

Planning and Optimization

A2. What is Planning?

Malte Helmert and Gabriele Röger

Universität Basel

September 17, 2025

Planning and Optimization

September 17, 2025 — A2. What is Planning?

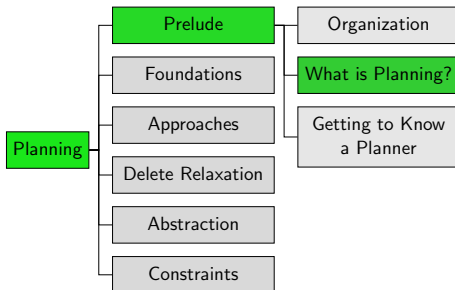
A2.1 Planning

A2.2 Planning Task Examples

A2.3 How Hard is Planning?

A2.4 Summary

Content of the Course



Before We Start...

- Prelude** (Chapters A1–A3): very high-level intro to planning
- ▶ our goal: give you a little feeling what planning is about
 - ▶ **preface** to the actual course
 - ~> main course content (beginning with Chapter B1)
will be mathematically formal and rigorous
 - ▶ You can ignore the prelude when preparing for the exam.

A2.1 Planning

General Problem Solving

Wikipedia: General Problem Solver

General Problem Solver (GPS) was a computer program created in 1959 by Herbert Simon, J.C. Shaw, and Allen Newell intended to work as a universal problem solver machine.

Any formalized symbolic problem can be solved, in principle, by GPS. [...]

GPS was the first computer program which separated its knowledge of problems (rules represented as input data) from its strategy of how to solve problems (a generic solver engine).

- ↪ these days called “domain-independent automated **planning**”
- ↪ this is what the course is about

So What is Domain-Independent Automated Planning?

Automated Planning (Pithy Definition)

“Planning is the art and practice of thinking before acting.”

— Patrik Haslum

Automated Planning (More Technical Definition)

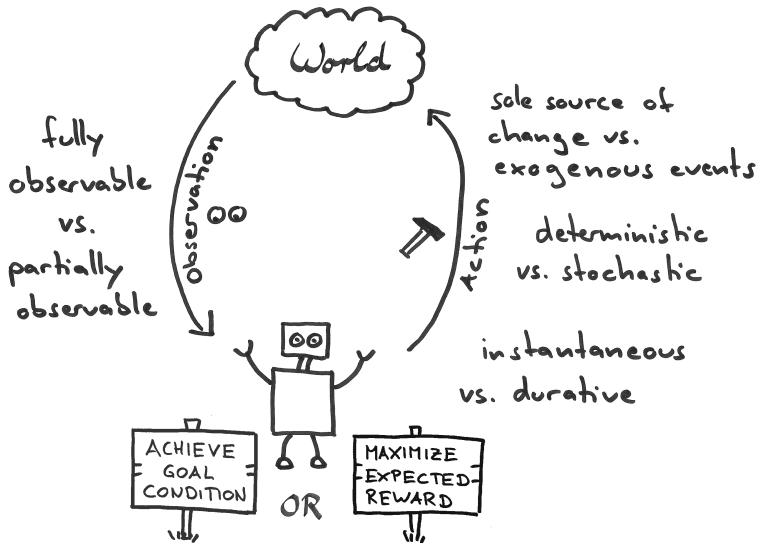
“Selecting a goal-leading course of action
based on a high-level description of the world.”

— Jörg Hoffmann

Domain-Independence of Automated Planning

Create **one** planning algorithm that performs sufficiently well
on **many** application domains (including future ones).

General Perspective on Planning



Example: Earth Observation



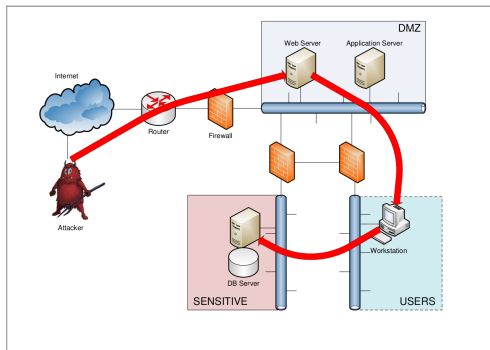
- ▶ satellite takes images of patches on Earth
- ▶ use [weather forecast](#) to optimize probability of high-quality images

