02285 AI and MAS, SP18
Domain-independendent heuristics

Curriculum for week 3: Section 10.2.3 in Russell & Norvig, Sections 2.5 and 2.7 in Geffner & Bonet except the subsection "Relaxed Plan Heuristics".

Heuristics functions in planning

- $cost(a, s)$: The **cost** of executing a in s. In planning, normally $cost(a, s) = 1$ for all actions (also in these slides).

Heuristics functions in planning

- $cost(a, s)$: The **cost** of executing a in s. In planning, normally $cost(a, s) = 1$ for all actions (also in these slides).
- Let $P = (\mathcal{A}, s_0, g)$ be a planning problem. For any state s reachable from s_0, $P(s)$ refers to the planning problem $(\mathcal{A}, s, g)$.

Heuristics functions in planning

- $cost(a, s)$: The **cost** of executing a in s. In planning, normally $cost(a, s) = 1$ for all actions (also in these slides).
- Let $P = (\mathcal{A}, s_0, g)$ be a planning problem. For any state s reachable from s_0, $P(s)$ refers to the planning problem $(\mathcal{A}, s, g)$.
- The **(optimal) cost** of $P(s)$, denoted $h^*_P(s)$ (or $h^*(s)$ when P is clear from the context) is the length of a shortest solution to $P(s)$ (the number of actions in the shortest plan leading from s to g).

Heuristics functions in planning

- $cost(a, s)$: The **cost** of executing a in s. In planning, normally $cost(a, s) = 1$ for all actions (also in these slides).
- Let $P = (\mathcal{A}, s_0, g)$ be a planning problem. For any state s reachable from s_0, $P(s)$ refers to the planning problem $(\mathcal{A}, s, g)$.
- The **(optimal) cost** of $P(s)$, denoted $h^*_P(s)$ (or $h^*(s)$ when P is clear from the context) is the length of a shortest solution to $P(s)$ (the number of actions in the shortest plan leading from s to g).
- A **heuristic function** (or simply **heuristics**) for P is a mapping h from (reachable) states into natural numbers satisfying:
  1. $h(s)$ estimates/approximates the optimal cost $h^*_P(s)$.
  2. $h(s)$ is cheaper to compute than $h^*_P(s)$. Why this requirement? Computing h^*_P essentially means solving P.

Heuristics functions in planning

- $cost(a, s)$: The **cost** of executing a in s. In planning, normally $cost(a, s) = 1$ for all actions (also in these slides).
- Let $P = (\mathcal{A}, s_0, g)$ be a planning problem. For any state s reachable from s_0, $P(s)$ refers to the planning problem $(\mathcal{A}, s, g)$.
- The **(optimal) cost** of $P(s)$, denoted $h^*_P(s)$ (or $h^*(s)$ when P is clear from the context) is the length of a shortest solution to $P(s)$ (the number of actions in the shortest plan leading from s to g).
- A **heuristic function** (or simply **heuristics**) for P is a mapping h from (reachable) states into natural numbers satisfying:
    1. $h(s)$ estimates/approximates the optimal cost $h^*_P(s)$.
    2. $h(s)$ is cheaper to compute than $h^*_P(s)$. Why this requirement? Computing h^*_P essentially means solving P.
- Heuristic functions can be used in best-first search algorithms like $A^\star$, $WA^\star$ and greedy best-first search.

Admissible and dominating heuristics

Assume given a heuristics h for a problem P with $h = h^*_P$. How many states will be generated by a greedy best-first search with heuristics h?

$O(n \cdot b)$ where n is length of of shortest solution, and b is branching factor (cf. last weeks exercises).

Admissible and dominating heuristics

Assume given a heuristics h for a problem P with $h = h_P^*$. How many states will be generated by a greedy best-first search with heuristics h? $O(n \cdot b)$ where n is length of of shortest solution, and b is branching factor (cf. last weeks exercises).

- A heuristics h for P is **admissible** if $h(s) \leq h_P^*(s)$ for all states s. Are inadmissible heuristics useless? No, but we can't then guarantee optimality of $A^\star$.

