Get me out of here: Determining optimal policies

SEMINAR: CYBER-PHYSICAL SYSTEMS

ADVISOR: CHRISTIAN PEK

Outline

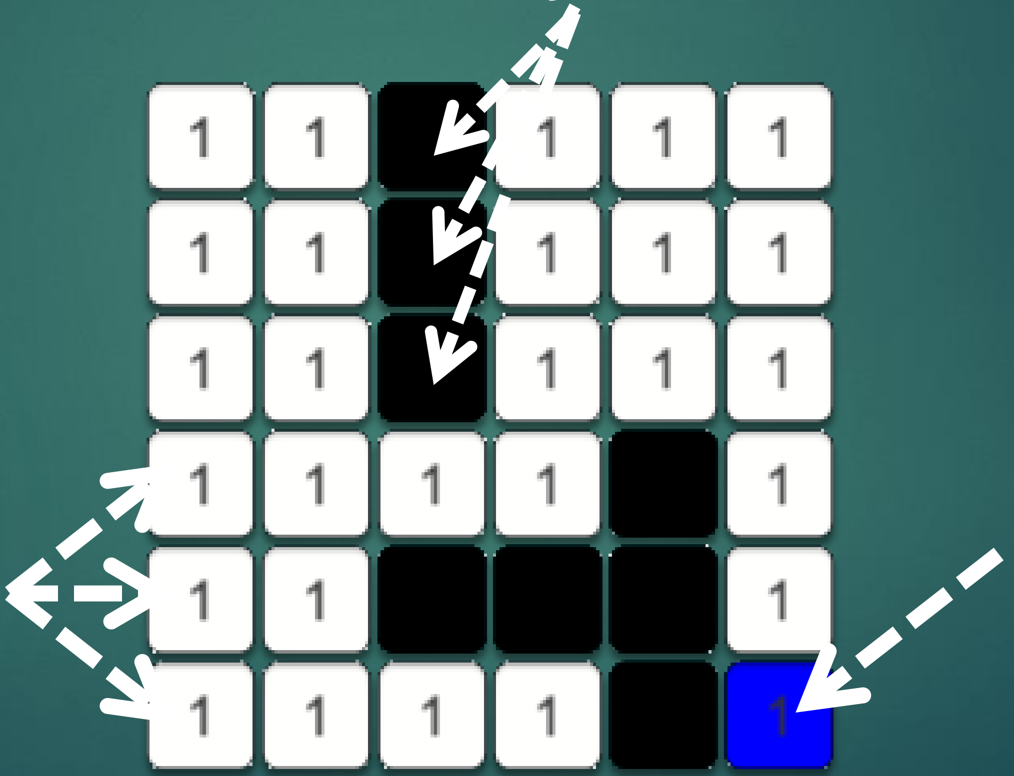
- Background and Problem Statement
- Fundamentals of Dynamic Programming
- Solving the Dynamic Programming equations
- Related work and use cases in robot motion planning

