashTable$class.addEntry(FlatHashTable.scala:159)
  at scala.collection.mutable.HashSet.addEntry(HashSet.scala:40)
  at scala.collection.mutable.FlatHashTable$class.addElem(FlatHashTable.scala:139)
  at scala.collection.mutable.HashSet.addElem(HashSet.scala:40)
  at scala.collection.mutable.HashSet.$plus$eq(HashSet.scala:59)
  at oscar.cp.scheduling.constraints.EnergeticReasoning$$anonfun$computeIntervals$1$$anonfun$apply$mcVI$sp$2.apply(EnergeticReasoning.scala:127)
  at oscar.cp.scheduling.constraints.EnergeticReasoning$$anonfun$computeIntervals$1$$anonfun$apply$mcVI$sp$2.apply(EnergeticReasoning.scala:126)
  at scala.collection.TraversableLike$WithFilter$$anonfun$foreach$1.apply(TraversableLike.scala:778)
  at scala.collection.mutable.HashSet.foreach(HashSet.scala:78)
  at scala.collection.TraversableLike$WithFilter.foreach(TraversableLike.scala:777)
  at oscar.cp.scheduling.constraints.EnergeticReasoning$$anonfun$computeIntervals$1.apply$mcVI$sp(EnergeticReasoning.scala:126)
  at oscar.cp.scheduling.constraints.EnergeticReasoning$$anonfun$computeIntervals$1.apply(EnergeticReasoning.scala:126)
  at oscar.cp.scheduling.constraints.EnergeticReasoning$$anonfun$computeIntervals$1.apply(EnergeticReasoning.scala:126)
  at scala.collection.mutable.HashSet.foreach(HashSet.scala:78)
  at oscar.cp.scheduling.constraints.EnergeticReasoning.computeIntervals(EnergeticReasoning.scala:126)
```

References

[1] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: Towards a standard CP modelling language. In Christian Bessière, editor, *CP 2007*, volume 4741 of *LNCS*, pages 529–543. Springer, 2007. The MiniZinc toolchain is available at `https://www.minizinc.org`.

More LaTeX and Technical Writing Advice

Unnumbered itemisation (only to be used when the order of the items does *not* matter):[3]

- Unnumbered displayed formula:
$$E = m \cdot c^2$$

- Numbered displayed formula, which is cross-referenced somewhere:
$$E = m \cdot c^2$$

- Formula — the same as formula (B) — spanning more than one line:
$$\begin{aligned} &E \\ &= m \cdot c^2 \end{aligned}$$

Numbered itemisation (only to be used when the order of the items *does* matter):

1. First do this.
2. Then do that.
3. If we are not finished, then go back to Step 2, else stop.

Tables and elementary mathematics are typeset as exemplified in Table 2; see `http://tug.ctan.org/info/short-math-guide/short-math-guide.pdf` for many more details.

Use `\mathit{...}` in mathematical mode for each multiple-letter identifier in order to avoid typesetting the identifier like the product of single-letter ones. For example, note the typographic difference between the identifier WL, obtained through `$\mathit{WL}$`, and the product WL, where there is a small space between the W and the L, obtained through `$WL$`.

Do *not* use programming-language-style lower-ASCII notation (such as ! for negation, && for conjunction, || for disjunction, and the equality sign = for assignment) in algorithms or formulas (but rather use $\neg$ or **not**, $\wedge$ or & or **and**, $\vee$ or **or**, and $\leftarrow$ or $:=$, respectively), as this testifies to a very strong confusion of concepts.

Figures can be imported with `\includegraphics` or drawn inside the LaTeX source code using the highly declarative notation of the `tikz` package: see Figure 1 for sample drawings. It is perfectly acceptable in this course to include scans or photos of drawings that were carefully done by hand.

Algorithms can be typeset as pseudo-code as exemplified in Algorithm 1: study its LaTeX source code.

If you are not sure whether you will stick to your current choice of notation or terminology, then introduce a new (possibly parametric) command. For example, upon

```
\newcommand{\Cardinality}[1]{\left\lvert#1\right\rvert}
```

the formula `$\Cardinality{S}$` typesets the cardinality of set S as $|S|$ with autosized vertical bars and proper spacing, but upon changing the definition of that parametric command to

```
\newcommand{\Cardinality}[1]{\# #1}
```

[3]Use footnotes very sparingly, and note that footnote pointers are *never* preceded by a space and always glued immediately *behind* the punctuation, if there is any.