Admissible and dominating heuristics

Assume given a heuristics h for a problem P with $h = h^*_P$. How many states will be generated by a greedy best-first search with heuristics h? $O(n \cdot b)$ where n is length of of shortest solution, and b is branching factor (cf. last weeks exercises).

- A heuristics h for P is **admissible** if $h(s) \leq h^*_P(s)$ for all states s. Are inadmissible heuristics useless? No, but we can't then guarantee optimality of $A^\star$.
- A heuristics h_2 is said to **dominate** a heuristics h_1 if $h_2(s) \geq h_1(s)$ for all s (R&N Chapter 3).

Admissible and dominating heuristics

Assume given a heuristics h for a problem P with $h = h^*_P$. How many states will be generated by a greedy best-first search with heuristics h? $O(n \cdot b)$ where n is length of of shortest solution, and b is branching factor (cf. last weeks exercises).

- A heuristics h for P is **admissible** if $h(s) \leq h^*_P(s)$ for all states s. Are inadmissible heuristics useless? No, but we can't then guarantee optimality of $A^\star$.
- A heuristics h_2 is said to **dominate** a heuristics h_1 if $h_2(s) \geq h_1(s)$ for all s (R&N Chapter 3).
- If h_2 dominates h_1 and both are admissible, $A^\star$ will never expand more nodes using h_2 than using h_1.

Admissible and dominating heuristics

Assume given a heuristics h for a problem P with $h = h^*_P$. How many states will be generated by a greedy best-first search with heuristics h? $O(n \cdot b)$ where n is length of of shortest solution, and b is branching factor (cf. last weeks exercises).

- A heuristics h for P is **admissible** if $h(s) \leq h^*_P(s)$ for all states s. Are inadmissible heuristics useless? No, but we can't then guarantee optimality of $A^\star$.
- A heuristics h_2 is said to **dominate** a heuristics h_1 if $h_2(s) \geq h_1(s)$ for all s (R&N Chapter 3).
- If h_2 dominates h_1 and both are admissible, $A^\star$ will never expand more nodes using h_2 than using h_1.
- If $h_1, \ldots, h_n$ are admissible heuristics, then so is $h(s) = \max\{h_1(s), \ldots, h_n(s)\}$. Is it then always better to use h as heuristics than either of the h_i? Yes, unless the penalty in computation time of h is too high.

Relaxed problems

Most heuristics are generated via *relaxed problems*.

Relaxed problem: A simplified version of a problem with fewer restrictions. A solution to the original problem should also be a solution to the relaxed problem.

Example. Relaxing the sliding puzzle game (R&N Chapter 3):

1. A tile can move to any adjacent square.
2. A tile can move to any square.

Generating heuristics via relaxed problems

Given any problem P and relaxation P', an admissible heuristics h for P can defined by:

$$h(s) = h^*_{P'}(s).$$

In words: The **estimated cost** of a solution to the **real problem** is taken to be the **actual cost** of a solution to the **relaxed problem**.

1. Why is h defined above admissible? Because any solution to P is also a solution to P'. So $h(s) = h^*_{P'}(s) \leq h^*_P(s)$.

Generating heuristics via relaxed problems

Given any problem P and relaxation P', an admissible heuristics h for P can defined by:

$$h(s) = h^*_{P'}(s).$$

In words: The **estimated cost** of a solution to the **real problem** is taken to be the **actual cost** of a solution to the **relaxed problem**.

1. Why is h defined above admissible? Because any solution to P is also a solution to P'. So $h(s) = h^*_{P'}(s) \leq h^*_P(s)$.
2. Which heuristics do we get from the sliding puzzle relaxations of the previous slide (1. Move to any adjacent square; 2. Move to any square)? 1. Sum of Manhattan distances of tiles to goal positions. 2. Number of misplaced tiles (goal count).

Generating heuristics via relaxed problems

Given any problem P and relaxation P', an admissible heuristics h for P can defined by:

$$h(s) = h^*_{P'}(s).$$

In words: The **estimated cost** of a solution to the **real problem** is taken to be the **actual cost** of a solution to the **relaxed problem**.

