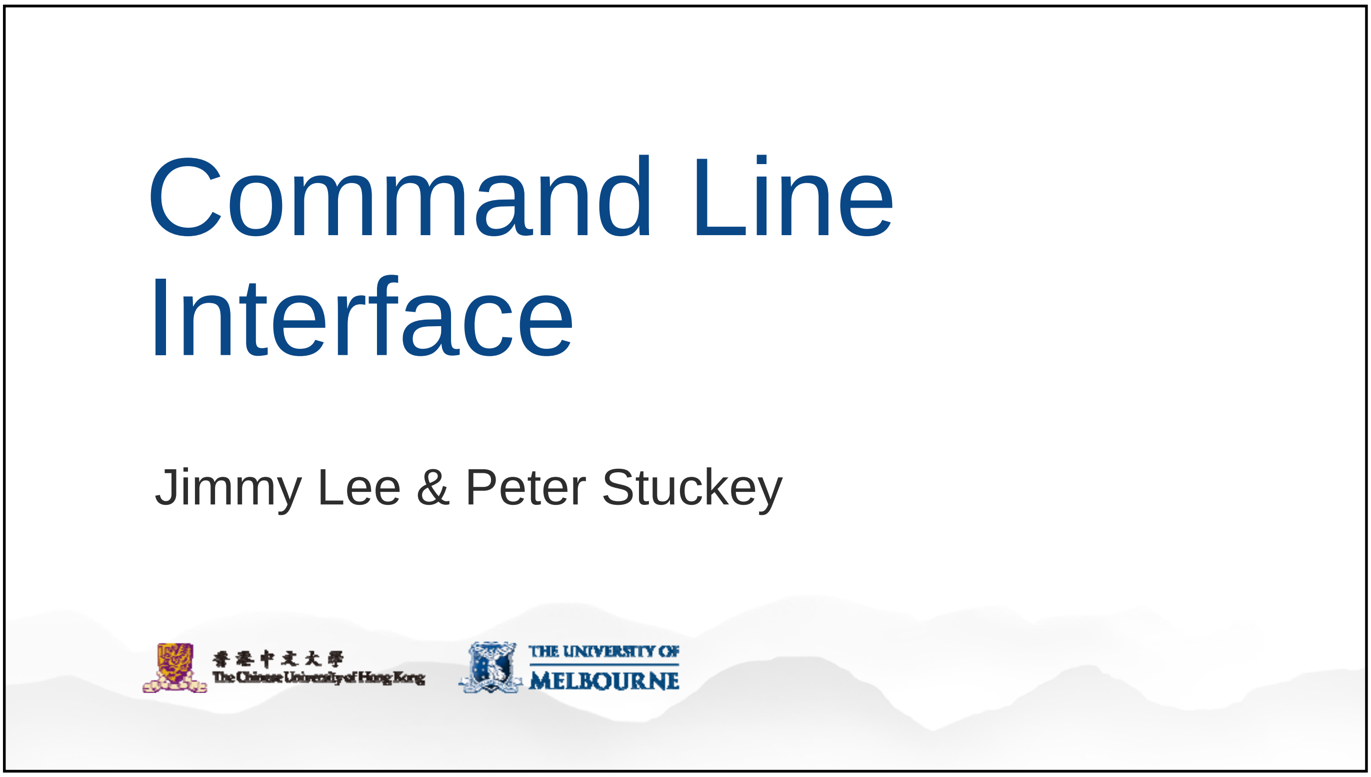
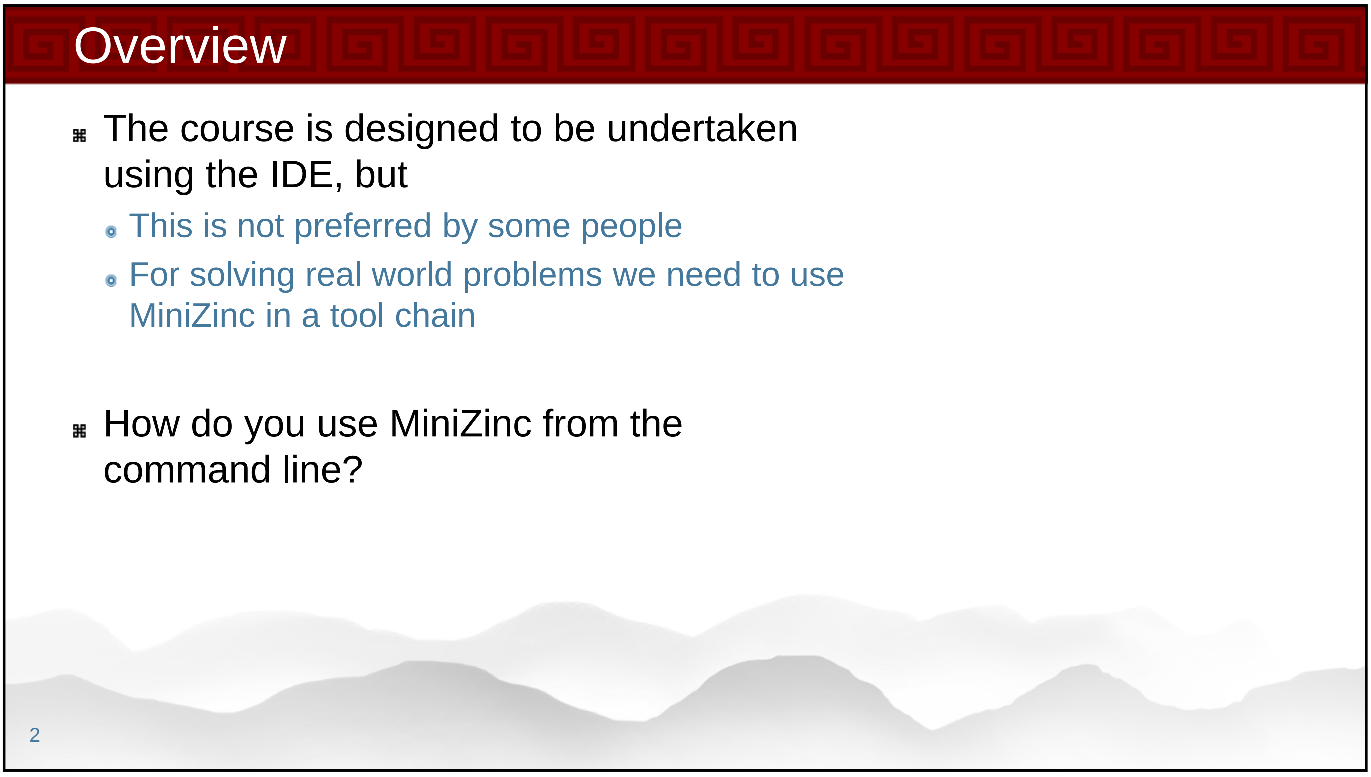


