How hard is planning?

- We have seen that planning can be done in time polynomial in the size of the transition system.
- However, we have not seen algorithms which are polynomial in the input size (size of the task description).
- What is the precise computational complexity of the planning problem?

Why computational complexity?

- understand the problem better
- know what is not possible
- get a licence for using heuristic search methods (or other methods to solve NP-hard problems)
- find interesting subproblems that are easier to solve
- distinguish essential features from syntactic sugar
  - Is STRIPS planning easier than general planning?
  - Is planning for FDR tasks harder than for propositional tasks?
Background

December 21, 2018  B. Nebel, R. Mattmüller – AI Planning
5 / 29

Turing machines

December 21, 2018  B. Nebel, R. Mattmüller – AI Planning
6 / 29

Motivation

Background

Turing machines

Complexity classes

Summary

December 21, 2018  B. Nebel, R. Mattmüller – AI Planning
5 / 29

Nondeterministic Turing machines

Definition (nondeterministic Turing machine)

A nondeterministic Turing machine (NTM) is a 6-tuple \( \langle \Sigma, \square, Q, q_0, q_Y, \delta \rangle \) with the following components:

- **input alphabet** \( \Sigma \) and blank symbol \( \square \notin \Sigma \)
- **alphabets** always nonempty and finite
- **tape alphabet** \( \Sigma \square = \Sigma \cup \{ \square \} \)
- **finite set** \( Q \) of internal states with initial state \( q_0 \in Q \) and accepting state \( q_Y \in Q \)
- **nonterminal states** \( Q' := Q \setminus \{ q_Y \} \)
- **transition relation** \( \delta \subseteq (Q' \times \Sigma) \times (Q \times \Sigma \times \{-1,+1\}) \)

December 21, 2018  B. Nebel, R. Mattmüller – AI Planning
6 / 29

Deterministic Turing machines

Definition (deterministic Turing machine)

A deterministic Turing machine (DTM) is an NTM where the transition relation is functional, i.e., for all \( \langle q, a \rangle \in Q' \times \Sigma \), there is exactly one triple \( \langle q', a', \Delta \rangle \) with \( \langle\langle q, a\rangle, \langle q', a', \Delta \rangle\rangle \in \delta \).

Notation: We write \( \delta(q,a) \) for the unique triple \( \langle q', a', \Delta \rangle \) such that \( \langle\langle q, a\rangle, \langle q', a', \Delta \rangle\rangle \in \delta \).

December 21, 2018  B. Nebel, R. Mattmüller – AI Planning
7 / 29

Turing machine configurations

Definition (Configuration)

Let \( M = \langle \Sigma, \square, Q, q_0, q_Y, \delta \rangle \) be an NTM.

A configuration of \( M \) is a triple \( \langle w, q, x \rangle \in \Sigma^c \times Q \times \Sigma^c \).

- **w**: tape contents before tape head
- **q**: current state
- **x**: tape contents after and including tape head

December 21, 2018  B. Nebel, R. Mattmüller – AI Planning
8 / 29
Turing machine transitions

Definition (yields relation)
Let \( M = (\Sigma, \square, Q, q_0, q_Y, \delta) \) be an NTM.
A configuration \( c \) of \( M \) yields a configuration \( c' \) of \( M \), in symbols \( c \vdash c' \), as defined by the following rules, where \( a, a', b \in \Sigma \), \( w, x \in \Sigma^* \), \( q, q' \in Q \) and \( \langle q, a \rangle, \langle q', a', \Delta \rangle \in \delta \):

\[
\begin{align*}
\langle w, q, ax \rangle & \vdash \langle wa', q', x \rangle \quad \text{if } \Delta = +1, |x| \geq 1 \\
\langle w, q, a \rangle & \vdash \langle wa', q', \square \rangle \quad \text{if } \Delta = +1 \\
\langle wb, q, ax \rangle & \vdash \langle w, q', ba'x \rangle \quad \text{if } \Delta = -1 \\
\langle \varepsilon, q, ax \rangle & \vdash \langle \varepsilon, q', a'x \rangle \quad \text{if } \Delta = -1
\end{align*}
\]

Accepting configurations

Definition (accepting configuration, time)
Let \( M = (\Sigma, \square, Q, q_0, q_Y, \delta) \) be an NTM, let \( c = \langle w, q, x \rangle \) be a configuration of \( M \), and let \( n \in \mathbb{N}_0 \).
- If \( q = q_Y \) and \( M \) accepts \( c \) in time \( n \), then \( M \) accepts \( c \) in time \( n + 1 \).
- If \( q \neq q_Y \) and \( M \) accepts some \( c' \) with \( c \vdash c' \) in time \( n \), then \( M \) accepts \( c \) in time \( n + 1 \).