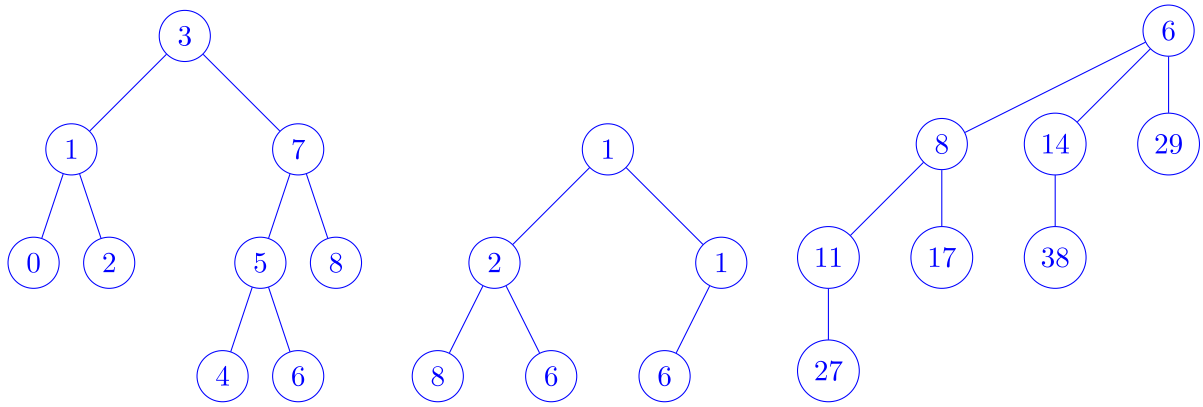

Figure 1: A binary search tree (on the left), a binary min-heap (in the middle), and a binomial tree of rank 3 (on the right).

function $f(n)$
if $n < 0$ **then** // optional comment
    $n := -2 \cdot n$ // optional comment
else // $n \geq 0$
    $n := 3 \cdot n$
while $n > 0$ **do** // optional comment
    $n := n - 1$
return n

Algorithm 1: Silly algorithm

and recompiling, the formula `$\Cardinality{S}$` typesets the cardinality of set S as $\#S$. Similarly, upon

```
\newcommand{\MiniZinc}{\textit{Mini\-Zinc}}
```

the text `\MiniZinc\` typesets into *MiniZinc*, hyphenation being only possible in the middle, but upon changing the definition of that non-parametric command to

```
\newcommand{\MiniZinc}{\textsc{Mini\-Zinc}}
```

and recompiling, the text `\MiniZinc\` typesets into MiniZinc. You can thus obtain an arbitrary number of changes in the document with a *constant*-time change in its source code, rather than having to perform a *linear*-time find-and-replace operation within the source code, which is painstaking and error-prone. The imported file `macros.tex` has a lot of useful predefined commands about mathematics, CP, Gecode, modelling, MiniZinc, and algorithms.

Use commands on positioning (such as `\hspace`, `\vspace`, and `\noindent`) and appearance (such as `\small` for reducing the font size, and `\textit` for italics) very sparingly, and ideally only in (parametric) commands, as the very idea of mark-up languages such as LaTeX is to let the class designer (usually a trained professional typesetter) decide on where things appear and how they look. For example, `\emph` (for emphasis) compiles (outside italicised environments, such as `theorem`) into *italics* under the `article` class used for this document, but it may compile into **boldface** under some other class.

If you do not (need to) worry about *how* things look,
then you can fully focus on *what* you are trying to express!

Note that *no* absolute numbers are used in the LaTeX source code for any of the references inside this document. For ease of maintenance, `\label` is used for giving a label to something that is automatically numbered (such as an algorithm, equation, figure, footnote, item, line, part, section, subsection, or table), and `\ref` is used for referring to a label. An item in the bibliography file is referred to by `\cite` instead. Upon changing the text, it suffices to recompile, once or twice, and possibly to run BibTeX again, in order to update all references consistently.