Command Line Interface

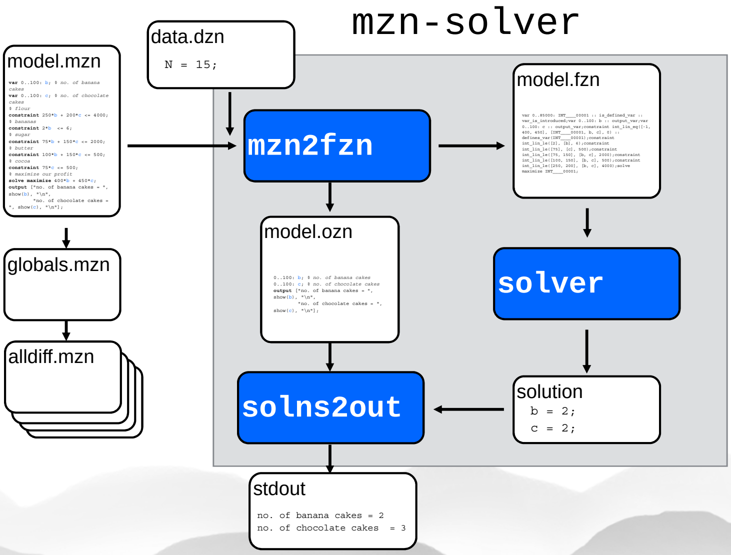
Jimmy Lee & Peter Stuckey



Overview

- The course is designed to be undertaken using the IDE, but
  - This is not preferred by some people
  - For solving real world problems we need to use MiniZinc in a tool chain
- How do you use MiniZinc from the command line?

The MiniZinc Tool Chain



model.mzn

```
var 0..100: b; $ no. of banana cakes
var 0..100: c; $ no. of chocolate cakes
$ flour
constraint 250*b + 200*c <= 4000;
$ bananas
constraint 2*b  <= 6;
$ sugar
constraint 75*b + 150*c <= 2000;
$ butter
constraint 100*b + 150*c <= 500;
$ cocoa
constraint 75*c <= 500;
$ maximize our profit
solve maximize 400*b + 450*c;
output ["no. of banana cakes = ", show(b), "\n",
        "no. of chocolate cakes = ", show(c), "\n"];
```

data.dzn

```
N = 15;
```

model.fzn

```
var 0..85000: INT____00001 :: is_defined_var :: var_is_introduced;var 0..100: b :: output_var;var 0..100: c :: output_var;constraint int_lin_eq([-1, 400, 450], [INT____00001, b, c], 0) :: defines_var(INT____00001);constraint int_lin_le([2], [b], 6);constraint int_lin_le([75], [c], 500);constraint int_lin_le([75, 150], [b, c], 2000);constraint int_lin_le([100, 150], [b, c], 500);constraint int_lin_le([250, 200], [b, c], 4000);solve maximize INT____00001;
```