Definition (accepting configuration, space)
Let \( M = (\Sigma, \square, Q, q_0, q_Y, \delta) \) be an NTM, let \( c = \langle w, q, x \rangle \) be a configuration of \( M \), and let \( n \in \mathbb{N}_0 \).
- If \( q = q_Y \) and \( |w| + |x| \leq n \), \( M \) accepts \( c \) in space \( n \).
- If \( q \neq q_Y \) and \( M \) accepts some \( c' \) with \( c \vdash c' \) in space \( n \), then \( M \) accepts \( c \) in space \( n \).

Accepting words and languages

Definition (accepting words)
Let \( M = (\Sigma, \square, Q, q_0, q_Y, \delta) \) be an NTM.
\( M \) accepts the word \( w \in \Sigma^* \) in time (space) \( n \in \mathbb{N}_0 \)iff \( M \) accepts \( \langle \varepsilon, q_0, w \rangle \) in time (space) \( n \).
- Special case: \( M \) accepts \( \varepsilon \) in time (space) \( n \in \mathbb{N}_0 \)iff \( M \) accepts \( \langle \varepsilon, q_0, \square \rangle \) in time (space) \( n \).

Definition (accepting languages)
Let \( M = (\Sigma, \square, Q, q_0, q_Y, \delta) \) be an NTM, and let \( f : \mathbb{N}_0 \rightarrow \mathbb{N}_0 \).
\( M \) accepts the language \( L \subseteq \Sigma^* \) in time (space) \( f \)iff \( M \) accepts each word \( w \in L \) in time (space) \( f(|w|) \), and \( M \) does not accept any word \( w \notin L \) (in any time/space).
**Polynomial time and space classes**

Let $\mathcal{P}$ be the set of polynomials $p : \mathbb{N}_0 \to \mathbb{N}_0$ whose coefficients are natural numbers.

**Definition (P, NP, PSPACE, NPSPACE)**

- $P = \bigcup_{p \in \mathcal{P}} \text{DTIME}(p)$
- $NP = \bigcup_{p \in \mathcal{P}} \text{NTIME}(p)$
- $PSPACE = \bigcup_{p \in \mathcal{P}} \text{DSPACE}(p)$
- $NPSPACE = \bigcup_{p \in \mathcal{P}} \text{NSPACE}(p)$

**Theorem (complexity class hierarchy)**

$P \subseteq NP \subseteq PSPACE = NPSPACE$

**Proof.**

- $P \subseteq NP$ and $PSPACE \subseteq NPSPACE$ is obvious because deterministic Turing machines are a special case of nondeterministic ones.
- $NP \subseteq NPSPACE$ holds because a Turing machine can only visit polynomially many tape cells within polynomial time.
- $PSPACE = NPSPACE$ is a special case of a classical result known as Savitch’s theorem (Savitch 1970).

**Savitch’s theorem**

**Theorem**

For all $f \in \Omega(\log(n))$: $\text{NSPACE}(f) \subseteq \text{DSPACE}(f^2)$.

**Proof sketch.**

Let $C$ be the set of all configurations with $|C| = |Q| \cdot f(n) \cdot |\Sigma + 1|^{f(n)}$ and for each $c \in C$ we have $|c| = f(n)$. Then the following function checks whether a configuration $c'$ is reachable from $c$ by calling it with $\text{k_path}(c, c', |V|)$ using only $f(n)^2$ space.

```python
def k_path(s, t, k):
    if k = 0:
        return s = t
    if k = 1:
        return s = t or s != t
    for u in C:
        if k_path(s, u, (k/2)) and k_path(u, t, (k/2)):
            return true
```

**Complexity of propositional planning**
The propositional planning problem

**Definition (plan existence)**
The propositional plan existence problem (PlanEx) is the following decision problem:

**GIVEN:** Planning task $\Pi$
**QUESTION:** Is there a plan for $\Pi$?

$\Rightarrow$ decision problem analogue of satisficing planning

**Definition (bounded plan existence)**
The propositional bounded plan existence problem (PlanLen) is the following decision problem:

**GIVEN:** Planning task $\Pi$, length bound $K \in \mathbb{N}_0$
**QUESTION:** Is there a plan for $\Pi$ of length at most $K$?

$\Rightarrow$ decision problem analogue of optimal planning

Membership in PSPACE

**Theorem (PSPACE membership for PlanLen)**

$\text{PlanLen} \in \text{PSPACE}$

**Proof.**

Show $\text{PlanLen} \in \text{NPSPACE}$ and use Savitch’s theorem. Nondeterministic algorithm:

\[
\text{def plan}(\langle A, I, O, G \rangle, K):
    \begin{align*}
    & s := I \\
    & k := K \\
    & \text{while } s \not\models G: \\
    & \quad \text{guess } o \in O \\
    & \quad \text{fail if } o \text{ not applicable in } s \text{ or } k = 0 \\
    & \quad s := \text{app}(s) \\
    & \quad k := k - 1 \\
    & \text{accept}
    \end{align*}
\]

Hardness for PSPACE

**Idea:** generic reduction

- For an arbitrary fixed DTM $M$ with space bound polynomial $p$ and input $w$, generate planning task which is solvable iff $M$ accepts $w$ in space $p(|w|)$.
- For simplicity, restrict to TMs which never move to the left of the initial head position (no loss of generality).
Reduction: state variables

Let \( M = \langle \Sigma, \square, Q, q_0, q_Y, \delta \rangle \) be the fixed DTM and let \( p \) be its space-bound polynomial.

