02285 AI and MAS, SP18
Multiagent planning and agent architectures

Curriculum for week 5: Sections 11.3.3 *Online replanning* and 11.4 *Multiagent planning* of Russell & Norvig 3ed + Chapter 2 of Michael Wooldridge's "Reasoning about rational agents" (available on CampusNet).

Todays subjects:

- Online planning.
- Multiagent planning.
- Agent architectures.
- The BDI agent architecture.

Programming project solution example 2016

http://www2.compute.dtu.dk/~tobo/AImuffins_on_MAsojourner.mov

Assumptions in classical planning

Assumptions in classical planning (PDDL):

1. Environments are **deterministic**.
2. Environments are **fully observable**.
3. Environments are **static** (no exogenous events).
4. Planning is **offline** (compute entire plan before starting to act).
5. Environments are **single-agent** (only one planning agent).

Today we will consider loosening assumptions 3, 4 and 5 (the most relevant to the programming project).

Offline planning vs. online replanning

Offline planning: Compute entire plan before executing first action (plan first, act later).

- **Ideal** when the domain is: **deterministic**, **static**, **fully observable**.
- **Problematic** in all other cases: one has to plan for **all contingencies** which easily leads to a combinatorial explosion.

Example. Box locations in hospital domain not known in advance.

A **better solution** is often to do **online (re)planning**: interleave planning and acting.

Even humans do seldom do pure offline planning, for the same reasons that AI agents shouldn't: planning for all eventualities can lead to unmanageable branching factors.

The necessity of replanning

Online replanning is in some cases our only option:

- The agent's model of the world (action schemas) might have **missing effects** or even **incorrect effects**. E.g. Shakey might leave a trace of oil behind itself when moving; or the effect of $TurnOn(x, r)$ might fail to appear if the light bulb in room r is blown.
- More generally, action descriptions can be **incorrect** in any number of ways, and only a replanning (and/or learning) agent can be robust against such problems.
- It can be impossible for the agent to model all possible **exogenous events** and their effects, e.g. unexpected phone calls or earthquakes.

Execution monitoring

Execution monitoring: Monitoring the execution of a computed plan in the case of online planning. Used to figure out when **replanning** is required.

Three levels of **execution monitoring** (from simple to more advanced):

- **Action monitoring**: when an action is about to be performed, check that all preconditions of the action still hold (e.g. by **sensing**).
- **Plan monitoring**: when an action is about to be performed, check that the remaining plan will still succeed (nothing has destroyed the possibility of carrying out the remaining actions of the plan).
- **Goal monitoring**: when an action is about to be performed, check to see if there is a better set of goals that the agent might try to achieve. This requires some degree of **intentionality**, which will be discussed further in the following.

Online replanning

Challenge: How to avoid that **replanning** means planning all over from scratch?

E.g. **partial-order planning** techniques are quite good, as they allow significant reuse in case of replanning.

Example. **(plan repair)**. Simple approach: whenever action monitoring calls for replanning, **repair** the plan by planning how to get back to a state from which the original plan can be continued.

Sometimes too naive: consider the case of the sphex wasp.

Multiagent planning

Multiagent planning problem: Planning problem with several planning agents.

Goals: Can be **individual**, **common** (e.g. players on the same team) or **conflicting** (e.g. players on opposing teams).

Basic multiagent concepts:

- **(Intelligent) agent**: Computer system/process capable of **independent** (**autonomous**) action in some environment—pursuing its own goals (**goal-directed behavior**).
- **Proactive agent**: Generating and seeking to achieve goals; not driven solely by external events; taking the **initiative**.
- **Multiagent system**: A system of **interacting (proactive) agents**. E.g. systems of hospital robots, softbots on the Internet, teams of soccer-playing robots or non-player characters in video games.
- Agents sharing the same environment (resources) will most often need to **cooperate**, **coordinate**, and **negotiate** with each other. Cf. human social interaction.

Single-agent multibody planning

Multibody planning problem: A single agent is planning on behalf of all agents. **Centralised** planning.

Multibody planning essentially reduces to single-agent planning (the other agents are reduced to "puppets").

What could be the advantages/disadvantages of multibody planning (centralised) compared to "true" multiagent planning (decentralised)?

Multiagent architecture: The arrangement of the agents and their protocols of interaction and collaboration in a multiagent system.

Example: A centralised planning agent distributing goals (jobs) to the other agents that then compute their own individual plans for achieving these goals.

Example: The Martha project

The **Martha project** (1998) studies a planning system for dock-worker robots.