model.ozn

```
0..100: b; $ no. of banana cakes
0..100: c; $ no. of chocolate cakes
output ["no. of banana cakes = ", show(b), "\n",
        "no. of chocolate cakes = ", show(c), "\n"];
```

solution

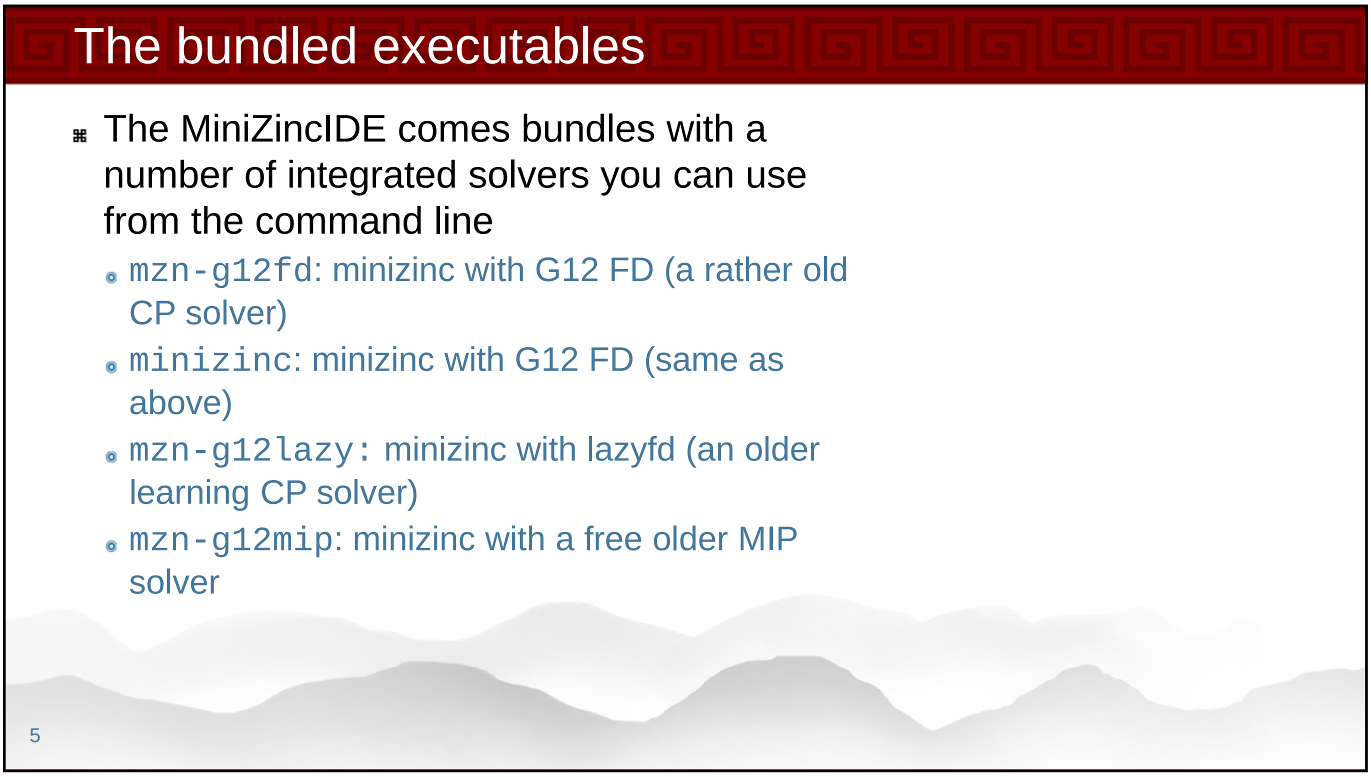
```
b = 2;
c = 2;
```

stdout

```
no. of banana cakes = 2
no. of chocolate cakes  = 3
```

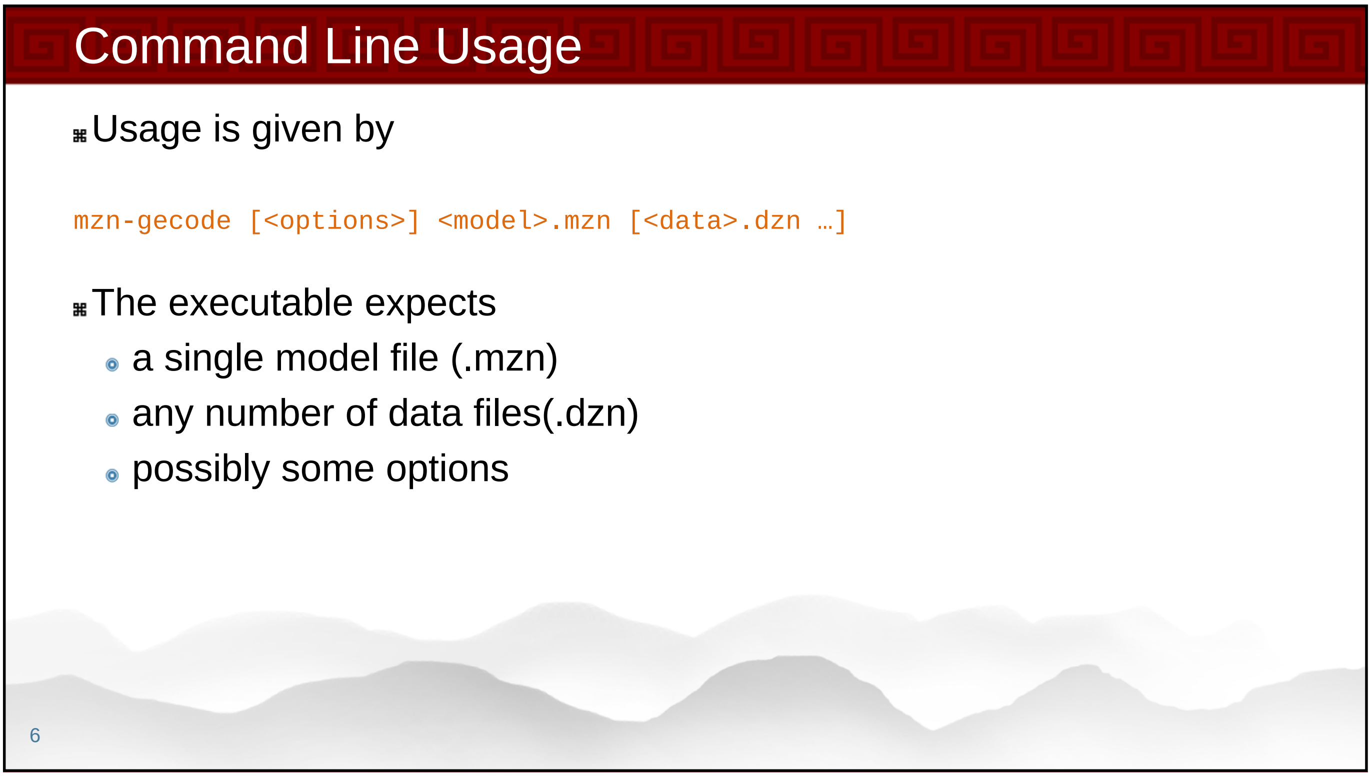
The bundled executables

- The MiniZincIDE comes bundles with a number of integrated solvers you can use from the command line
  - `mzn-gecode`: minizinc with Gecode (the default solver for the course, recommended)
  - `mzn-chuffed`: minizinc with chuffed (a learning CP solver)
  - `mzn-cbc`: minizinc with COIN OR CBC (a free MIP solver)
  - `mzn-gurobi`: minizinc with Gurobi (a commercial MIP solver - but you need to install the DLL)



The bundled executables

- The MiniZincIDE comes bundles with a number of integrated solvers you can use from the command line
  - `mzn-g12fd`: minizinc with G12 FD (a rather old CP solver)
  - `minizinc`: minizinc with G12 FD (same as above)
  - `mzn-g12lazy:` minizinc with lazyfd (an older learning CP solver)
  - `mzn-g12mip`: minizinc with a free older MIP solver



Command Line Usage

- Usage is given by

```
mzn-gecode [<options>] <model>.mzn [<data>.dzn …]
```

- The executable expects
  - a single model file (.mzn)
  - any number of data files(.dzn)
  - possibly some options

Default Behaviour

- By default
`mzn-gecode model.mzn data.dzn`
- For a satisfaction problem: `solve satisfy`
  - search for a solution
  - stop and print the first solution found, or
  - print =====UNSATISFIABLE=====
- For optimisation problem: `solve minimize` or `solve maximize`
  - search for the optimal solution
  - print the optimal solution, or
  - print =====UNSATISFIABLE=====
  - beware may take a long time to find the optimal

7

All Solutions

- An important option for MiniZinc is
  - `—all-solutions`, or `-a`

`mzn-gecode model.mzn data.dzn -a`
- For satisfaction problems
  - print all solutions found, with ——— separators
  - or, print =====UNSATISFIABLE=====
- For optimisation problems
  - print all solutions found on the way to the optimal
    - only one optimal solution, it will be the last printed
  - or, print =====UNSATISFIABLE=====
  - Note: here you can see progress to optimal

