Introduction to MiniZinc

COMP3608 - Intelligent Systems

Solving optimisation problems and CSPs

- As we discussed in the lecture, there are conventional algorithms to solve optimisation and constraint-satisfaction problems
- In addition, recall that we often write such problems in a “canonical” format
- There are domain-specific languages (DSLs) that allow us to easily convert the formal statement of the problems to code that can be used to find solutions
  - These DSLs then piggyback off of solvers designed for specific problem types

MiniZinc

- MiniZinc is one such language
- Provides high-level primitives to allow us to solve optimisation problems and CSPs
- Compiles down into another language called FlatZinc that is then fed into a solver that actually does the heavy-lifting

Fundamental MiniZinc features

- There are several key features of MiniZinc we will be using to develop our models
  - Types
  - Parameters
  - Decision Variables
  - Arithmetic Expressions
  - Constraint Expressions
- All MiniZinc statements must end in a semicolon!
- All MiniZinc models are stored in .mzn files the describe our problem
  - To help make our code more modular, we can store the model structure in a .mzn file and the data in a .dzn file

Types in MiniZinc

- Integers - denoted *int* or range *1..n*
  - NB *1..n* is short hand for the set of integers between 1 and n inclusive
- Real numbers - approximated by *float*
  - NB: *1.0..f* is shorthand for the set of all real numbers between 1.0 and f inclusive
- Booleans - *bool*
- Fixed strings - *strings*
- Arrays
- Sets

Variables in MiniZinc

- Two types of variables in MiniZinc
  - Parameters
  - Decision Variables
- Parameters, in the context of MiniZinc, store our data.
  - E.g. the max number of screws available
- Decision Variables are what we are trying to find, i.e. they represent our decisions to optimise on or ensure that constraints are satisfied
  - E.g. the number of glass doors to manufacture or the placement of a queen on a chessboard

Parameter Syntax

- MiniZinc models describe parameters using the following syntax:
  - <type> : <varname> [= <expr>];
- <type> - gives the type of the parameter
- <varname> - gives the name of the variable
- <expr> - is some optional expression or value. Note that can defer giving a value to a parameter by providing a data file (.dzn)

Parameter Syntax - Examples

```
int: capacity = 10;
float: coeff1 = 2.3;
int: max_size;
array[1..3] of float: coeffs = [1.0, 2.0, 3.0];
```

Decision Variable Syntax

- Decision Variable syntax is similar to parameter syntax, except
  - We need to use the *var* keyword to specify to MiniZinc that we are declaring a decision variable
  - Decision variable values are decided by the algorithm. They are the holes for the algorithm to fill. We do not specify a value for a decision variable. Remember the purpose of optimisation is to find values for the decision variables
- *var* <type> : <varname>;

Decision Variable Syntax - Examples

```
var 0..1: x_1;
var int: num_doors;
var float: prop1;
array[1..10] of var 0..1: x;
```

Arithmetic Expressions

- Just like in most programming languages, you are free to use variables and values to construct arithmetic expressions
    - We can use a combination of parameters and decision variables in the same expression
    - In fact, we need to express constraints and objectives!

Constraints

- MiniZinc allows us to encode constraints as conditional expressions using less than, greater than, equal, etc… operators

  - Also, special functions such as *alldifferent* that forces all decision variables used as arguments to take on different values

Constraints - Examples

```
constraint 8 * x_1 + 5 * x_2 + 3 * x_3 <= 18;
```

Constraints - Examples

```
constraint 8 * x_1 + 5 * x_2 + 3 * x_3 <= 18;
```

arithmetic expression

Constraints - Examples

```
constraint 8 * x_1 + 5 * x_2 + 3 * x_3 <= 18;
```

comparison operator

Constraint Conjunction

- Often, we want more that one constraint to hold
  - We want to apply the boolean and to the constraints
- To do this we have two options:
  - We simply list each constraint separately in our models
  - We use the connection operator (/\)

Constraint Conjunction - Example

```
constraint 8 * x_1 + 5 * x_2 + 3 * x_3 <= 18;
constraint x_1 <= x_2;
```

is equivalent to

```
constraint (8 * x_1 + 5 * x_2 + 3 * x_3 <= 18) /\ (x_1 <= x_2);
```

Constraint Conjunction - Example

```
constraint 8 * x_1 + 5 * x_2 + 3 * x_3 <= 18;
constraint x_1 <= x_2;
```

is equivalent to

```
constraint (8 * x_1 + 5 * x_2 + 3 * x_3 <= 18) /\ (x_1 <= x_2);
```