Background and Problem Statement

Background

Walls:
$M(i,j) = \infty$

Gridmap:
$M = \mathbb{N}^{n \times m}$



Cost to step on tile:
$M(i,j) \in \mathbb{N}$

Set of exits:
X_{goals}

Problem Statement

- Determine optimal policies

$\mathbb{N}^{n\times m} \rightarrow A \in \{\vee, <, >, \wedge\}$

1	1		1	1	1
1	1		1	1	1
1	1		1	1	1
1	1	1	1		1
1	1				1
1	1	1	1		1

$\rightarrow$

v	v		v	v	v
v	v		v	v	v
v	v		>	>	v
>	>	>	^		v
^	^				v
^	^	<	<		*

Fundamentals of Dynamic Programming

What is Dynamic Programming

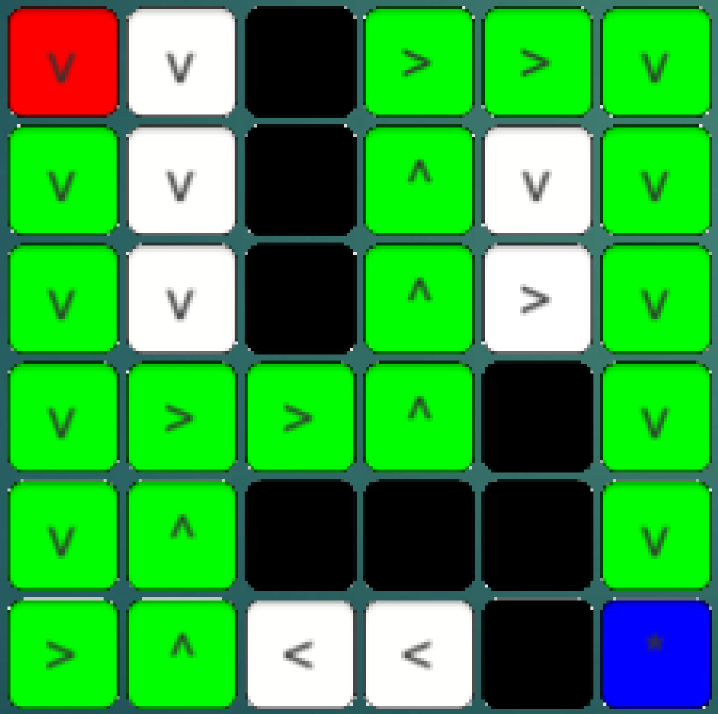
- Method used in mathematical optimization
- Transforms complex problems into sequences of simpler subproblems
- General framework for a broad variety of problems
- Base concept: Sequential decision making problem

The principle of optimality

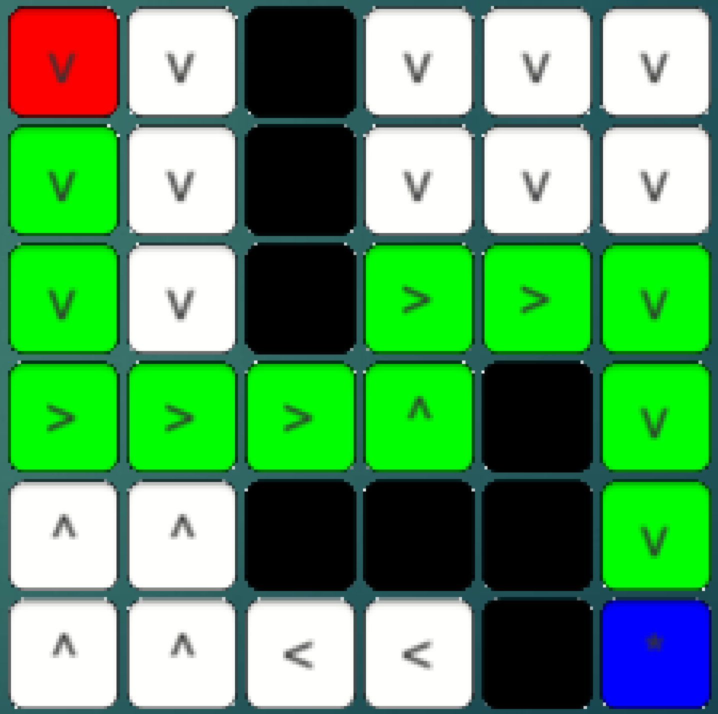
- A **policy** is a sequence of decisions
- An **optimal policy** is a sequence of decisions that has the best outcome w.r.t. a predefined criterion