1. Why is h defined above admissible? Because any solution to P is also a solution to P'. So $h(s) = h^*_{P'}(s) \leq h^*_P(s)$.
2. Which heuristics do we get from the sliding puzzle relaxations of the previous slide (1. Move to any adjacent square; 2. Move to any square)? 1. Sum of Manhattan distances of tiles to goal positions. 2. Number of misplaced tiles (goal count).
3. Does one of the heuristics from the previous question dominate the other? Yes, the heuristics from 1 dominates the one from 2.

Generating heuristics via relaxed problems

Given any problem P and relaxation P', an admissible heuristics h for P can defined by:

$$h(s) = h^*_{P'}(s).$$

In words: The **estimated cost** of a solution to the **real problem** is taken to be the **actual cost** of a solution to the **relaxed problem**.

1. Why is h defined above admissible? Because any solution to P is also a solution to P'. So $h(s) = h^*_{P'}(s) \leq h^*_P(s)$.
2. Which heuristics do we get from the sliding puzzle relaxations of the previous slide (1. Move to any adjacent square; 2. Move to any square)? 1. Sum of Manhattan distances of tiles to goal positions. 2. Number of misplaced tiles (goal count).
3. Does one of the heuristics from the previous question dominate the other? Yes, the heuristics from 1 dominates the one from 2.
4. Given P', how do we calculate $h^*_{P'}$? Using breadth-first search on P'.

Types of problem relaxations

Two types of problem relaxations:

1. **Adding edges**: Add edges to state space, making it easier to reach goal.
2. **Reducing number of nodes**: Group nodes together (abstraction) or disregard certain nodes, making the state space smaller.

Give examples of each type of relaxation for the problem considered in the warmup assignment.

Relaxed problems and domain-independent heuristics

Key development in modern planning research: Domain-independent heuristics obtained from **automatic** problem relaxations.

Example. Recall the **goal count** heuristics from last week:

$$h_{gc}(s) = \text{number of goal literals } \textit{not} \text{ satisfied in } s.$$

Note that for the simplified gripper problem we have $h_{gc} = h^*$ (the heuristics is optimal).

Is this heuristics admissible for all planning problems? No. Some actions can achieve several goal literals at the same time. E.g. a block-pushing agent that can push two blocks at a time.

In the following we introduce several further (and better) heuristics induced by relaxed problems: **ignore preconditions heuristics**, **delete-relaxation heuristics**, **additive heuristics** and **max heuristics**.

Ignore preconditions heuristics

Ignore preconditions heuristics: Relax problem by ignoring all preconditions. Thus every action becomes applicable in every state.

ACTION: $MoveAB(x)$
PRECONDITION: $\cancel{Box(x)} \land \cancel{In(x, A)}$
EFFECT: $In(x, B) \land \neg In(x, A)$

ACTION: $Buy(x, y)$
PRECONDITION: $\cancel{Buyable(x)} \land \cancel{Place(y)} \land \cancel{At(y)} \land \cancel{Sells(y, x)}$
EFFECT: $Have(x)$

How will this heuristics work on simplified Gripper problem? Quite bad. Calculating it takes just as long as solving the original problem.

Ignore preconditions heuristics

Ignore preconditions heuristics: Relax problem by ignoring all preconditions. Thus every action becomes applicable in every state.

ACTION: $MoveAB(x)$
PRECONDITION: $\cancel{Box(x)} \wedge \cancel{In(x, A)}$
EFFECT: $In(x, B) \wedge \neg In(x, A)$

ACTION: $Buy(x, y)$
PRECONDITION: $\cancel{Buyable(x)} \wedge \cancel{Place(y)} \wedge \cancel{At(y)} \wedge \cancel{Sells(y, x)}$
EFFECT: $Have(x)$

How will this heuristics work on simplified Gripper problem? Quite bad. Calculating it takes just as long as solving the original problem.

And what about the milk-and-bananas problem? A bit better, but a rather expensive way to calculate the goal count heuristics.