Given input \( w_1 \ldots w_n \), define relevant tape positions \( X := \{1, \ldots, p(n)\} \).

State variables

- \( \text{state}_q \) for all \( q \in Q \)
- \( \text{head}_i \) for all \( i \in X \cup \{0, p(n) + 1\} \)
- \( \text{content}_{i,a} \) for all \( i \in X, a \in \Sigma \)

\( \Rightarrow \) allows encoding a Turing machine configuration

Reduction: initial state

Let \( M = \langle \Sigma, \square, Q, q_0, q_Y, \delta \rangle \) be the fixed DTM and let \( p \) be its space-bound polynomial.

Given input \( w_1 \ldots w_n \), define relevant tape positions \( X := \{1, \ldots, p(n)\} \).

Initial state

Initially true:

- \( \text{state}_{q_0} \)
- \( \text{head}_1 \)
- \( \text{content}_{i,w} \) for all \( i \in \{1, \ldots, n\} \)
- \( \text{content}_{i,\square} \) for all \( i \in X \setminus \{1, \ldots, n\} \)

Initially false:

- all others

Reduction: operators

Let \( M = \langle \Sigma, \square, Q, q_0, q_Y, \delta \rangle \) be the fixed DTM and let \( p \) be its space-bound polynomial.

Given input \( w_1 \ldots w_n \), define relevant tape positions \( X := \{1, \ldots, p(n)\} \).

Operators

One operator for each transition rule \( \delta(q, a) = (q', a', \Delta) \) and each cell position \( i \in X \):

- precondition: \( \text{state}_q \land \text{head}_i \land \text{content}_{i,a} \)
- effect: \( \neg \text{state}_q \land \neg \text{head}_i \land \neg \text{content}_{i,a} \)
  \( \land \text{state}_{q'} \land \text{head}_{i+\Delta} \land \text{content}_{i,a'} \)

If \( q = q' \) and/or \( a = a' \), omit the effects on \( \text{state}_q \) and/or \( \text{content}_{i,a} \), to avoid consistency condition issues.

Reduction: goal

Let \( M = \langle \Sigma, \square, Q, q_0, q_Y, \delta \rangle \) be the fixed DTM and let \( p \) be its space-bound polynomial.

Given input \( w_1 \ldots w_n \), define relevant tape positions \( X := \{1, \ldots, p(n)\} \).

Goal

\( \text{state}_{q_Y} \)
Theorem (PSPACE-completeness; Bylander, 1994)

PlanEx and PlanLen are PSPACE-complete. This is true even when restricting to STRIPS tasks.

Proof.

Membership for PlanLen was already shown. Hardness for PlanEx follows because we just presented a polynomial reduction from an arbitrary problem in PSPACE to PlanEx. (Note that the reduction only generates STRIPS tasks.)

Membership for PlanEx and hardness for PlanLen follows from the polynomial reduction from PlanEx to PlanLen.

In addition to the basic complexity result presented in this chapter, there are many special cases, generalizations, variations and related problems studied in the literature:

- **different planning formalisms**
  - e.g., finite-domain representation, nondeterministic effects, partial observability, schematic operators, numerical state variables, state-dependent action costs
- **syntactic restrictions of planning tasks**
  - e.g., without preconditions, without conjunctive effects, STRIPS without delete effects
- **semantic restrictions of planning task**
  - e.g., restricting to certain classes of causal graphs
- **particular planning domains**
  - e.g., Blocksworld, Logistics, FreeCell

Some results for different planning formalisms:

- **FDR tasks**: same complexity as for propositional tasks ("folklore") also true for the SAS+ special case
- **schematic operators**: usually adds one exponential level to PlanEx complexity e.g., classical case EXPSPACE-complete (Erol et al., 1995)
- **numerical state variables**: undecidable in most variations (Helmert, 2002)
Propositional planning is PSPACE-complete.

The hardness proof is a polynomial reduction that translates an arbitrary polynomial-space DTM into a STRIPS task:
- Configurations of the DTM are encoded by propositional variables.
- Operators simulate transitions of the DTM.
- The DTM accepts an input if there is a plan for the corresponding STRIPS task.

This implies that there is no polynomial algorithm for classical planning unless P=PSPACE.

It also means that classical planning is not polynomially reducible to any problem in NP unless NP=PSPACE.