Multiagent architecture of Martha:

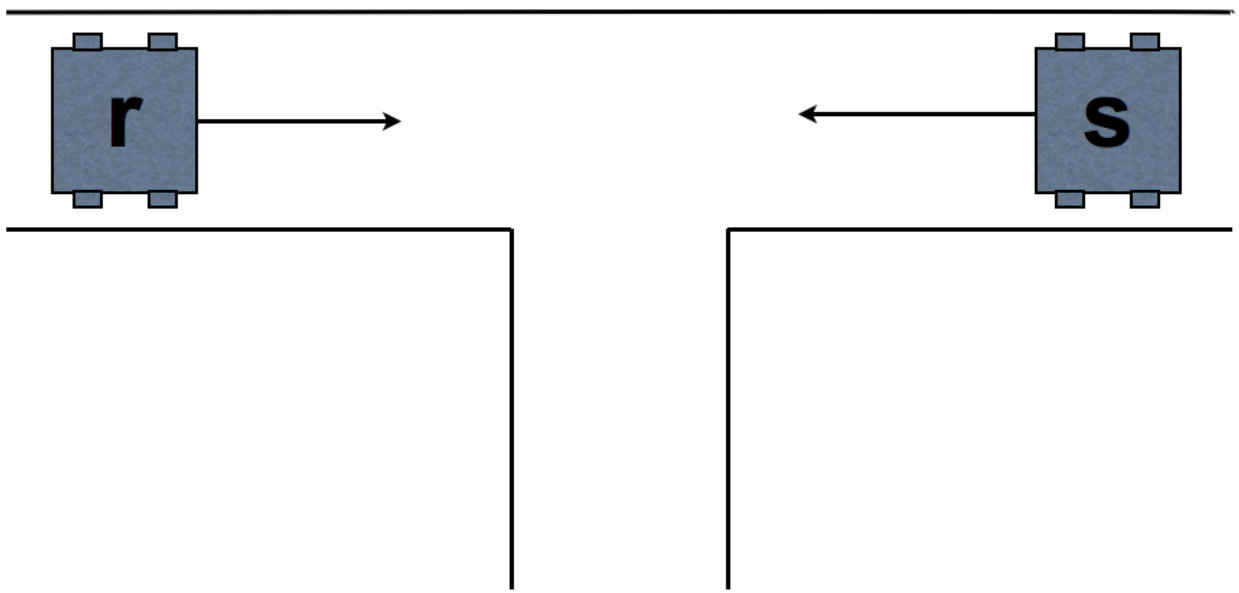
1. A **centralized planner** distributes transportation jobs to the individual dock-worker robots.
2. The robots then plan and schedule their own transportation jobs **individually**, not taking the other robots into account.
3. Each computed individual plan is **publicly broadcasted**.
4. In case of **resource conflicts**, the robot causing it will be asked to replan, see next slide.

Example cont'd

Resolving resource conflicts: Robot *r* broadcasts its computed plan. If robot *s* needs one of *r*'s scheduled resources, *s* will ask *r* to replan.

Problem: *r* might not be able to replan, e.g. in case of a **deadlock loop**.

In this case, the robot *r* will perform **centralised planning** (**multibody planning**) for the subset of robots involved in the conflict/deadlock loop.



Centralised vs. decentralised planning

Decentralised systems lead to a number of non-trivial challenges:

- **Communication**.
- **Coordination**.
- **Deadlock handling**.
- ...

So isn't a centralised system (multibody planning) the better choice?

No, the complexity of multibody planning **grows exponentially** with the number of agents planned for...

Example. Suppose a centralised planner plans on behalf of all dock-worker robots. It has to consider which **joint actions** $\langle a_1, \ldots, a_n \rangle$ involving the n robots has to be performed. If each robot has b different possible actions to perform, the branching factor of the search becomes b^n. Note the number of agents in the exponent!

When the agents' tasks are **loosely coupled** as in the Martha project, this is unnecessarily expensive.

Example cont'd

In the Martha project, the distribution of planning tasks reduces the complexity to **almost linear** in the number of robots (up to at least 50 robots in total).

Distributing planning tasks between agents is an example of **problem decomposition**. Relates to **partial-order planning** and **hierarchical planning**.

General approach underlying Martha project:

- First "pretend" the individual agents' problems are completely decoupled (a **relaxed problem**).
- Solve the individual problems separately.
- Fix up the interactions (**conflicts**) by communication and coordination.

Joint plans in multiagent planning

Consider the **milk-and-bananas problem** involving two agents, A(nn) and B(ob). The common goal of the two agents is to have milk and bananas.