8

All Solutions

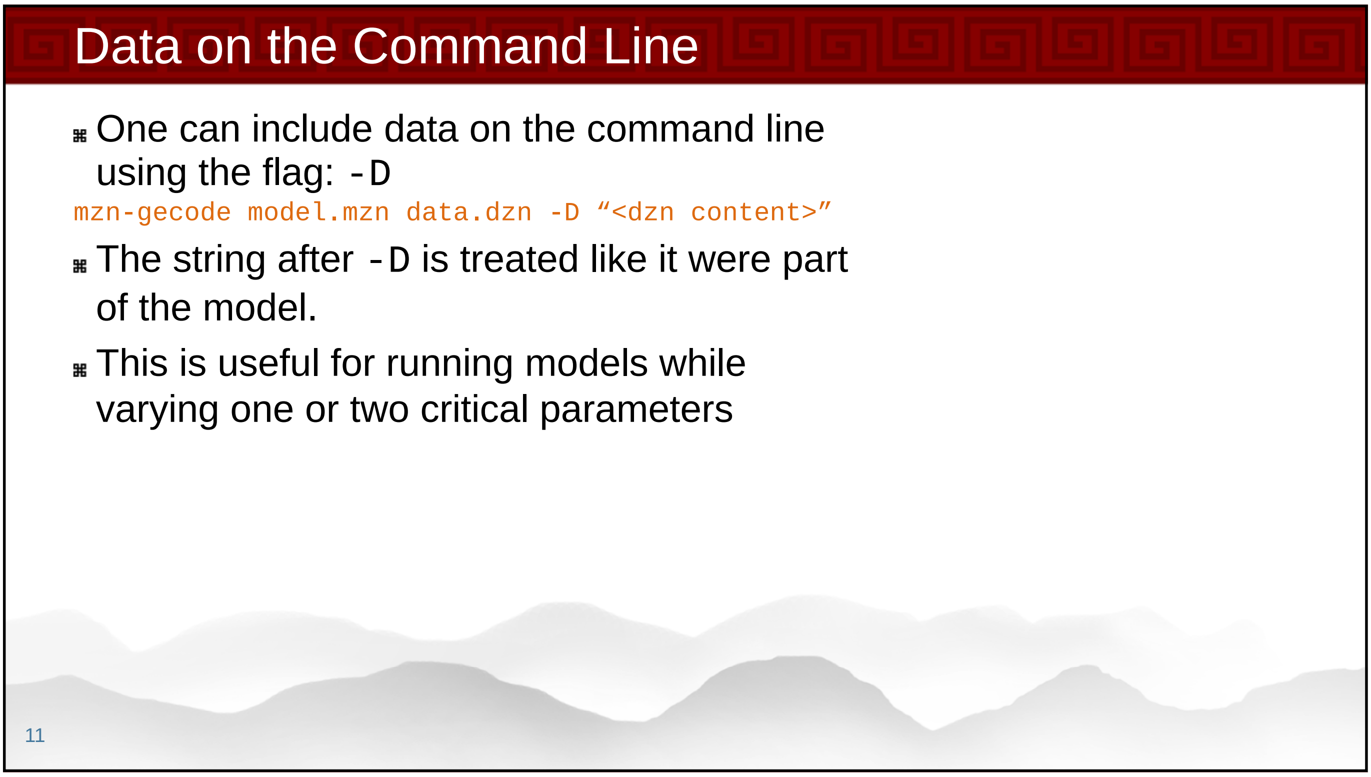
- Note that only CP solvers tend to support the all solutions flag
- MIP solvers such as
  - `mzn-cbc`
  - `mzn-gurobi`
  - `mzn-g12mip`
  - only ever return one (optimal) solution

Statistics

- An important option for MiniZinc is
  - `–statistics`, or `-s`

```
mzn-gecode model.mzn data.dzn -s
```

- The solver will print statistics about the solving process
- The statistics printed depends on the solver
- Examples
  - runtime, solvetime,
  - number of solutions found
  - size of problem: vars + constraints
  - search stats: nodes, failures, restarts