We use the first as it is clearer to read

Constraint Disjunction

- Can also apply disjunction (logical or) to constraints using the $\vee$ operator.
- We would not be using this much

Model Solving

- Once we have specified parameters, decision variables, and constraints we have specified most of the model
- Next we need to specify what it means to solve a model. We have three options:
  - maximize - mazimize some function of the decision variables
  - minimize - minimize some function of the decision variables
  - satisfy - simply find a configuration of the decision variables that respect the constraints

Model Solving - Syntax

- maximize - *solve maximize* <arithmetic expression>;
- minimize - *solve minimize* <arithmetic expression>;
- satisfy - *solve satisfy*;

Putting it all together

- Let’s put a simple MiniZinc model together by formalising a model for a problem and the translating that into MiniZinc!

Example Problem

3.1-9. The Primo Insurance Company is introducing two new product lines: special risk insurance and mortgages. The expected profit is $5 per unit on special risk insurance and $2 per unit on mortgages.

Management wishes to establish sales quotas for the new product lines to maximize total expected profit. The work requirements are as follows:

	Work-Hours per Unit		
Department	**Special Risk**	**Mortgage**	**Work-Hours Available**
Underwriting	3	2	2400
Administration	0	1	800
Claims	2	0	1200

Example Problem - Characterise our objective

3.1-9. The Primo Insurance Company is introducing two new product lines: special risk insurance and mortgages. The expected profit is $5 per unit on special risk insurance and $2 per unit on mortgages.

Management wishes to establish sales quotas for the new product lines to maximize total expected profit. The work requirements are as follows:

Department	Work-Hours per Unit		Work-Hours Available
	Special Risk	Mortgage	
Underwriting	3	2	2400
Administration	0	1	800
Claims	2	0	1200

- If we let
  - x_1 be the number the number of units of special risk insurance
  - x_2 be the number the number of units of mortgages
- then we hope to maximise $f(x_1, x_2) = 5x_1 + 2x_2$

Example Problem

3.1-9. The Primo Insurance Company is introducing two new product lines: special risk insurance and mortgages. The expected profit is $5 per unit on special risk insurance and $2 per unit on mortgages.

Management wishes to establish sales quotas for the new product lines to maximize total expected profit. The work requirements are as follows:

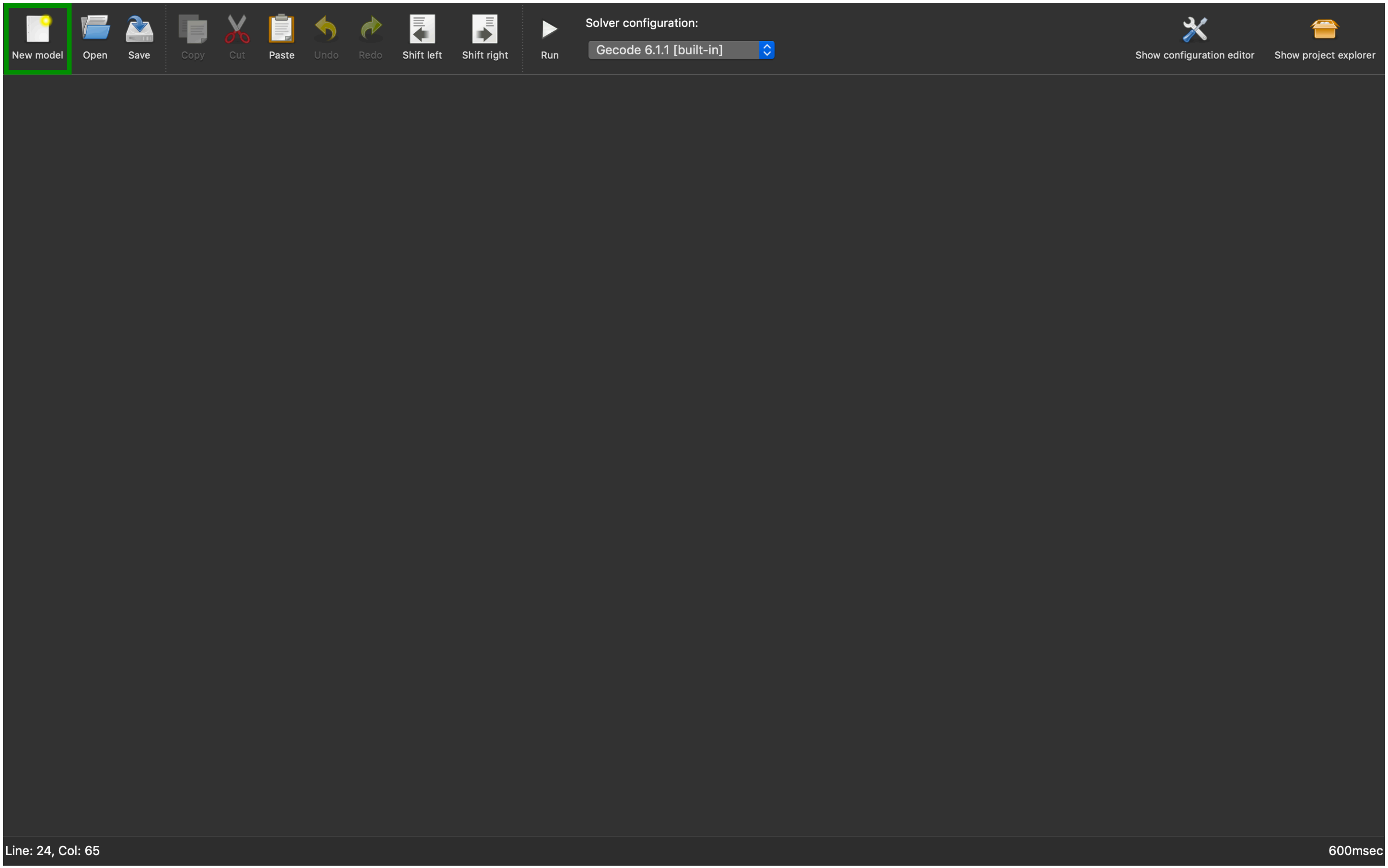
Department	Work-Hours per Unit: Special Risk	Work-Hours per Unit: Mortgage	Work-Hours Available
Underwriting	3	2	2400
Administration	0	1	800
Claims	2	0	1200