The principle of optimality

A policy



An optimal policy



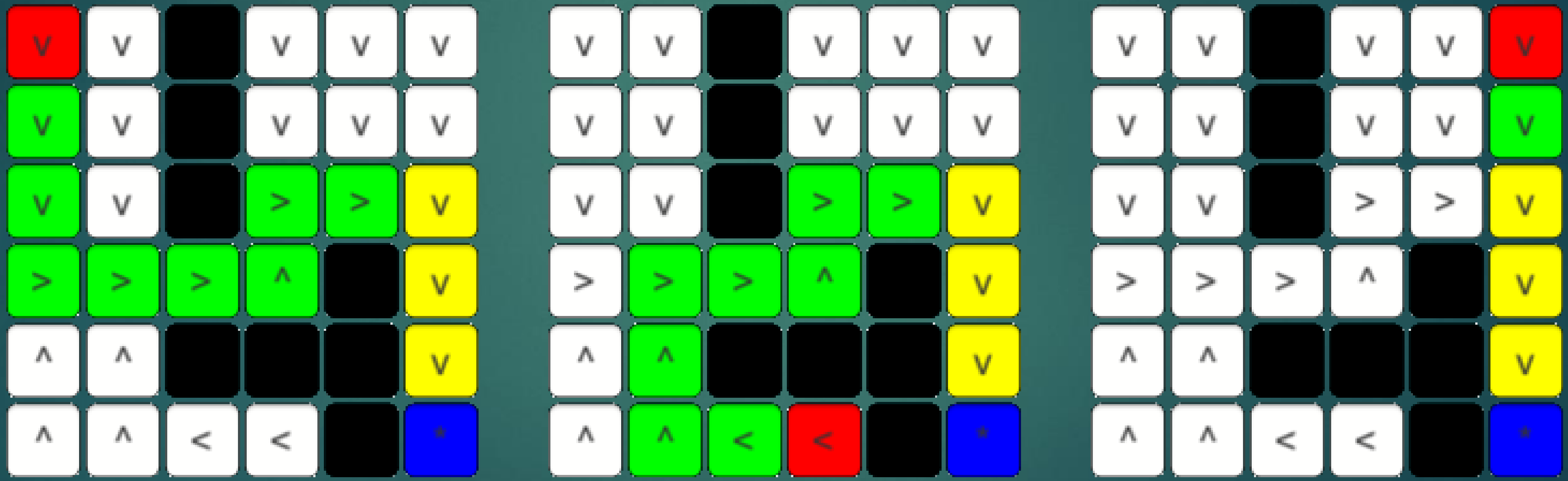
Characteristics of problems solvable by Dynamic Programming

FUNDAMENTALS OF DYNAMIC PROGRAMMING

Overlapping Subproblems

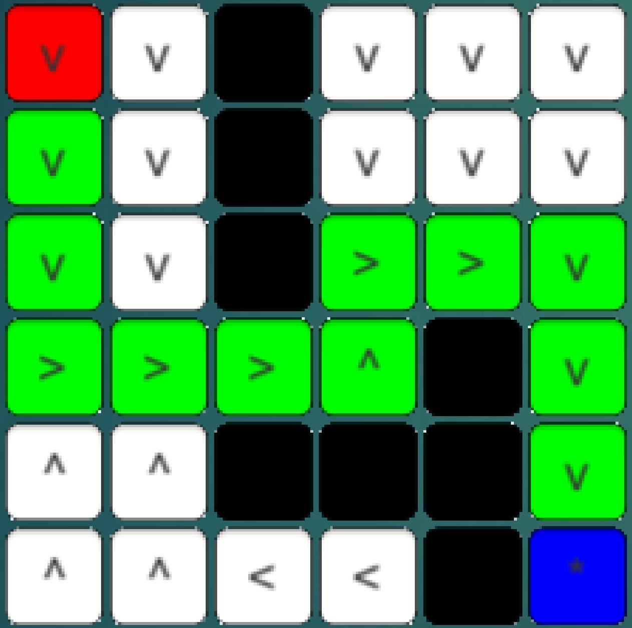
- Space of subproblems is small
- Solutions reused many times
- Recursive algorithm solves same subproblems over and over