Example: Termes



Harvard TERMES robots, based on termites

Example: Cybersecurity



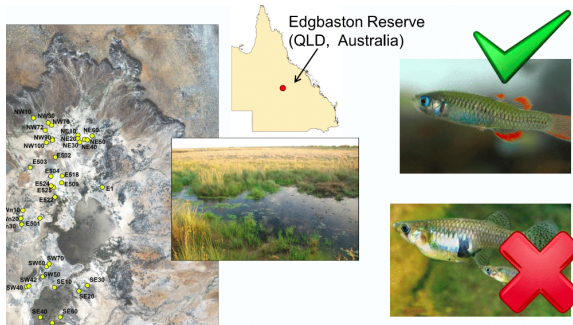
CALDERA automated adversary emulation system

Example: Intelligent Greenhouse



photo © LemnaTec GmbH

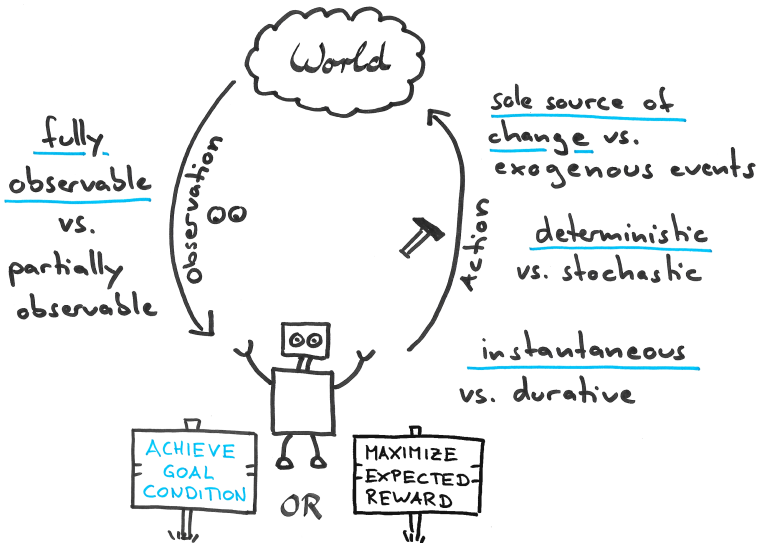
Example: Red-finned Blue-eye



Picture by Iadine Chadès

- ▶ red-finned blue-eye population threatened by **gambusia**
- ▶ springs **connected probabilistically** during rain season
- ▶ find strategy to **save** red-finned blue-eye from **extinction**

Classical Planning



Model-based vs. Data-driven Approaches



Model-based approaches know the “inner workings” of the world
~> reasoning



Data-driven approaches rely only on collected data from a black-box world
~> learning

We focus on model-based approaches.

Planning Tasks

input to a planning algorithm: **planning task**

- ▶ initial state of the world
- ▶ actions that change the state
- ▶ goal to be achieved

output of a planning algorithm:

- ▶ **plan**: sequence of actions taking initial state to a goal state
- ▶ or confirmation that no plan exists

↪ formal definitions later in the course

The Planning Research Landscape

- ▶ one of the major subfields of Artificial Intelligence (AI)
- ▶ represented at major AI conferences (IJCAI, AAAI, ECAI)
- ▶ annual specialized conference ICAPS (\approx 250 participants)
- ▶ major journals: general AI journals (AIJ, JAIR)

Classical Planning

This course covers **classical planning**:

- ▶ offline (static)
- ▶ discrete
- ▶ deterministic
- ▶ fully observable
- ▶ single-agent
- ▶ sequential (plans are action sequences)
- ▶ domain-independent

This is just **one facet** of planning.

Many others are studied in AI. Algorithmic ideas often (but not always) translate well to more general problems.