constraint per department

Example Problem - Constraints

- Our constraints are
  - $3x_1 + 2x_2 \leq 2400$
  - $0x_1 + 1x_2 \leq 800 \implies x_2 \leq 800$
  - $2x_1 + 0x_2 \leq 1200 \implies 2x_1 \leq 1200 \implies x_1 \leq 600$
  - $x_1, x_2 \geq 0$
  - $x_1, x_2 \in \mathbb{Z}$

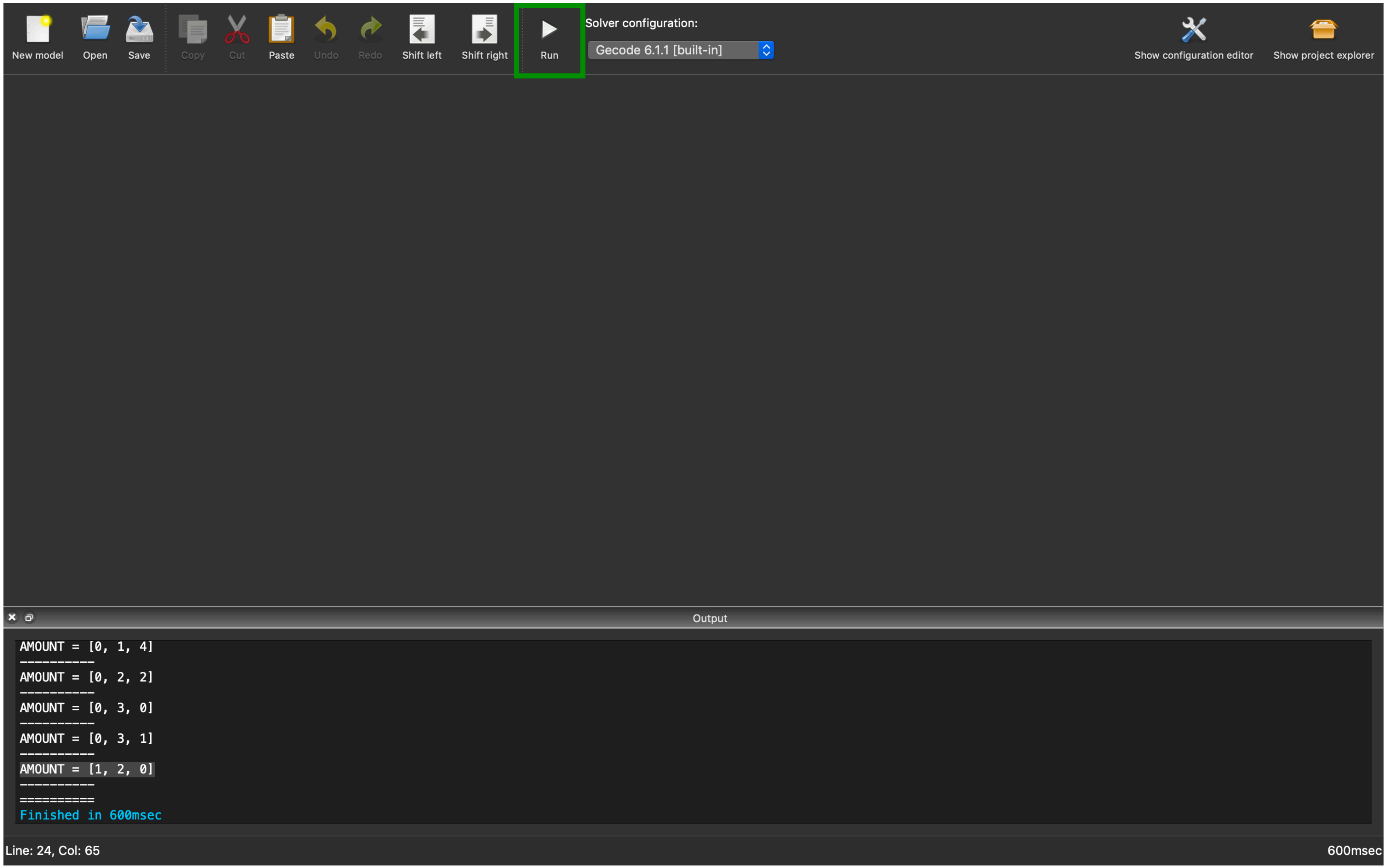
