Discrete Optimization

Assignments: Optimization Tools

Optimization Tools

- General purpose optimization tools!
  - Let us not reinvent the wheel

Optimization Tools

- General purpose optimization tools!
  - Let us not reinvent the wheel

- Pros:
  - Rapid prototyping of ideas
  - Include decades of research / fancy algorithms

Optimization Tools

- General purpose optimization tools!
  - Let us not reinvent the wheel

- Pros:
  - Rapid prototyping of ideas
  - Include decades of research / fancy algorithms

- Cons:
  - Learning curve, platform/language limitations
  - Often slower than dedicated algorithms
  - May appear to do strange things
    - if you do not know how they work.

Optimization Tools

- Types of Tools
  - CP Solvers
  - MIP Solvers
  - LS Solvers

Optimization Tools

- Types of Tools
  - CP Solvers
  - MIP Solvers
  - LS Solvers

- Availability
  - open source
  - free for non-commercial use
  - free for students
    - May need an *edu* e-mail
  - see the partial list we provide

N-Queens Example

```
range R = 1..8;
var{int} row[R] in R;
solve {
   forall(i in R,j in R: i < j) {
      row[i] ≠  row[j];
      row[i] ≠  row[j] + (j - i);
      row[i] ≠  row[j] - (j - i);
   }
}
```

N-Queens Example

```
range R = 1..8;
var{int} row[R] in R;
solve {
   forall(i in R,j in R: i < j) {
      row[i] ≠  row[j];
      row[i] ≠  row[j] + (j - i);
      row[i] ≠  row[j] - (j - i);
   }
}
```

looks like **alldifferent**

N-Queens Example

```
range R = 1..8;
var{int} row[R] in R;
solve {
   forall(i in R,j in R: i < j) {
      row[i] ≠  row[j];
      row[i] ≠  row[j] + (j - i);
      row[i] ≠  row[j] - (j - i);
   }
}
```

looks like **alldifferent**

would this be a better model?

```
range R = 1..8;
var{int} row[R] in R;
solve {
   alldifferent(row);
   alldifferent(all(i in R) row[i]+i);
   alldifferent(all(i in R) row[i]-i);
}
```

Optimization Tools

- Options
  - Go to the **alldifferent** lecture, implement your own propagator from scratch
  - Learn an optimization tool (e.g. CP solver) try out both models

Optimization Tools

- Options
  - Go to the **alldifferent** lecture, implement your own propagator from scratch
  - Learn an optimization tool (e.g. CP solver) try out both models

- An example of option 2,
  - the queens problem in Comet
    - A domain specific language for optimization, including a CP solver.

N-Queens in Comet

```
import cotfd;

Solver<CP> m();
range R = 1..8;
var<CP>{int} row[i in R](m,R);

solve<m> {
   forall(i in R,j in R: i < j) {
      m.post(row[i] != row[j]);
      m.post(row[i] != row[j]+(j-i));
      m.post(row[i] != row[j]-(j-i));
   }
} using {
   label(row);
   cout << row << endl;
}
```

N-Queens in Comet

```
import cotfd;

Solver<CP> m();
range R = 1..8;
var<CP>{int} row[i in R](m,R);

solve<m> {

   m.post(alldifferent(row));
   m.post(alldifferent(all(i in R) row[i] + i));
   m.post(alldifferent(all(i in R) row[i] - i));

} using {
   label(row);
   cout << row << endl;
}
```

Final Note

- Top results on the leader board are most often dedicated algorithms, not general purpose solvers.

Final Note

- Top results on the leader board are most often dedicated algorithms, not general purpose solvers.

- No solver technology is officially supported or recommended by this class
  - We are not required to answer questions about different solvers
  - Your welcome to help each other learn to use them

Have Fun!