Dynamic Programming and Subtree Perfectness for Deterministic Discrete-Time Systems with Uncertain Rewards

Nathan Huntley Matthias Troffaes

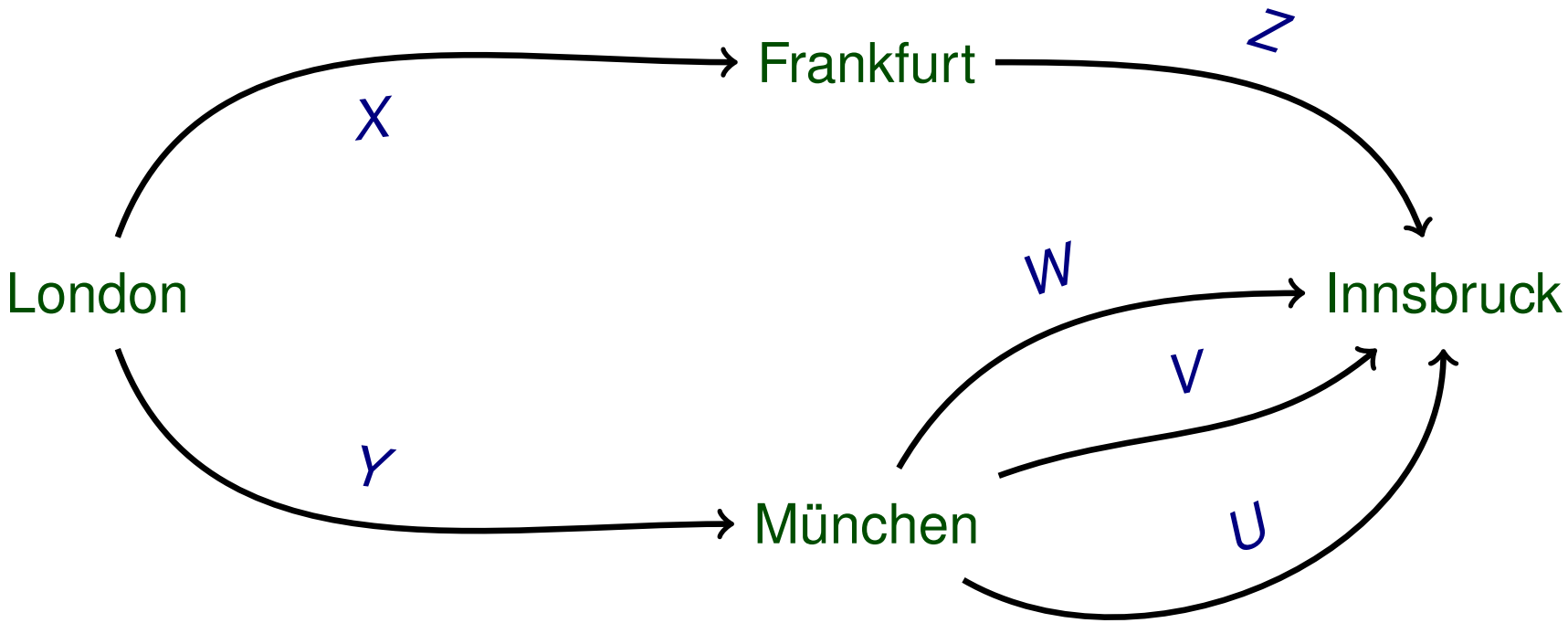
Durham University, United Kingdom

27 July, 2011

About Us

- Nathan Huntley
  - Redcar $\rightarrow$ Durham $\rightarrow$ Gent
  - master's thesis on Bayesian decision making in contaminated land
  - PhD student since 2007
  - main research interest is ~~Bayesian~~ ~~imprecise~~ decision theory under very weak assumptions
- Matthias Troffaes
  - Gent $\rightarrow$ Pittsburgh $\rightarrow$ Durham
  - decision making under severe uncertainty
  - computational tools
  - algorithms
  - engineering applications