Always write `Table~\ref{tab:maths}` instead of `Table \ref{tab:maths}`, by using the non-breaking space (which is typeset as the tilde ~) instead of the normal space, because this avoids that a cross-reference is spread across a line break, as for example in "Table 2", which is considered poor typesetting.

The rules of English for how many spaces to use before and after various symbols are given in Table 3. Beware that they may be very different from the rules in your native language.

☞ Feel free to report to the head teacher any other features that you would have liked to see discussed and exemplified in this template document.

Topic	LaTeX code	Appearance
Greek letter	`$\Theta,\Omega,\epsilon$`	Θ, Ω, ϵ
multiplication	`$m \cdot n$`	$m \cdot n$
division	`$\frac{m}{n}, m \div n$`	$\frac{m}{n}, m \div n$
rounding down	`$\left\lfloor n \right\rfloor$`	$\left\lfloor n \right\rfloor$
rounding up	`$\left\lceil n \right\rceil$`	$\left\lceil n \right\rceil$
binary modulus	`$m \bmod n$`	$m \bmod n$
unary modulus	`$m \equiv n \mod \ell$`	$m \equiv n \mod \ell$
root	`$\sqrt{n},\sqrt[3]{n}$`	$\sqrt{n}, \sqrt[3]{n}$
exponentiation, superscript	`$n^{i}$`	n^{i}
subscript	`$n_{i}$`	n_{i}
overline	`$\overline{n}$`	$\overline{n}$
base 2 logarithm	`$\lg n$`	$\lg n$
base b logarithm	`$\log_b n$`	$\log_b n$
binomial	`$\binom{n}{k}$`	$\binom{n}{k}$
sum	`\[\sum_{i=1}^n i\]`	$$\sum_{i=1}^n i$$
numeric comparison	`$\leq,<,=,\neq,>,\geq$`	$\leq, <, =, \neq, >, \geq$
non-numeric comparison	`$\prec,\nprec,\preceq,\succeq$`	$\prec, \nprec, \preceq, \succeq$
extremum	`$\min,\max,+\infty,\bot,\top$`	$\min, \max, +\infty, \bot, \top$
function	`$f\colon A\to B,\circ,\mapsto$`	$f\colon A\to B, \circ, \mapsto$
sequence, tuple	`$\langle a,b,c \rangle$`	$\langle a,b,c \rangle$
set	`$\{a,b,c\},\emptyset,\mathbb{N}$`	$\{a,b,c\}, \emptyset, \mathbb{N}$
set membership	`$\in,\not\in$`	$\in, \not\in$
set comprehension	`$\{i \mid 1 \leq i \leq n\}$`	$\{i \mid 1 \leq i \leq n\}$
set operation	`$\cup,\cap,\setminus,\times$`	$\cup, \cap, \setminus, \times$
set comparison	`$\subset,\subseteq,\not\supset$`	$\subset, \subseteq, \not\supset$
logic quantifier	`$\forall,\exists,\nexists$`	$\forall, \exists, \nexists$
logic connective	`$\land,\lor,\neg,\Rightarrow$`	$\land, \lor, \neg, \Rightarrow$
logic	`$\models,\equiv,\vdash$`	$\models, \equiv, \vdash$
miscellaneous	`$\&,\#,\approx,\sim,\ell$`	$\&, \#, \approx, \sim, \ell$
dots	`$\ldots,\cdots,\vdots,\ddots$`	$\ldots, \cdots, \vdots, \ddots$
dots (context-sensitive)	`$1,\dots,n; 1+\dots+n$`	$1,\dots,n; 1+\dots+n$
parentheses (autosizing)	`$\left(m^{n^k}\right),(m^{n^k})$`	$\left(m^{n^k}\right),(m^{n^k})$
identifier of > 1 character	`$\mathit{identifier}$`	$\mathit{identifier}$
hyphen, n-dash, m-dash, minus	`-, --, ---, $-$`	-, –, —, $-$

Table 2: The typesetting of elementary mathematics. Note very carefully when italics are used by LaTeX and when not, as well as all the horizontal and vertical spacing performed by LaTeX.

		number of spaces after	
		0	1
number of spaces before	0	/ -	, : ; . ! ?)] } ’ ” %
	1	([{ ‘ “	– (n-dash) — (m-dash)

Table 3: Spacing rules of English