A **joint plan** that would work (assuming **perfect synchronization**):

$$A : Go(H, N), Buy(M), Buy(B), Go(N, H)$$
$$B : WatchTV, WatchTV, WatchTV, WatchTV$$

Ann, however, might have something more along the following lines in mind:

$$A : WatchTV, WatchTV, WatchTV, WatchTV$$
$$B : Go(H, N), Buy(M), Buy(B), Go(N, H)$$

Deciding which joint plan to choose requires either **communication** ("you do it, Bob!") or some kind of agreed **convention** or **social law** (Bob always does the shopping on Tuesdays).

Example. Collision avoidance when driving vs. when walking.

Task sharing in multiagent planning

The **optimal plan** for Ann and Bob in terms of number of joint actions is:

$$A : Go(H, N), Buy(M), Go(N, H)$$
$$B : Go(H, N), Buy(B), Go(N, H)$$

It might be achieved through **negotiation** ("I'll go if you come along").

Task sharing: How to share tasks between agents through **communication** and **coordination**.

Common methods for **task sharing**:

- **Negotiation**.
- **Bidding**.
- **Voting**.
- **Conventions** and **social laws**.

Multibody planning

All single-agent planning algorithms presented in this course can with only minor modifications be used for centralised **multibody planning**.

One should however be aware of the potential **exponential blowup** of complexity in the number of agents.

Partial-order planning can work well in multibody planners for loosely coupled multiagent problems. Why?

Agent architectures

In complex dynamic multiagent environments, agents have to be more than just planners.

We must consider more complex **architectures** for agents. In these, agents are build from many independent **computational modules**, of which the planning engine is just one.

Definition of **agent architecture** from Patti Maes (1991):

> *A particular methodology for building agents. It specifies how [...] the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. [...] An agent architecture encompasses techniques and algorithms that support this methodology.*

One such agent architecture is the **BDI architecture**, that we will describe in the following.

Deliberation and means-end reasoning

In pure planning, an agent is given a **predefined goal** to be achieved.

Planners are thus suitable for finding out **how** to do something, but not for finding out **what** to do.

- **Deliberation**: The process of finding out **what** state of affairs (which goal) we want to achieve.
- **Means-end reasoning**: The process of deciding **how** to achieve this state of affairs

A planner does means-end reasoning but not deliberation.

A **proactive agent** will both need a deliberation module and a means-end reasoning module (planner).

Intentions and the intentional stance

- The output of a **deliberation process** is **intentions**: the state of affairs that the agent has **commited** itself to achieving. E.g. buying milk and bananas.
- The output of a **means-end reasoning process** is a **plan**: how to achieve the intentions. E.g. a plan for buying milk and bananas.

The intentional stance: Describing agents in terms of their beliefs, intentions, desires, hopes, etc.

The intentional stance is a very **powerful abstraction**: It provides us with a convenient and familiar way of describing, explaining, and predicting the behaviour of complex systems.

It also helps in **designing** such AI systems (in analogy to how the **object** abstraction helps us designing large computer programs).

Example. Taking the intentional stance towards a thermostat will perhaps not buy us much, but e.g. in the dock-worker environment it will.

Intentions

Since **intentions** are goals that an agent commits to, if it has the intention ϕ then usually the following should be the case:

- The agent should **devote resources** to obtaining ϕ.
- The agent should not adopt an intention in **conflict** with ϕ (e.g. eat the banana and drink the milk on the way back from Netto).
- **Persistence**: If the agent's first attempt on achieving ϕ fails, it should **try again** (possibly using a different plan).
- The agent should **believe** that ϕ is possible to achieve, and that under "normal circumstances" it will in fact eventually achieve ϕ.

Intentions vs. desires

Note that intentions are stronger than desires: **desires** are everything we would like to achieve in an "ideal world" (be young, rich, famous, beautiful, happy, etc.), and some of these might be unattainable or mutually exclusive.

In contrast, **intentions** are desires that we have already chosen to commit to (e.g. complete ones studies in the hope that this is a step on the way to become rich).

The BDI architecture: version 1

The **BDI architecture** is an agent architecture based on the 3 attitudes **b**elief, **d**esire, and **i**ntention. An agent designed by the BDI architecture will continously be engaged in an **agent control loop** where the agent alternately does deliberation and means-end reasoning.

First version of the **agent control loop**:

```
Agent Control Loop Version 1
1. while true do
2.    perceive the world;
3.    update beliefs about the world;
4.    deliberate about what intention to achieve next;
5.    use means-ends reasoning to compute
          a plan for the intention;
6.    execute the plan
7. end while
```

Steps 5 and 6 corresponds to what we have already been covering on automated planning.

