1.Pitch 1

Imagine you live in a country that is regularly hit by natural disasters like earthquakes or hurricanes. When disaster strikes, you want your infrastructure to not break down entirely.

Consider for instance your electric grid. When an earthquake hits, each powerline in the grid has a certain chance of breaking. If too many of your power lines break, important buildings like hospitals may lose power.

Suppose I am now in charge of a big maintenance project for a power grid. It is my job to make sure that, when an earthquake happens, as many of the important buildings as possible are still connected to the power grid. I can do this by reinforcing power lines to increase the probability that they will remain intact after a natural disaster. I am given a bag of money and have to decide how to spend it.

This problem has three components:

1) Probability. Each powerline has a certain chance of breaking during a disaster.
2) A constraint. My budget is limited.
3) Optimisation. My task is to maximise the expected number of important buildings that are still connected to a power plant after a natural disaster.

This problem is a typical example of a Stochastic Constraint Optimisation Problem.

Examples of such problems are found in areas like governance, services and industry.

So how do we solve these problems?

This is my actual job as a computer scientist.

Our aim is to develop programming and solving methods for these kinds of problems, that are

1. generic: applicable to a wide range of problems;
2. accessible to anyone, even if they aren't programmers.

We found that we need to integrate modelling and solving techniques from the fields of Constraint Programming and Probabilistic Programming to achieve our goal.

If you have ever solved a sudoku puzzle, you have done constraint programming yourself. If you have ever tried to make a multiple choice test that you hadn't studied for, you have likely done some probabilistic reasoning for yourself; you *guessed* some of the answers.

What I love about these two fields is that their programming paradigms are *declarative*. When people think about programming, they imagine having to explain to the computer how to solve their problem. With a declarative language, you only have to tell the computer everything you know about the problem, and then ask your question. The computer will figure out how to answer that question for you. The beauty is that declarative languages are very intuitive and super quick and easy to learn.

While both Constraint Programming and Probabilistic Programming are declarative in nature, neither of them can solve our power grid problem by itself. Constraint Programming lacks a concept of uncertainty, while Probabilistic Programming does not allow hard constraints on expectations.

By marrying these fields, we have opened up many possibilities. We are now able to solve a range of problems that could not be solved, or even modelled, before. On top of that: they can be solved by anyone who is willing to learn a simple declarative language.

So back to our powergrid problem.

Can we fix it?

Yes, we can.

Thank you.