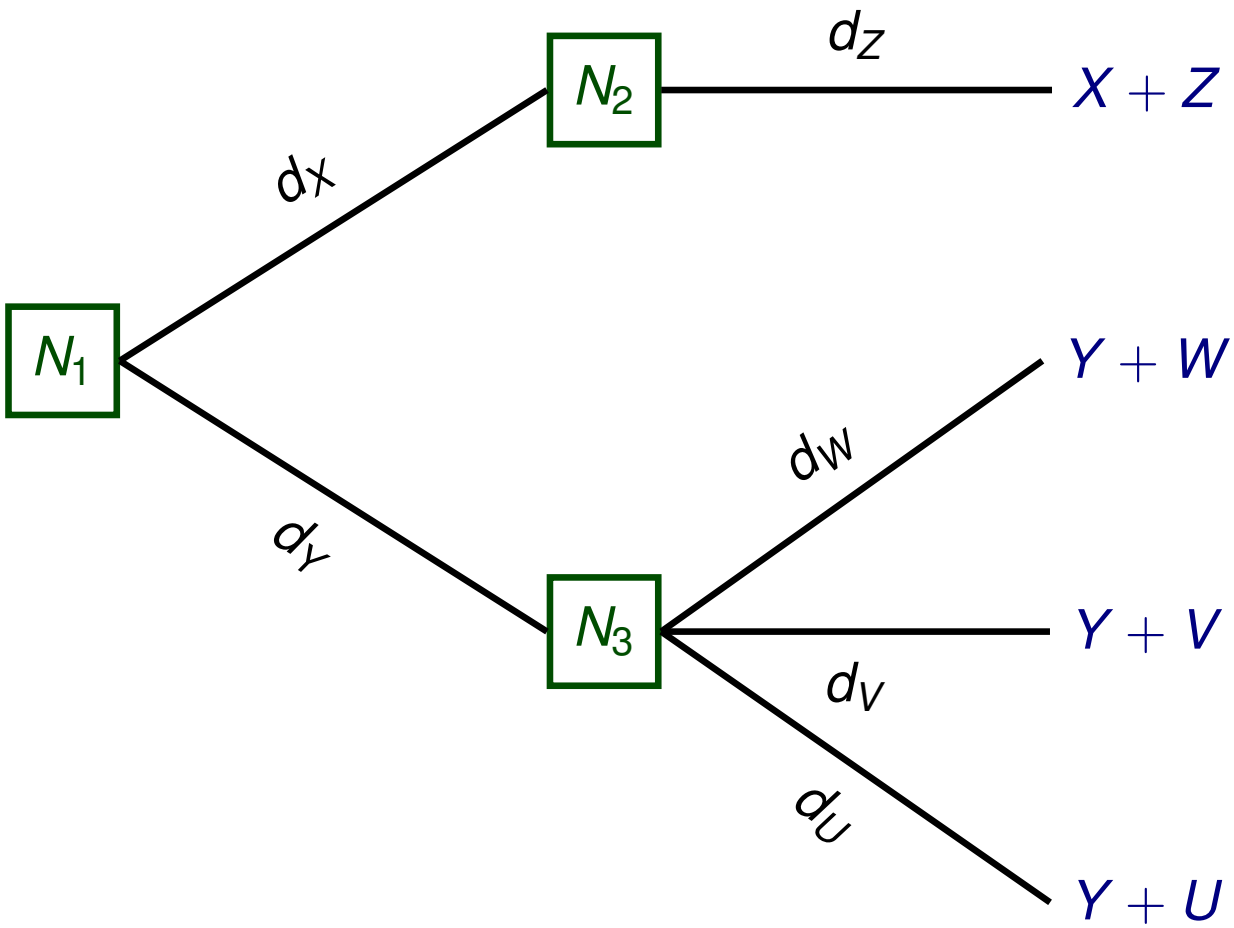
Simple Example (Certain Rewards)



Key Features

- at every node, subject must choose a single arc
- every arc has a certain reward expressed in utility
- overall reward is sum of rewards of the chosen arcs

Decision Tree Representation



Arbitrary Rewards

- $\mathcal{R}$: set of all possible rewards (arbitrary!)
- $+$: binary operator for combining rewards along paths

required assumptions on $+$ for what follows:

- left identity element: $0 + r = r$
- left inverse: $(-r) + r = 0$
- no additional structure assumed (no utility, no full ranking)

Adding Uncertainty

more generally, each arc induces a gamble instead of a certain reward:

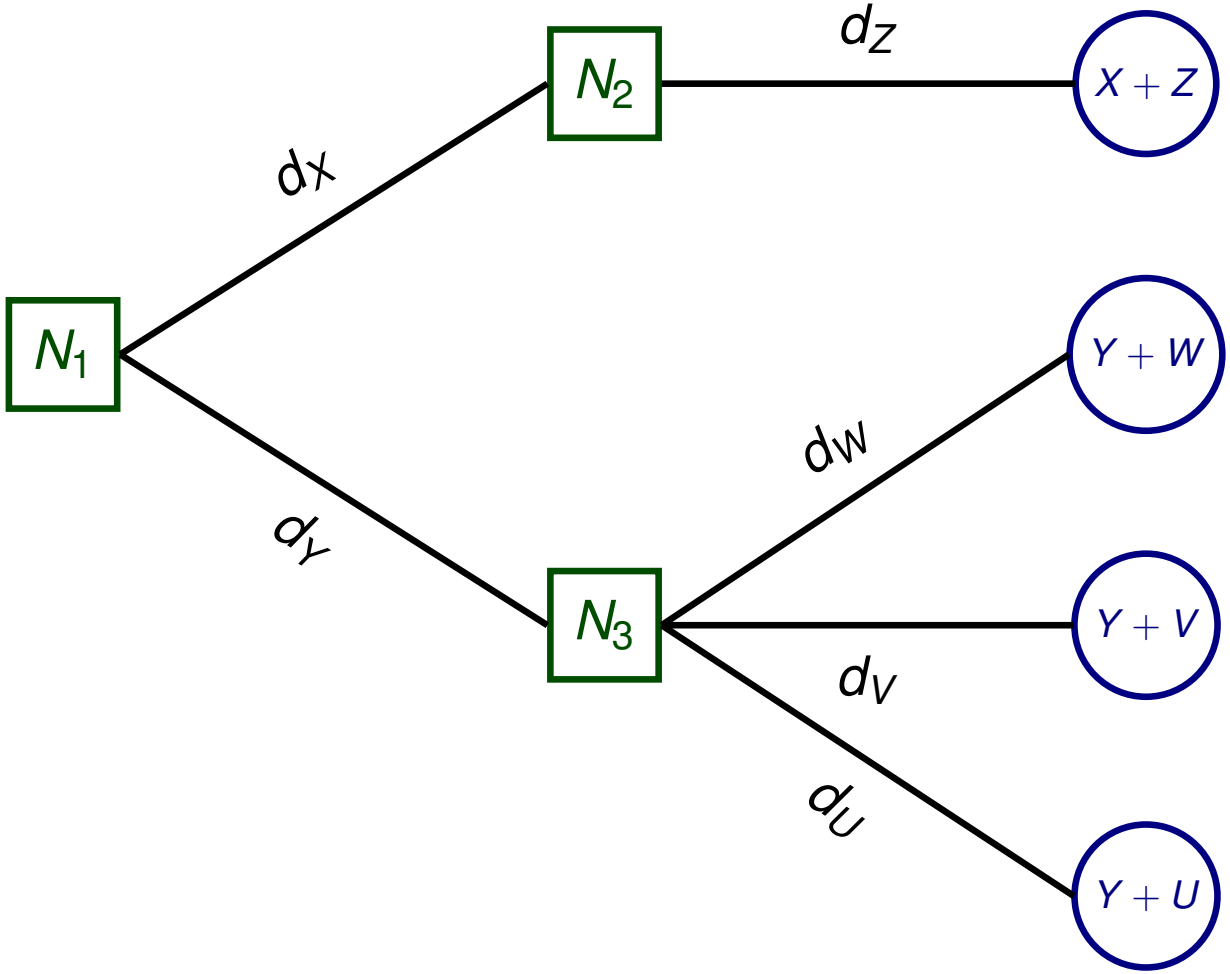
Definition (Gamble)

- Ω: set of all possible states of natures
- $\mathcal{R}$: set of all possible rewards
- gamble $X : \Omega \to \mathcal{R}$: assigns a reward to each state of nature

A Note on Conditioning

actual reward an arc gives is not known until the end of the process
$\Longrightarrow$ no conditional choice

Decision Tree with Uncertain Rewards



The Problem as a Set of Gambles

- each path in tree corresponds to a gamble
- set of gambles for our problem is

$$\{X + Z, Y + W, Y + V, Y + U\}$$

  (apply $+$ pointwise)
- use gambles to select between paths

Choice Functions

Definition (Choice Function)

a choice function maps each set of gambles $\mathcal{X}$
to a non-empty (optimal) subset of $\mathcal{X}$

examples:

- expected utility
- maximin
- pointwise dominance
- E-admissibility
- ...

Finding Optimal Paths: Backward Induction

Definition

Standard Normal Form Solution apply choice function to gambles of the problem to find the best paths

exploit structure of the problem to solve this more efficiently?
recursively remove paths by starting at end of tree

Theorem (Backward Induction)

we found necessary and sufficient conditions
to solve the problem by backward induction

Subtree Perfectness

even if backward induction works,
solution may still exhibit weirdness:
solution of subproblem may depend on problem it is embedded in

Theorem (Subtree Perfectness)

we found necessary and sufficient conditions
for the solution to be non-weird

- one of these conditions is total preorder (full ranking)
- in practice, only maximizing expected utility is non-weird

Want to learn more?
Visit our subtree perfect poster!

Thanks!