Numberjack Tutorial

(Adapted from Hebrard et al.’s AAAI 2010 tutorial and parts of the Numberjack website)

CS 275

January 7, 2016
1 Introduction

2 Intro to Python

3 Modeling in Numberjack

4 Examples
  - Map Coloring: Australia
  - N-Queens Problem
  - Magic Squares

5 Conclusion
1 Introduction

2 Intro to Python

3 Modeling in Numberjack

4 Examples
   - Map Coloring: Australia
   - N-Queens Problem
   - Magic Squares

5 Conclusion
What is Numberjack?

- A platform for constraints
- Written in Python - a front-end to C++-based solvers
- Excellent for rapidly trying out models
1 Introduction

2 Intro to Python

3 Modeling in Numberjack

4 Examples
   - Map Coloring: Australia
   - N-Queens Problem
   - Magic Squares

5 Conclusion
Overview of Python

- High-level, general purpose language
- Supports classes, objects, etc.
Basic Structures

Variables

\[ a = 2 \]

Functions

```python
def double(a):
    return a * 2
```

Lists

```python
foo = [1, 4, 5, 10, 2]
bar = ["this", "is", "a", "list"]
```

Tuples

```python
triplet = (1, 2, 3)
```
Control

```python
if <boolean_exp>:
    do_stuff()

while <boolean_exp>:
    do_stuff()
```
For Loops

For loops in C/C++/Java

```c
for (int i = 0; i < n; ++i) {
    do_stuff(i)
}
```

For loops in Python

```python
for i in range(n):
    do_stuff(i)
```

- Based on iterating through an iterable object
For Loops

```python
for element in list:
    do_stuff_with(element)

pairs = [(0, "Foo"),
         (1, "Bar"),
         (2, "Baz")]
for id, item in pairs:
    print "ID ", id, ":", item

ID 0 : Foo
ID 1 : Bar
ID 2 : Baz
```
List Comprehensions

A very useful feature!

```python
>>> range(4)
[0, 1, 2, 3]
>>> [x * 2 for x in range(4)]
[0, 2, 4, 6]
>>> [x * 2 for x in range(4) if x >= 2]
[4, 6]
```

Generally,

```
[<expression> for x in <Iterable> (if <condition>)]
```
1 Introduction

2 Intro to Python

3 Modeling in Numberjack

4 Examples
   - Map Coloring: Australia
   - N-Queens Problem
   - Magic Squares

5 Conclusion
Overview

- Constructs
  - Variables
  - Constraints
  - Model

- A common API to interface with back-end solvers
Variables

# binary variable
Variable()

# domain from 0 to N−1
Variable(N)

# domain from L to U
Variable(L, U)

# domain specified by a list
Variable(list)

Useful method (used after a solution has been found)

get_value()
Variables

More constructors:

# create a list of N binary variables
VarArray(N)

# create a list of N variables with domains from 0 to D–1
VarArray(N, D)

# create a list of N variables with domains from L to U
VarArray(N, L, U)
Variables

...and even more constructors:

```python
# create a matrix of M x N binary variables
m = Matrix(M, N)
# create a matrix of M x N variables with domains from L to U
m = Matrix(M, N, L, U)
```

Special operators

```python
# Return a VarArray containing all of the elements of the Matrix
m.flat
# Return a list of VarArrays corresponding to each row
m.row
# Return a list of VarArrays corresponding to each column
m.col
```
Constraints

- Arithmetic operators on variables
  
  \[
  x > y \\
  x == y + 2 \\
  m[1][4] != n[4][3]
  \]

- Global constructors

  \[
  \text{AllDiff}([a, b, c, d, e]) \\
  \text{AllDiff}(\text{myVarArray}) \\
  \text{AllDiff}(\text{myMatrix}) \\
  \text{Sum}([a, b, c, d]) \geq e
  \]
# Model

- Used to collect the constraints together to define a problem

- Constructors

  ```python
  # empty model
  model = Model()
  
  # model with constraints
  model = Model(constraints,...)
  ```

- Adding more constraints

  ```python
  model.add(constraints)
  # or
  model += constraints
  ```
Using a Solver

- Different solvers supported
  - SAT: MiniSat, Walksat
  - MIP: Gurobi, CPLEX, SCIP
  - CP: Mistral
Using a Solver

- Methods

```python
# Get a solver to solve the given problem specified
# by the model,
solver = model.load('nameOfSolver')
# attempts to solve the problem
solver.solve()
# for search–based solvers only (to generate multiple solutions)
solver.startNewSearch()
while solver.getNewSolution():
    # do something with solution
```

- Results are stored in the Variable objects
Outline of Usage

- Specify variables
- Specify constraints over those variables
- Construct a model with the constraints
- Construct the solver using that model
- Call solve() and extract results from Variables using get_value()
- Can alternatively use the print statement on Variables directly to output their values
1. Introduction

2. Intro to Python

3. Modeling in Numberjack

4. Examples
   - Map Coloring: Australia
   - N-Queens Problem
   - Magic Squares

5. Conclusion
Color the map such that no two adjacent territories have the same color.
N-Queens Problem

1. Introduction
2. Intro to Python
3. Modeling in Numberjack
4. Examples
   - Map Coloring: Australia
   - N-Queens Problem
   - Magic Squares
5. Conclusion
Problem Definition

- Place queens on the chessboard such that no two queens are attacking each other
1. Introduction

2. Intro to Python

3. Modeling in Numberjack

4. Examples
   - Map Coloring: Australia
   - N-Queens Problem
   - Magic Squares

5. Conclusion
Problem Definition

- Given an $N \times N$ square, place numbers ranging from 1 to $N^2$ such that each row, column, and diagonal has the same sum.
Conclusion

- Rapid prototyping of problems
- Easy to test out different solvers
- Numberjack website: http://numberjack.ucc.ie (also linked from the course page)