MiniZinc Usage



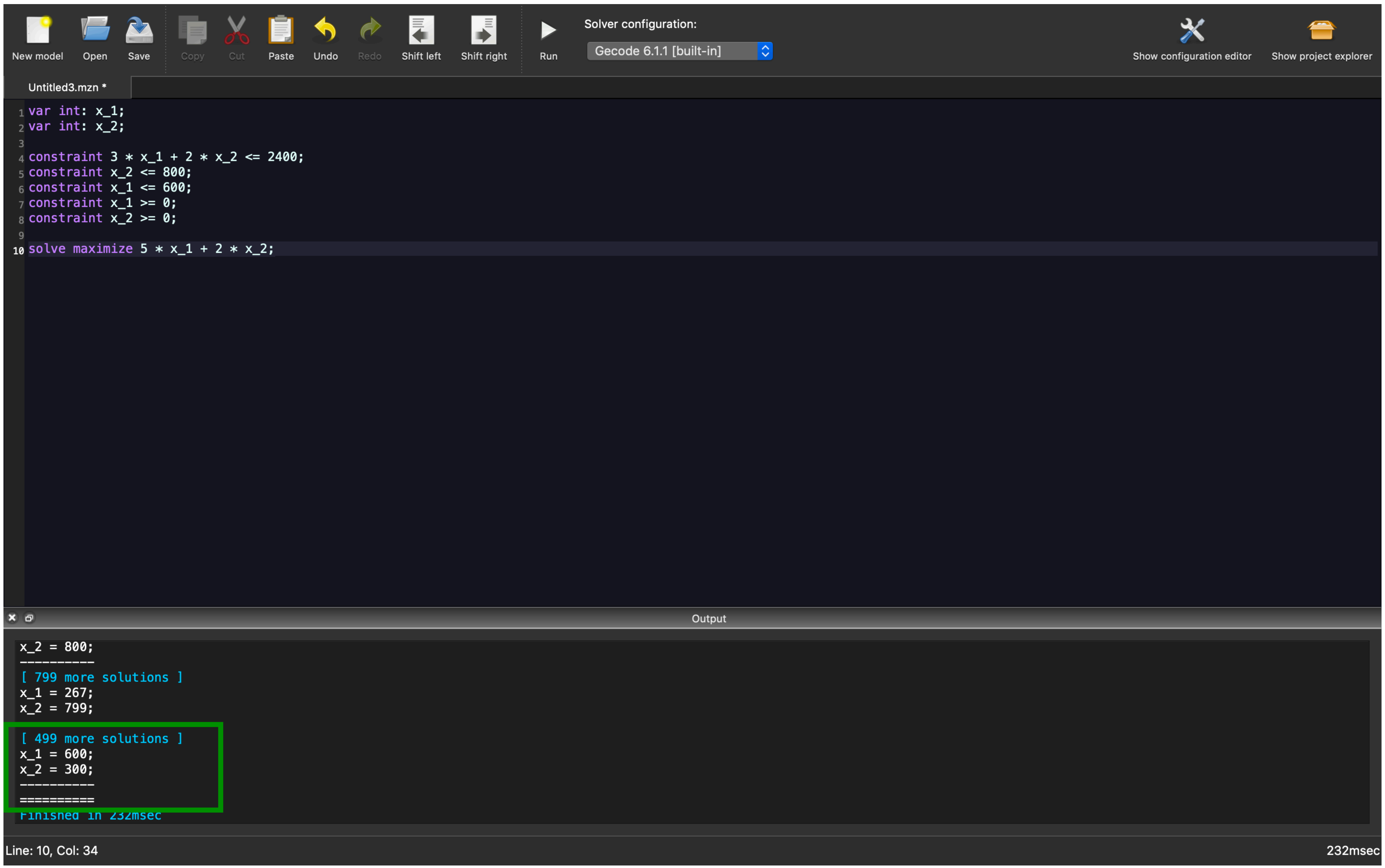
MiniZinc Model for Problem

```
var int: x_1;
var int: x_2;

constraint 3 * x_1 + 2 * x_2 <= 2400;
constraint x_2 <= 800;
constraint x_1 <= 600;
constraint x_1 >= 0;
constraint x_2 >= 0;

solve maximize 5 * x_1 + 2 * x_2;
```