BDI formal notation

We will now make the agent control loop a bit more formal. To be able to do this, we must first introduce some new notation:

- B: **current beliefs**. $B \in Bel$: the set of **all possible beliefs**.
- D: **current desires**. $D \in Des$: the set of **all possible desires**.
- I: **current intentions**. $I \in Int$: the set of **all possible intentions**.
- ρ: a **percepts**. $\rho \in Per$: the set of **all possible percepts**.
- π: **plan**. $\pi \in Plan$: the set of **all possible plans**. Plans are sequences of actions.
- $hd(\pi)$: **first action** of the plan π.
- $tail(\pi)$: **remaining sequence of actions** in π.
- $execute(\pi)$: the procedure executing π.

BDI formal notation (cont'd)

- **Belief revision function**:

$$brf : \mathbb{P}(Bel) \times Per \rightarrow \mathbb{P}(Bel)$$

$brf(B, \rho)$ is your updated beliefs based on current beliefs B and percept ρ.

- **Deliberation function**:

$$deliberate : \mathbb{P}(Bel) \rightarrow \mathbb{P}(Int)$$

$deliberate(B)$ is the set of intentions chosen based on current beliefs B.

- **Planning function**:

$$plan : \mathbb{P}(Bel) \times \mathbb{P}(Int) \rightarrow Plan$$

$plan(B, I)$ is the plan chosen based on current beliefs B and current intentions I.

The BDI architecture: version 2

Using the notations introduced above, we can now express the agent control loop in a slightly more formal way:

```
Agent Control Loop Version 2
1. B := B_0 /* initial beliefs */
2. while true do
3.    get next percept ρ;
4.    B := brf(B, ρ);
5.    I := deliberate(B);
6.    π := plan(B, I);
7.    execute(π)
8. end while
```

Deliberation

The deliberation step (step 5 in the algorithm above) often splits into two parts:

1. Generate the **desires**, that is, everything the agent might want to achieve.
2. Generate the **intentions** by choosing between competing desires and commit to some of them.

First part is called **option generation**:

$$\mathit{options} : \mathbb{P}(\mathit{Bel}) \times \mathbb{P}(\mathit{Int}) \rightarrow \mathbb{P}(\mathit{Des})$$

Second part is called **filtering**:

$$\mathit{filter} : \mathbb{P}(\mathit{Bel}) \times \mathbb{P}(\mathit{Des}) \times \mathbb{P}(\mathit{Int}) \rightarrow \mathbb{P}(\mathit{Int})$$

The BDI architecture: version 3

Splitting the **deliberation step** into an **option generation step** followed by a **filtering step**, we now get the following agent loop:

```
Agent Control Loop Version 3
 1. B := B_0 /* initial beliefs */
 2. I := I_0  /* initial intentions */
 3. while true do
 4.    get next percept ρ;
 5.    B := brf(B, ρ);
 6.    D := options(B, I);
 7.    I := filter(B, D, I);
 8.    π := plan(B, I);
 9.    execute(π)
10. end while
```

Blind commitment vs. plan monitoring

Weakness of current agent control loop: The agent is **blindly commited** to both its **ends** (intentions) and its **means** (plans).

In other words, the agent is doing **offline deliberation** and **offline planning**.

This is insufficient if the domain is **partially observable**, **dynamic** (e.g. has other agents) or **nondeterministic**.

First step towards **online (re)planning** is to introduce **plan monitoring**: **Replan** if current plan is no longer **sound** given the agent's intentions and beliefs.

We use the notation $sound(\pi, I, B)$ to mean that π is a **sound plan** for achieving I given beliefs B.

Version 4 of the agent control loop integrates this type of **plan monitoring**...

The BDI architecture: version 4

```
Agent Control Loop Version 4
 1. B := B_0; I := I_0;
 2. while true do
 3.    get next percept ρ;
 4.    B := brf(B, ρ); D := options(B, I); I := filter(B, D, I);
 5.    π := plan(B, I);
 6.    while ¬empty(π) do
 7.       execute(hd(π));
 8.       π := tail(π);
 9.       get next percept ρ;
10.       B := brf(B, ρ);
11.       if not sound(π, I, B) then
12.          π := plan(B, I)
13.       end-if
14.    end-while
15. end while
```

Goal monitoring

In addition to **plan monitoring** above, we can introduce **goal monitoring**.

Basic goal monitoring. Dismiss current plan if:

- It turns out that the intentions (goals) have already been achieved, which we denote $\mathit{succeeded}(I, B)$.
- It turns out that the intentions (goals) are impossible to obtain, which we denote $\mathit{impossible}(I, B)$.