Overlapping Subproblems

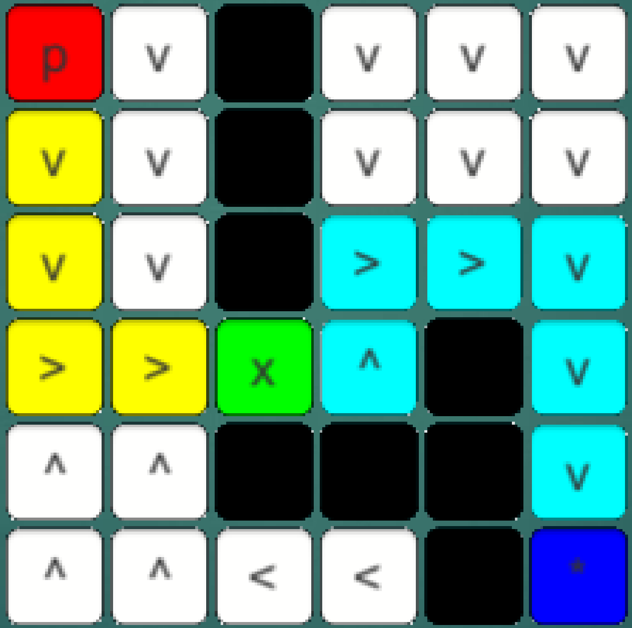


Optimal Substructure

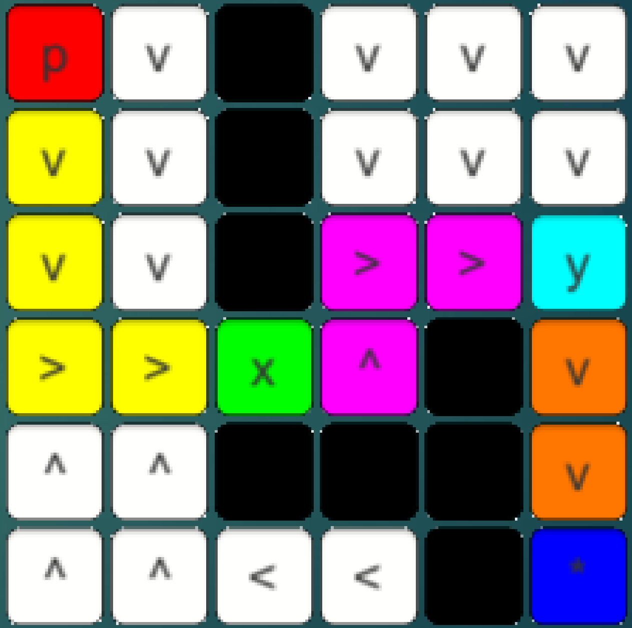
- Solution for whole problem by combination of solutions for subproblems



$S = red \rightarrow blue$



$S = red \rightarrow x \cup x \rightarrow blue$



$S = red \rightarrow x \cup x \rightarrow y \cup y \rightarrow blue$

How to apply Dynamic Programming in Computer Science

FUNDAMENTALS OF DYNAMIC PROGRAMMING

Memoization

- Break problem down into set of subproblems
- Store results in a table
- Use cached result if subproblem reoccures

Bellman equation

- Find a connection between one step and the next one
- $V(x) = \max\limits_{a \in \Gamma(x)} \{Payoff(x, a) + V(Transformation(x, a))\}$
- Richard Bellman introduced backwards induction
  - Begin at step N → Derive $(N - 1)$ → ⋯ → Initial step

Solving the Dynamic Programming equations

Finding the value function

- Goal is to find value function $v(i,j) \in \mathbb{N}$
- Cost function $c(i,j) = \begin{cases} \infty, & if\ (i,j) \notin M \\ M(i,j) & else \end{cases}$
- Trivial cases $v(i,j) = \begin{cases} \infty, & if\ c(i,j) = \infty \\ 0, & if\ (i,j) \in X_{goal} \end{cases}$

Finding the value function

- Backwards iteration: $(N, N-1, \dots, 0)$
- Store results in memoization table $\mathrm{V} = \mathbb{N}^{n \times m}$
- $$\left.\begin{array}{lr} v(i-1,j) & (left) \\ v(i+1,j) & (right) \\ v(i,j-1) & (up) \\ v(i,j+1) & (down) \end{array}\right\} = \quad v(i,j) + c(i,j)$$

Finding optimal policies

- Memoization table V holding the values of all tiles
- Find shortest paths $\Leftrightarrow$ Sequence with lowest sum of values
- $a(i,j) = \begin{cases} up, & if\ v(i,j-1) = \min\limits_{(x,y)}\{v(x,y)\} \\ right, & if\ v(i+1,j) = \min\limits_{(x,y)}\{v(x,y)\} \\ down, & if\ v(i,j+1) = \min\limits_{(x,y)}\{v(x,y)\} \\ left, & if\ v(i-1,j) = \min\limits_{(x,y)}\{v(x,y)\} \end{cases}$
- With $(x,y) \in \{(i+1,j),(i-1,j),(i,j+1),(i,j-1)\}$