Running MiniZinc Model



Running MiniZinc Model



Try this one on your own!

3.4-8. Ralph Edmund loves steaks and potatoes. Therefore, he has decided to go on a steady diet of only these two foods (plus some liquids and vitamin supplements) for all his meals. Ralph realizes that this isn't the healthiest diet, so he wants to make sure that he eats the right quantities of the two foods to satisfy some key nutritional requirements. He has obtained the nutritional and cost information shown at the top of the next column.

Ralph wishes to determine the number of daily servings (may be fractional) of steak and potatoes that will meet these requirements at a minimum cost.

Ingredient	Grams of Ingredient per Serving		Daily Requirement (Grams)
	Steak	Potatoes	
Carbohydrates	5	15	≥ 50
Protein	20	5	≥ 40
Fat	15	2	≤ 60
Cost per serving	\$8	\$4	

A Slightly harder problem

12.1-2* A young couple, Eve and Steven, want to divide their main household chores (marketing, cooking, dishwashing, and laundering) between them so that each has two tasks but the total time they spend on household duties is kept to a minimum. Their efficiencies on these tasks differ, where the time each would need to perform the task is given by the following table:

	Time Needed per Week			
	Marketing	**Cooking**	**Dishwashing**	**Laundry**
Eve	4.5 hours	7.8 hours	3.6 hours	2.9 hours
Steven	4.9 hours	7.2 hours	4.3 hours	3.1 hours

O.o

Decisions, decisions, decisions

- When confronted by a scenario like this, always start with asking yourself about what decisions you need to make.
- Rereading the problem description we have 8 decisions to make:
  - Does Steve do laundry?
  - Does Eve do laundry?
  - Does Steve do the cooking?
  - Does Eve do the cooking?
  - etc …

Binary Integer Programming

- Cases where we need to make a bunch of yes or no decisions are called BIPs (Binary Integer Programming) problems

- We say that each decision variable can take on a value of 0 (no) or 1(yes)

- We will look at some modelling tricks now

BIP modelling tricks

- Suppose that we have decisions $x_1, x_2, \ldots x_n$ decisions to make. We can only choose to do only $m < n$ things from that list.

- How can we use a constraint to make sure that we do only m?

  - Hint, recall that $x_1, x_2, \ldots x_n \in \{0,1\}$

BIP modelling tricks

- Suppose that we have decisions $x_1, x_2, \ldots x_n$ decisions to make. We can only choose to do only $m < n$ things from that list.
- How can we use a constraint to make sure that we do only m?
  - Hint, recall that $x_1, x_2, \ldots x_n \in \{0,1\}$
  - We add $\sum_{i=1}^{n} x_i = m$ as a constraint

Lets’ model our scenario

- L_s - Steve does the laundry
- L_E - Eve does the laundry
- C_s - Steve does the cooking
- C_E - Eve does the cooking
- D_s - Steve does the dishwashing
- D_E - Even does the dishwashing
- M_S - Steve does the marketing
- M_E - Even does the marketing
- where all of the above variables are elements of $\{0,1\}$
- What are we trying to minimise? Try coming up with the loss function

Constraints

- We need to ensure that only one does each chore.
- Hence, we need to ensure that
  - $L_S + L_E = 1$
  - $C_S + C_E = 1$
  - $D_S + D_E = 1$
  - $M_S + M_E = 1$

Constraints

- We need to also make sure that Steve and Eve both only do two chores
- Hence
  - $L_S + C_S + D_S + M_S = 2$
  - $L_E + C_E + D_E + M_E = 2$

Using these, try to formulate your MiniZinc model :D