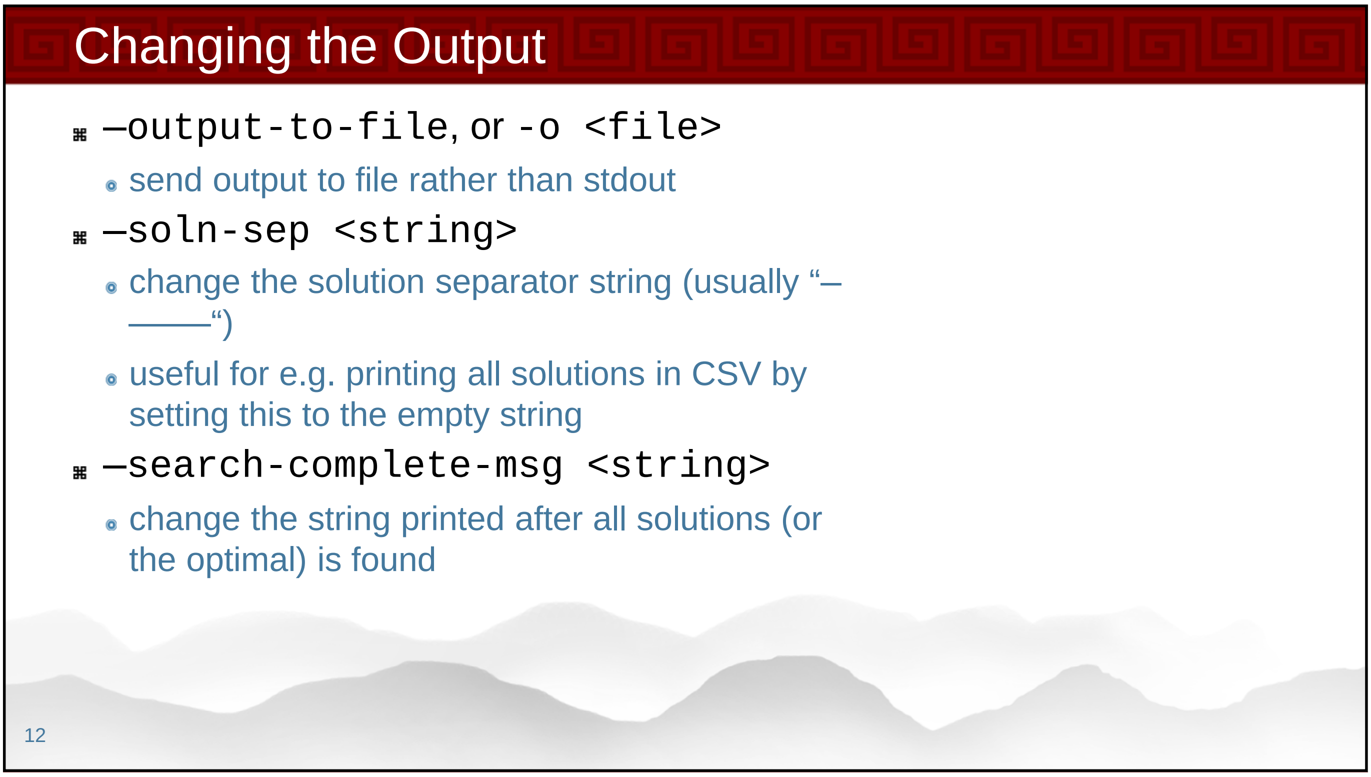


Data on the Command Line

- One can include data on the command line using the flag: `-D`

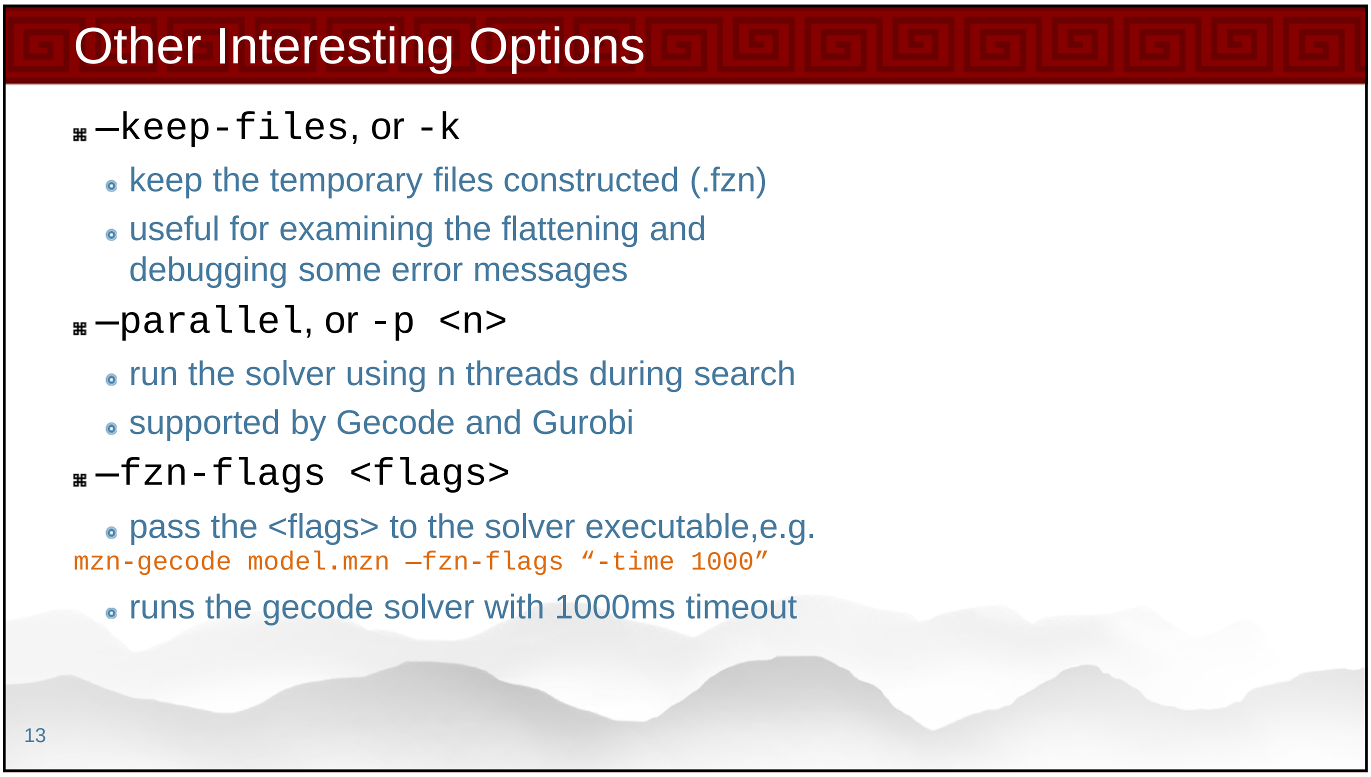
```
mzn-gecode model.mzn data.dzn -D “<dzn content>”
```