More General Planning Topics

More general kinds of planning include:

- ▶ ~~offline~~: online planning; planning and execution
- ▶ ~~discrete~~: continuous planning (e.g., real-time/hybrid systems)
- ▶ ~~deterministic~~: FOND planning; probabilistic planning
- ▶ ~~single-agent~~: multi-agent planning; general game playing; game-theoretic planning
- ▶ ~~fully observable~~: POND planning; conformant planning
- ▶ ~~sequential~~: e.g., temporal planning

Domain-dependent planning problems in AI include:

- ▶ pathfinding, including grid-based and multi-agent (MAPF)
- ▶ continuous motion planning

A2.2 Planning Task Examples

Example: The Seven Bridges of Königsberg

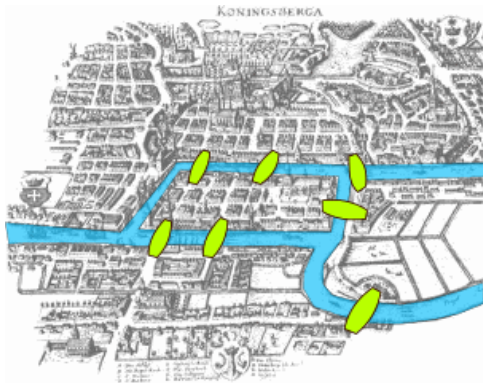


image credits: Bogdan Giuscă (public domain)

Demo

\$ ls demo/koenigsberg

Example: Intelligent Greenhouse



photo © LemnaTec GmbH

Demo

```
$ ls demo/ipc/scanalyzer-08-strips
```

Example: FreeCell

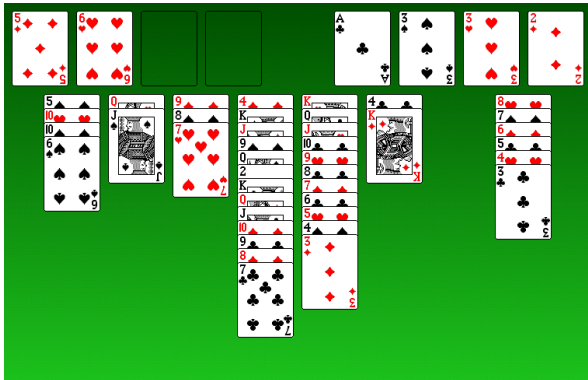


image credits: GNOME Project (GNU General Public License)

Demo Material

```
$ ls demo/ipc/freecell
```

Many More Examples

Demo

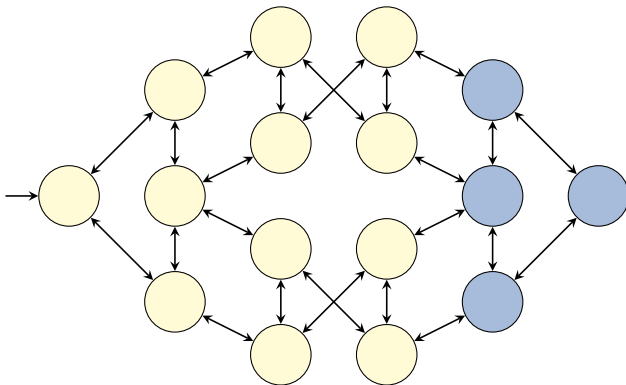
```
$ ls demo/ipc
agricola-opt18-strips
agricola-sat18-strips
airport
airport-adl
assembly
barman-mco14-strips
barman-opt11-strips
barman-opt14-strips
...
```

↪ (most) benchmarks of planning competitions IPC since 1998

A2.3 How Hard is Planning?

Classical Planning as State-Space Search

classical planning as **state-space search**:



~> much more on this later in the course

Is Planning Difficult?

Classical planning is computationally challenging:

- ▶ number of states grows **exponentially** with description size when using (propositional) logic-based representations
- ▶ **provably hard** (PSPACE-complete)

↪ we prove this later in the course

problem sizes:

- ▶ Seven Bridges of Königsberg: **64** reachable states
- ▶ Rubik's Cube: **$4.325 \cdot 10^{19}$** reachable states
↪ consider 2 billion/second ↪ 1 billion years
- ▶ standard benchmarks: some with **$> 10^{200}$** reachable states

A2.4 Summary

Summary

- ▶ **planning** = thinking before acting
- ▶ major subarea of Artificial Intelligence
- ▶ **domain-independent** planning = general problem solving
- ▶ **classical planning** = the “easy case”
(deterministic, fully observable etc.)
- ▶ still hard enough!
 \rightsquigarrow PSPACE-complete because of huge number of states
- ▶ often solved by **state-space search**
- ▶ number of states grows **exponentially** with input size