1 Motivation

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
- know what is not possible
- 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?
2 Background

- Turing machines
- Complexity classes

Turing machines

Complexity classes

Complexity of planning

More complexity results

Summary

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 \( \square \not\in \Sigma \)
- 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 \( \delta \subseteq (Q' \times \Sigma) \times (Q \times \Sigma \times \{-1, +1\}) \)

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 \( (q, a) \in Q' \times \Sigma \), there is exactly one triple \( (q', a', \Delta) \) with \( (q, a, (q', a', \Delta)) \) ∈ \( \delta \).

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

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 \( \langle w, q, x \rangle \in \Sigma^* \times Q \times \Sigma^* \).

- \( w \): tape contents before tape head
- \( q \): current state
- \( x \): tape contents after and including tape head
### 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 \((q, a), (q', a', \Delta)) \in \delta:

- \( \langle w, q, ax \rangle \vdash \langle wa', q', x \rangle \) if \( \Delta = +1, |x| \geq 1 \)
- \( \langle w, q, a \rangle \vdash \langle wa', q', \square \rangle \) if \( \Delta = +1 \)
- \( \langle wb, q, ax \rangle \vdash \langle w, q', ba'x \rangle \) if \( \Delta = -1 \)
- \( \langle \epsilon, q, ax \rangle \vdash \langle \epsilon, q', a'a'x \rangle \) if \( \Delta = -1 \)

### 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 \( n + 1 \).

**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 \( n \).

**Definition (accepting languages)**

Let \( M = (\Sigma, \square, Q, q_0, q_Y, \delta) \) be an NTM, and let \( f : \mathbb{N}_0 \to \mathbb{N}_0 \).

- Special case: \( M \) accepts \( \epsilon \) in time (space) \( n \in \mathbb{N}_0 \) if \( M \) accepts \( \langle \epsilon, q_0, \square \rangle \) in time (space) \( n \).

**Definition (accepting languages)**

Let \( M = (\Sigma, \square, Q, q_0, q_Y, \delta) \) be an NTM, and let \( f : \mathbb{N}_0 \to \mathbb{N}_0 \).

- \( M \) accepts the language \( L \subseteq \Sigma^* \) in time (space) \( f \) if \( M \) 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).

### Time and space complexity classes

**Definition (DTIME, NTIME, DSPACE, NSPACE)**

Let \( f : \mathbb{N}_0 \to \mathbb{N}_0 \).

- Complexity class \( \text{DTIME}(f) \) contains all languages accepted in time \( f \) by some DTM.
- Complexity class \( \text{NTIME}(f) \) contains all languages accepted in time \( f \) by some NTM.
- Complexity class \( \text{DSPACE}(f) \) contains all languages accepted in space \( f \) by some DTM.
- Complexity class \( \text{NSPACE}(f) \) contains all languages accepted in space \( f \) by some NTM.
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)

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

Polynomial complexity class relationships

Theorem (complexity class hierarchy)

$P \subseteq NP \subseteq PSPACE = NPSPACE$

Beweis.

$P \subseteq NP$ 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).

3 Complexity of propositional planning

Definition (plan existence)

The plan existence problem ($\text{PlanEx}$) is the following decision problem:

$\text{GIVEN:}$ Planning task $\Pi$
$\text{QUESTION:}$ Is there a plan for $\Pi$?

$\Rightarrow$ decision problem analogue of satisficing planning

Definition (bounded plan existence)

The bounded plan existence problem ($\text{PlanLen}$) is the following decision problem:

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

$\Rightarrow$ decision problem analogue of optimal planning
Plan existence vs. bounded plan existence

Theorem (reduction from PlanEx to PlanLen)

\[ \text{PlanEx} \leq_p \text{PlanLen} \]

\[ \text{PlanEx} \leq_p \text{PlanLen} \]

**Beweis.**
A propositional planning task with \( n \) state variables has a plan iff it has a plan of length at most \( 2^n - 1 \).

\[ \Rightarrow \] map instance \( \Pi \) of PlanEx to instance \( \langle \Pi, 2^n - 1 \rangle \) of PlanLen, where \( n \) is the number of state variables of \( \Pi \)

\[ \Rightarrow \] polynomial reduction

February 3th, 2016 B. Nebel, R. Mattmüller – AI Planning 20 / 32

Membership in PSPACE

Theorem (PSPACE membership for PlanLen)

\[ \text{PlanLen} \in \text{PSPACE} \]

**Beweis.**
Show \( \text{PlanLen} \in \text{NPSPACE} \) and use Savitch’s theorem.

Nondeterministic algorithm:

\[ \text{def} \ \text{plan}(\langle A, I, O, G \rangle, K): \]

\[ s := I \]

\[ k := K \]

\[ \text{while} \ s \not\models G: \]

\[ \ \ \text{guess} \ o \in O \]

\[ \ \ \text{fail if} \ o \ \text{not applicable in} \ s \ \text{or} \ k = 0 \]

\[ s := \text{app}_o(s) \]

\[ k := k - 1 \]

\[ \text{accept} \]

February 3th, 2016 B. Nebel, R. Mattmüller – AI Planning 21 / 32

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).

February 3th, 2016 B. Nebel, R. Mattmüller – AI Planning 22 / 32

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 \( q \in Q \)
- \( \text{head}, \) 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

February 3th, 2016 B. Nebel, R. Mattmüller – AI Planning 23 / 32
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:
- $\text{state}_{q_0}$
- $\text{head}_1$
- $\text{content}_{i,w_i}$ 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: $\neg \text{state}_q \land \neg \text{head}_i \land \neg \text{content}_{i,a}$
\quad $\land \text{state}_{q'} \land \text{head}_{i+\Delta} \land \text{content}_{i,a'}$

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.

PSPACE-completeness for STRIPS plan existence

Theorem (PSPACE-completeness; Bylander, 1994)

PLANEx and PLANLen are PSPACE-complete.
This is true even when restricting to STRIPS tasks.

Beweis.

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.
4 More complexity results

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
- **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

Complexity results for different planning formalisms

- **FDR tasks:**
  - same complexity as for propositional tasks (“folklore”)
  - also true for the SAS* special case
- **nondeterministic effects:**
  - fully observable: EXP-complete (Littman, 1997)
  - unobservable: EXPSPACE-complete (Haslum & Jonsson, 1999)
  - partially observable: 2-EXP-complete (Rintanen, 2004)
- **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 iff 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.