- The string after `-D` is treated like it were part of the model.
- This is useful for running models while varying one or two critical parameters



Changing the Output

- `–output-to-file`, or `-o <file>`
  - send output to file rather than stdout
- `–soln-sep <string>`
  - change the solution separator string (usually “––––“)
  - useful for e.g. printing all solutions in CSV by setting this to the empty string
- `–search-complete-msg <string>`
  - change the string printed after all solutions (or the optimal) is found



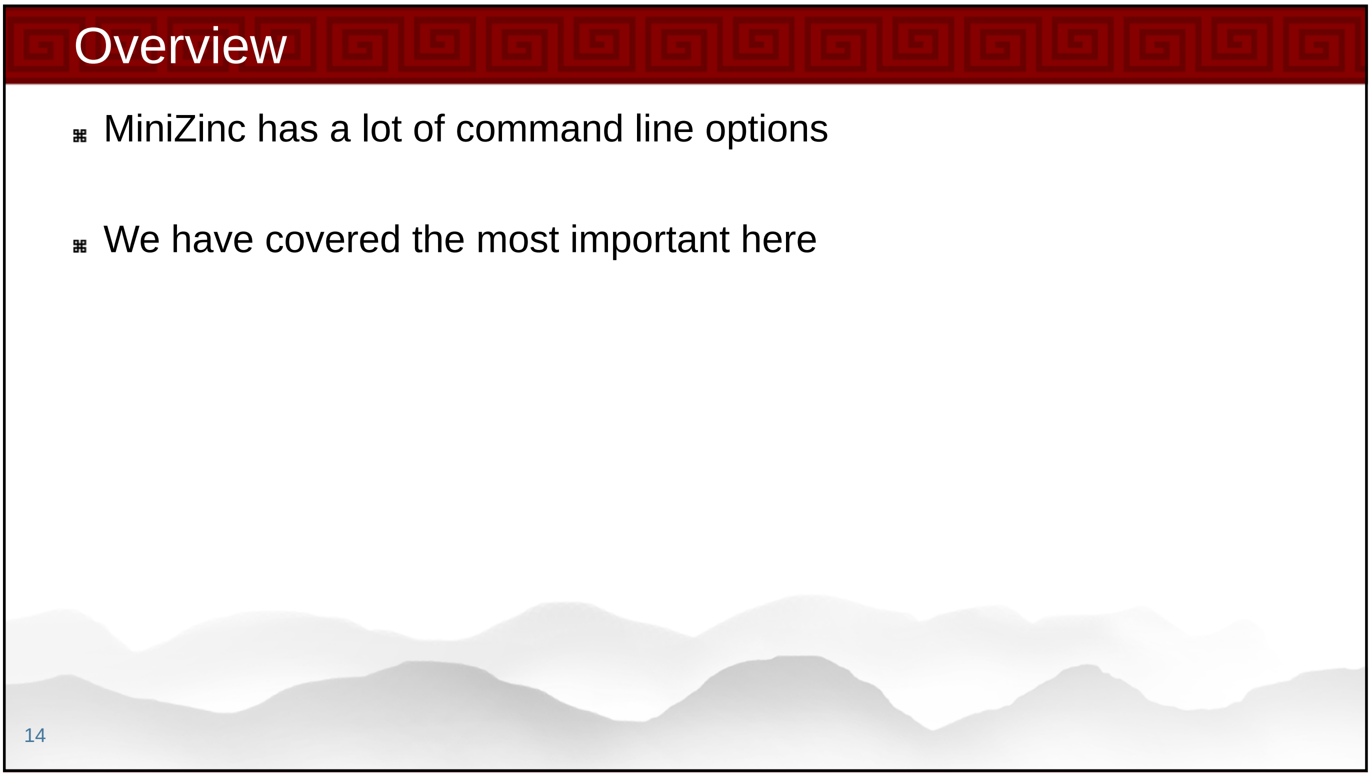
Other Interesting Options