Ignore preconditions and non-goal literals heuristics

Ignore preconditions and non-goal literals heuristics h_{ip}: Relax by ignoring all preconditions **and** all effect literals except those occurring in the goal.

$$\begin{array}{|l|}\hline \textsc{Action}: \; \mathit{MoveAB}(x) \\ \textsc{Precondition}: \; \cancel{\mathit{Box}(x)} \wedge \cancel{\mathit{In}(x, A)} \\ \textsc{Effect}: \; \mathit{In}(x, B) \wedge \cancel{\neg \mathit{In}(x, A)} \\ \hline \end{array}$$

Do we have $h_{ip} = h_{gc}$, that is, is the new heuristics equivalent to the goal count heuristics that simply counts the number of unsatisfied goal literals? No. Some actions can achieve several goal literals, and some actions might undo the effect of others.

Ignore preconditions and non-goal literals heuristics

Ignore preconditions and non-goal literals heuristics h_{ip}: Relax by ignoring all preconditions **and** all effect literals except those occurring in the goal.

ACTION: $MoveAB(x)$
PRECONDITION: $\cancel{Box(x)} \wedge \cancel{In(x, A)}$
EFFECT: $In(x, B) \wedge \cancel{\neg In(x, A)}$

Do we have $h_{ip} = h_{gc}$, that is, is the new heuristics equivalent to the goal count heuristics that simply counts the number of unsatisfied goal literals? No. Some actions can achieve several goal literals, and some actions might undo the effect of others.

h_{gc} was not admissible. Is h_{ip}? Yes.

Remark. Calculating $h_{ip}(s)$ is a **set-cover problem** (given set U and set of sets S, find the smallest subset $C \subseteq S$ s.t. $\bigcup C = U$). Set-covering is **NP-hard**, but a tractable greedy approximation exists (however, we then loose admissibility).

Delete-relaxation heuristics

Delete-relaxation heuristics (called **ignore delete list heuristics** in R&N) h^+: Relax problem by removing all negative literals from the effects of actions (equivalently: set all delete lists to $\emptyset$).

ACTION: $Go(x, y)$
PRECONDITION: $Place(x) \wedge Place(y) \wedge At(x)$
EFFECT: $At(y) \wedge \cancel{\neg At(x)}$

Only **admissible** if all *goals* and *preconditions* only contain positive literals, that is, atoms. Why? A negative goal will never be found by this heuristics.

Easy to ensure: Any planning problem P can be translated into an equivalent planning problem without negative literals in these places. How? Replace every negative literal $\neg A$ by a new positive literal $\overline{A}$.

Unfortunately, calculating h^+ is **NP-hard**. We need more ideas...

Additive heuristics

Additive heuristics h_{add}: Relax problem by

1. removing negative literals from effects (as in delete-relaxation); and
2. assume **subgoal independence**.

Subgoal independence assumption: The cost of achieving a goal $g_1 \wedge g_2 \wedge \cdots \wedge g_n$ is equal to the sum of the costs of achieving each of the g_i.

Letting $h^*(g, s)$ denote the optimal cost of achieving g from s, the subgoal independence assumption amounts to assuming $h^*(\bigwedge_i g_i, s) = \sum_i h^*(g_i, s)$.

Give an example of a planning problem where the subgoal indepence assumption is **optimistic**, that is, where $\sum_i h^*(g_i, s) < h^*(\bigwedge_i g_i, s)$.
Goal is to achieve p and q, and there is one action to achieve p but make q false, and another action to achieve q but make p false.

Additive heuristics

Additive heuristics h_{add}: Relax problem by

1. removing negative literals from effects (as in delete-relaxation); and
2. assume **subgoal independence**.

Subgoal independence assumption: The cost of achieving a goal $g_1 \wedge g_2 \wedge \cdots \wedge g_n$ is equal to the sum of the costs of achieving each of the g_i.

Letting $h^*(g, s)$ denote the optimal cost of achieving g from s, the subgoal independence assumption amounts to assuming $h^*(\bigwedge_i g_i, s) = \sum_i h^*(g_i, s)$.

Give an example of a planning problem where the subgoal indepence assumption is **optimistic**, that is, where $\sum_i h^*(g_i, s) < h^*(\bigwedge_i g_i, s)$.
Goal is to achieve p and q, and there is one action to achieve p but make q false, and another action to achieve q but make p false.