Version 5 of the agent control loop integrates this type of **goal monitoring**...

The BDI architecture: version 5

Agent Control Loop Version 5

1. $B := B_0; I := I_0;$
2. while true do
3. get next percept ρ;
4. $B := brf(B, \rho); D := options(B, I); I := filter(B, D, I);$
5. $\pi := plan(B, I);$
6. while $\neg empty(\pi) \wedge \neg succeeded(I, B) \wedge \neg impossible(I, B)$ do
7. $execute(hd(\pi));$
8. $\pi := tail(\pi);$
9. get next percept ρ;
10. $B := brf(B, \rho);$
11. if not $sound(\pi, I, B)$ then
12. $\pi := plan(B, I)$
13. end-if
14. end-while
15. end while

Intention reconsideration

Our BDI-agent only reconsiders its intentions once every time around the outer control loop, i.e., when:

- it has completely executed a plan to achieve its current intentions; or
- it believes it has achieved its current intentions; or
- it believes its current intentions are no longer possible.

In other words, we now have **online (re)planning** but still not **online deliberation**.

Not sufficiently general: Assume my **desire** is to have dinner in London. I might thus form the **intention** of finding a nice and cheap restaurant. I then form a **plan** for searching for such a restaurant. After some hours of unsuccessful search, I might want to reconsider my intention...

Version 6 of the agent control loop integrates **online deliberation**...

The BDI architecture: version 6

Agent Control Loop Version 6

1. $B := B_0; I := I_0;$
2. while true do
3. get next percept ρ;
4. $B := brf(B, \rho); D := options(B, I); I := filter(B, D, I);$
5. $\pi := plan(B, I);$
6. while $\neg empty(\pi) \wedge \neg succeeded(I, B) \wedge \neg impossible(I, B)$ do
7. $execute(hd(\pi));$
8. $\pi := tail(\pi);$
9. get next percept ρ;
10. $B := brf(B, \rho); D := options(B, I); I := filter(B, D, I);$
11. if not $sound(\pi, I, B)$ then
12. $\pi := plan(B, I)$
13. end-if
14. end-while
15. end while

The problem of intention reconsideration

Another problem: intention reconsideration is computationally costly!

Dilemma:

- Reconsidering intentions **too seldom**: the agent might continue pursuing its intentions after it should be clear that the intentions are no longer relevant.
- Reconsidering intentions **too often**: the agent may spend too much time reconsidering, and never achieve any of its intentions.

Possible solution: Incorporate an explicit meta-level control component, $\mathit{reconsider}(I, B)$, that decides whether or not to reconsider the intentions.

To make any sense, computing $\mathit{reconsider}(I, B)$ must be fairly cheap compared to doing the actual reconsideration of intentions.

Version 7 of the agent control loop integrates this...

Agent Control Loop Version 7

1. $B := B_0; I := I_0;$
2. while true do
3. get next percept ρ;
4. $B := brf(B, \rho); D := options(B, I); I := filter(B, D, I);$
5. $\pi := plan(B, I);$
6. while $\neg empty(\pi) \wedge \neg succeeded(I, B) \wedge \neg impossible(I, B)$ do
7. $execute(hd(\pi));$
8. $\pi := tail(\pi);$
9. get next percept ρ;
10. $B := brf(B, \rho);$
11. if $reconsider(I, B)$ then
12. $D := options(B, I); I := filter(B, D, I)$
13. end-if
14. if not $sound(\pi, I, B)$ then
15. $\pi := plan(B, I)$
16. end-if
17. end-while
18. end while

Summary and final notes on BDI

- In **dynamic (multiagent) environments**, pure (offline) planning is insufficient. The planner must be embedded in a more complex **agent architecture** as e.g. the **BDI architecture**.
- The BDI architecture is centered around the attitudes **belief**, **desire**, and **intention**.
- The BDI architecture can be presented as an **agent control loop**, which can be designed more or less complex depending on the application (version 1–7).
- The BDI architecture is not a fixed algorithm, but a **conceptual model** that can assist the design of agents.
- The origin of the BDI model is a theory of **human practical reasoning** by the philosopher Michael Bratman (1987).

BDI agent application examples

Examples of **implementations** of the BDI architecture:

- The air-traffic management system at one of Sydney's airports.
- At DTU, in a multiagent system consisting of a number of Lego NXT robots solving transportation tasks in a shared environment.
- Some DTU teams in the programming project of this course (using e.g. Jason).