- `–keep-files`, or `-k`
  - keep the temporary files constructed (.fzn)
  - useful for examining the flattening and debugging some error messages
- `–parallel`, or `-p <n>`
  - run the solver using n threads during search
  - supported by Gecode and Gurobi
- `–fzn-flags <flags>`
  - pass the <flags> to the solver executable,e.g.

```
mzn-gecode model.mzn –fzn-flags “-time 1000”
```

  - runs the gecode solver with 1000ms timeout

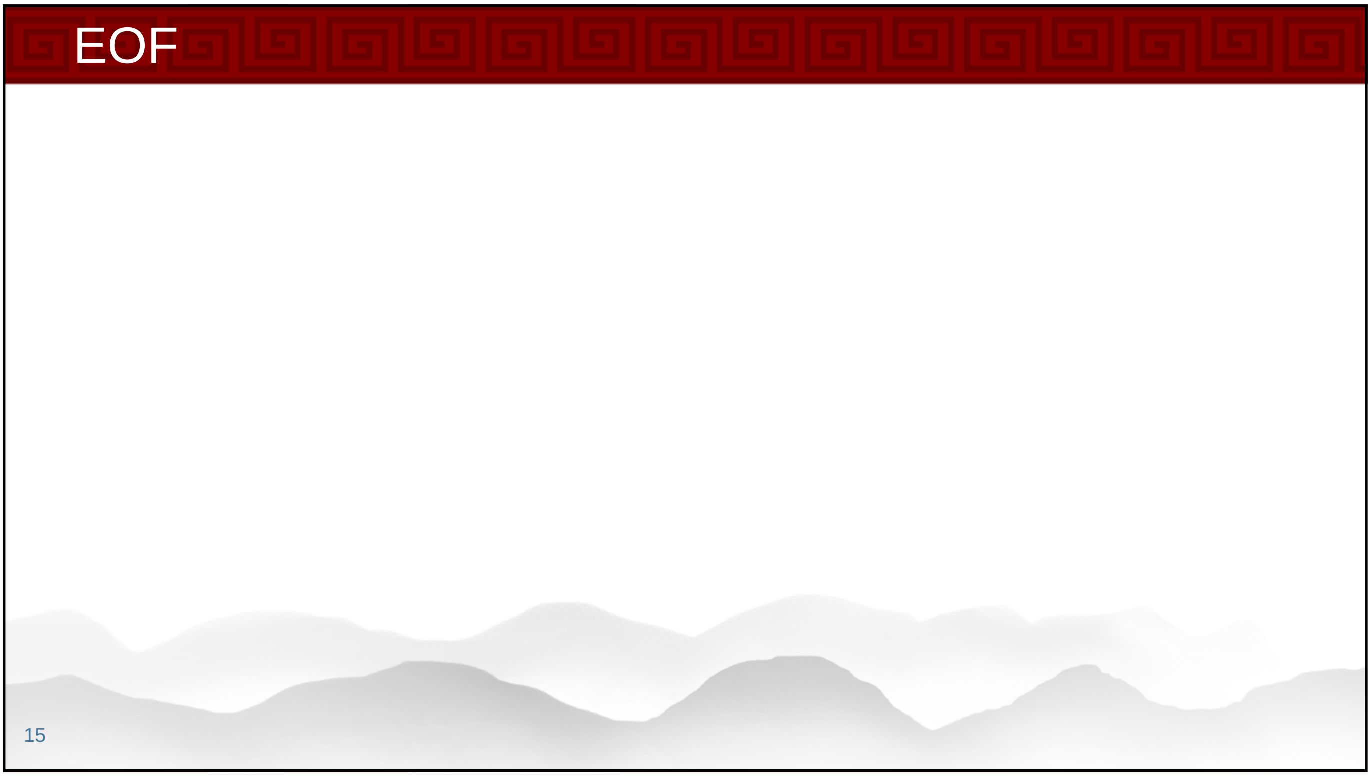
13



Overview

- MiniZinc has a lot of command line options
- We have covered the most important here

14



EOF

15