Give an example of a planning problem where the subgoal indepence assumption is **pessimistic**, that is, where $\sum_i h^*(g_i, s) > h^*(\bigwedge_i g_i, s)$.
Goal is to achieve p and q, and there is one action that achieves both.

Additive heuristics cont'd

Since the subgoal independence assumption is sometimes pessimistic, the additive heuristics is **not admissible**.

But the good news is: it is **tractable**.

Conventions for the remaining slides:

- All planning problems have only positive literals (atoms) in goals and preconditions (without loss of generality).
- We use symbols $p, p_1, p_2, \ldots$ to denote ground atoms (e.g. $In(1, A)$ or $Have(Milk)$).
- Recall from last week that $\textsc{Add}(a)$ denotes the set of positive literals in the effect of the action a. In a delete-relaxation, $\textsc{Add}(a)$ contains **all** effects of a.

Additive heuristics cont'd

The additive heuristics h_{add} for a planning problem $P = (\mathcal{A}, s_0, g)$ can be expressed quite neatly using a recursive definition. It is non-trivial to compute, however.

$$
\begin{aligned}
h_{add}(s) &= h_{add}(g, s) \qquad \text{(Note the overloading of } h_{add}\text{)} \\
h_{add}(\textstyle\bigwedge_i p_i, s) &= \textstyle\sum_i h_{add}(p_i, s) \qquad \text{(subgoal independence ass.)} \\
h_{add}(p, s) &= \begin{cases} 0 & \text{if } p \in s \\ \infty & \text{if } p \notin s \text{ and for every action } a,\ p \notin \textsc{Add}(a) \\ 1 + \min\{h_{add}(\textsc{Precond}(a), s) \mid p \in \textsc{Add}(a)\} & \\ & \text{otherwise} \end{cases}
\end{aligned}
$$

Note how the subgoal independence assumption is treated by the second clause for h_{add} above, and delete-relaxation is treated in the last clause (we only consider $\textsc{Add}$, not all effects).

Additive heuristics example

We consider the simplified Gripper problem from last week with action schemas

ACTION: $MoveAB(x)$
PRECONDITION: $Box(x) \wedge In(x, A)$
EFFECT: $In(x, B) \wedge \cancel{\neg In(x, A)}$

ACTION: $MoveBA(x)$
PRECONDITION: $Box(x) \wedge In(x, B)$
EFFECT: $In(x, A) \wedge \cancel{\neg In(x, B)}$

Init state is $s_0 = \bigwedge_{i=1,\dots,n} In(i, A)$ and goal is $g = \bigwedge_{i=1,\dots,n} In(i, B)$. Then:

$$h_{add}(g, s_0) = h_{add}(\bigwedge_{i=1,\dots,n} In(i, B), s_0) = \sum_{i=1,\dots,n} h_{add}(In(i, B), s_0).$$

For each $i = 1, \dots, n$ we get:

$$\begin{aligned} h_{add}(In(i, B), s_0) &= 1 + \min\{h_{add}(\text{PRECOND}(a), s_0) \mid In(i, B) \in \text{ADD}(a)\} \\ &= 1 + h_{add}(Box(i) \wedge In(i, A), s_0) \\ &= 1 + h_{add}(Box(i), s_0) + h_{add}(In(i, A), s_0) \\ &= 1 + 0 + 0 = 1. \end{aligned}$$

Combining the above we get $h_{add}(g, s_0) = n$. Hence $h_{add} = h^*$.

Another additive heuristics example

ACTION: $Buy(x, y)$ (buy item x at location y)
PRECONDITION: $Buyable(x) \wedge Place(y) \wedge At(y) \wedge Sells(y, x)$
EFFECT: $Have(x)$

ACTION: $Go(x, y)$
PRECONDITION: $Place(x) \wedge Place(y) \wedge At(x)$
EFFECT: $At(y) \wedge \cancel{\neg At(x)}$

$s_0 = Buyable(Milk) \wedge Buyable(Bananas) \wedge Buyable(Drill) \wedge Place(Home) \wedge Place(Netto) \wedge Place(Bilka) \wedge At(Home) \wedge Sells(Netto, Milk) \wedge Sells(Netto, Bananas) \wedge Sells(Bilka, Milk) \wedge Sells(Bilka, Bananas) \wedge Sells(Bilka, Drill)$.