Related work and use cases in robot motion planning

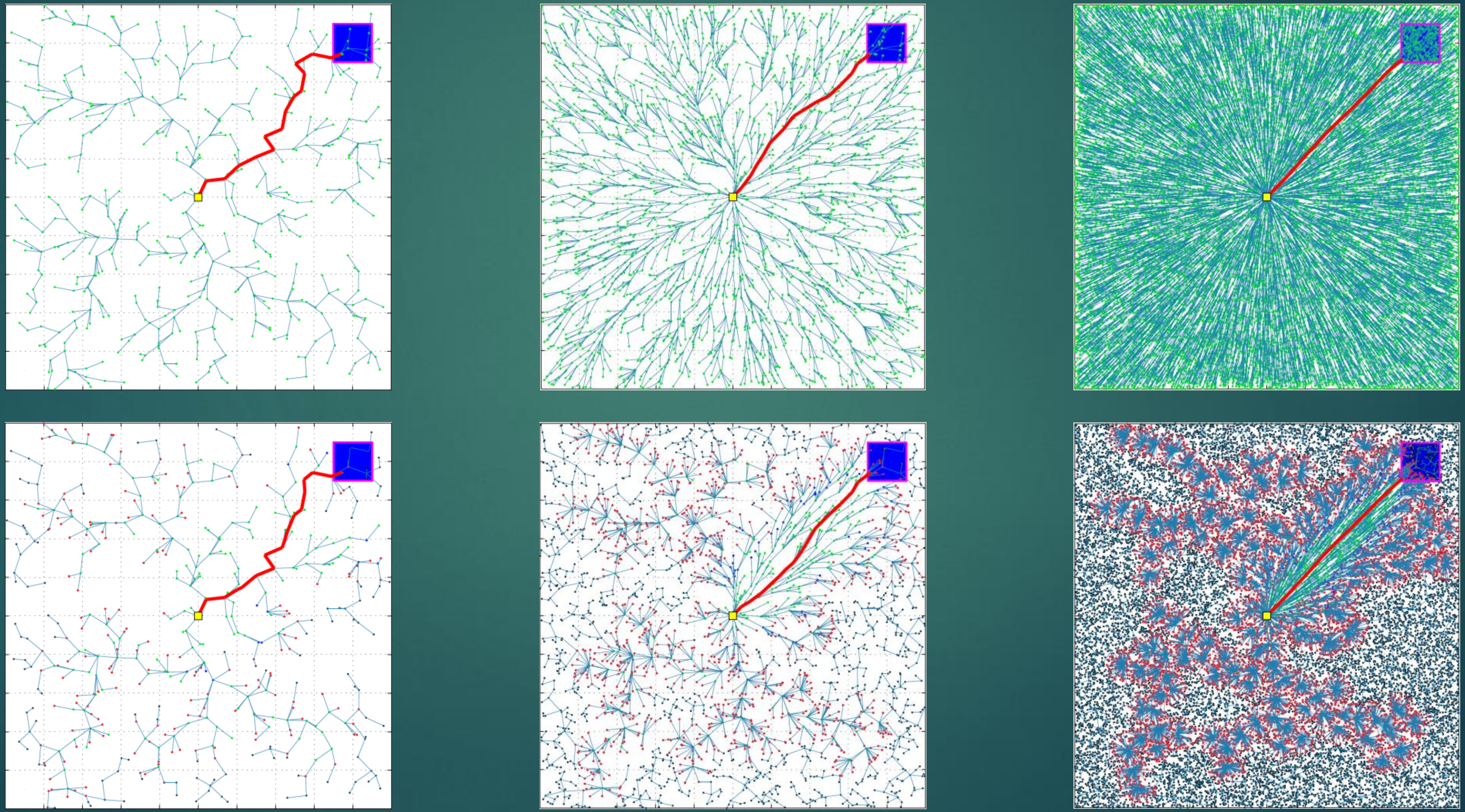
Extending the 2-dimensional algorithms onto N-dimensional problems

- Define a state as a node in a graph
- Define the transitions between states as edges in the graph
- Apply 2-dimensional algorithm onto graph to solve N-dimensional problem
- But curse of dimensions

Autonomous vehicles

- Cars etc. are just one type of robots
- Many research in last years
- Best action for every position is calculated
- Also applicable for vacuum robots…

Rapidly exploring random trees
Dynamic Programming



Some more examples!

Conclusion

- Multistage decision problems can be solved by

  Dynamic Programming

- Overlapping subproblems and optimal substructure

- Memoization and the Bellman equation

- Especially for robot motion planning and path finding