$g = At(Home) \wedge Have(Milk) \wedge Have(Bananas) \wedge Have(Drill)$.

$$\begin{aligned}
h_{add}(Have(M), s_0) &= 1 + \min\{h_{add}(Buyable(M) \wedge Place(N) \wedge At(N) \wedge Sells(N, M), s_0),\\
&\qquad h_{add}(Buyable(M) \wedge Place(B) \wedge At(B) \wedge Sells(B, M), s_0),\\
&\qquad h_{add}(Buyable(M) \wedge Place(H) \wedge At(H) \wedge Sells(H, M), s_0), \dots\}\\
&= 1 + \min\{1, 1, \infty, \dots\} = 2.
\end{aligned}$$

$$\begin{aligned}
h_{add}(s_0) &= h_{add}(g, s_0)\\
&= h_{add}(At(H), s_0) + h_{add}(Have(M), s_0) + h_{add}(Have(B), s_0) + h_{add}(Have(D), s_0)\\
&= 0 + 2 + 2 + 2 = 6 > h^*(s_0).
\end{aligned}$$

Why is h_{add} not optimal here? Because subgoal independence fails.

Max heuristics

The max heuristics h_{max} is exactly like the additive heuristics except the sum clause:

$$h_{add}(\bigwedge_i p_i, s) = \sum_i h_{add}(p_i, s)$$

is replaced by a maximum clause

$$h_{max}(\bigwedge_i p_i, s) = \max_i h_{max}(p_i, s).$$

(And the clause for $h_{max}(p, s)$ is exactly as the clause for $h_{add}(p, s)$ shown previously, except h_{add} is replaced by h_{max}.) This corresponds to a change in relaxation where subgoal independence is no longer assumed.

Is the new max heuristics admissible? Yes, it assumes that the cost of making a conjunction of atoms true is the cost of making the most expensive one true.

In example from previous slide: $h_{max}(s_0) = \max\{h_{max}(Have(M), s_0), h_{max}(Have(B), s_0), h_{max}(Have(D), s_0)\} = 2.$

Max heuristics

The max heuristics h_{max} is exactly like the additive heuristics except the sum clause:

$$h_{add}(\bigwedge_i p_i, s) = \sum_i h_{add}(p_i, s)$$

is replaced by a maximum clause

$$h_{max}(\bigwedge_i p_i, s) = \max_i h_{max}(p_i, s).$$

(And the clause for $h_{max}(p, s)$ is exactly as the clause for $h_{add}(p, s)$ shown previously, except h_{add} is replaced by h_{max}.) This corresponds to a change in relaxation where subgoal independence is no longer assumed.

Is the new max heuristics admissible? Yes, it assumes that the cost of making a conjunction of atoms true is the cost of making the most expensive one true.

In example from previous slide: $h_{max}(s_0) = \max\{h_{max}(Have(M), s_0), h_{max}(Have(B), s_0), h_{max}(Have(D), s_0)\} = 2.$

- **Additive heuristics**: Not admissible, but often very informative.
- **Max heuristics**: Admissible, but often not very informative.

Important about the additive and max heuristics

- The additive/max heuristics define a path-finding problem over the **atom space**: Each node is an atom rather than a state.
- It is, however, not an ordinary graph but a directed hypergraph: If p is a node (an atom) and a an action with $p \in \text{ADD}(a)$, then there is a hyperedge $(\text{PRECOND}(a), p)$ labelled by a in the graph (it is a hyperedge because $\text{PRECOND}(a)$ is a set of atoms, i.e., a set of nodes).
- The size of the hypergraph is linear in the number of ground atoms and actions, and hence the additive and max heuristics become *polynomial* in the number of ground atoms and actions.
- This is much better than calculating the entire state space which is *exponential* in the number of ground atoms and actions.