We Need to Combine Heuristic Functions!

We have covered the 4 different methods currently known:

- **Critical path heuristics:** Done. → Chapter 8
- **Delete relaxation:** Basically done. → Chapters 9 and 10
- **Abstractions:** Done. → Chapters 11–13
- **Landmarks:** Done. → Chapter 14

→ Every \( h \) yields good performance only in some domains.

Can we exploit their complementary strengths?
A Curious Observation in the 15-Puzzle

Our Agenda for This Chapter

- **Cost Partitioning**: We introduce the concept, illustrate it, and prove admissibility of the partitioned sum.
- **Domination of Previous Orthogonality Criteria**: We prove that there always exists a cost partitioning dominating our orthogonality criteria for PDBs and LMs.
- **How To Find a Cost Partitioning?**: We prove that, for PDBs and LMs and their combination, we can always find the best possible cost partitioning in polynomial time.
- **ps. Landmarks and Hitting Sets**: Departing from the fully general combination technique of cost partitioning, we have a look back at LMs and consider a technique even stronger than cost partitioning for combining this particular class of heuristic functions.

**Definition (Cost Partitioning)**. Let $\Pi$ be a planning task with actions $A$ and cost function $c$. An ensemble of functions $c_1, \ldots, c_n : A \mapsto \mathbb{R}^+_0$ is a cost partitioning for $\Pi$ if, for all $a \in A$, $\sum_{i=1}^n c_i(a) \leq c(a)$. The cost partitioning is full if, for all $a \in A$, $\sum_{i=1}^n c_i(a) = c(a)$.

**Notes and Notations**: 
- “=” (full cost partitioning) is more intuitive; but only “≤” is required for admissibility, and some practical cost partitioning methods are more naturally described that way.
- If $h$ is a heuristic for $\Pi$, then $h[c_i]$ denotes the same heuristic but computed on the modification of $\Pi$ where $c$ is replaced by $c_i$.
  → We assume that $h[c_i]$ is defined, for any $h$.
- If $h_1, \ldots, h_n$ is an ensemble of heuristic functions for $\Pi$, then the partitioned sum of $h_1, \ldots, h_n$ given $c_1, \ldots, c_n$ is $\sum_{i=1}^n h_i[c_i]$, for which we use the short-hand $\sum h[c]$. 

A Simple Example: Driving a Car

Planning task: Drive a car from left to right.

Heuristics: Two times the same heuristic. \( h_1 \): PDB for \( P_1 = \{ \text{car} \} \); \( h_2 \): PDB for \( P_2 = \{ \text{car} \} \).

\[
P_1: \quad \text{car} \\
\]

\[
P_2: \quad \text{car} \\
\]

Cost partitioning: For each action \( a \), \( c_1(a) = 0.2 \) and \( c_2(a) = 0.8 \).
\[
\rightarrow h_1[c_1(I)] + h_2[c_2(I)] =
\]

Theorem (Partitioned Sums are Admissible). Let \( II \) be a planning task, and let \( h_1, \ldots, h_n \) be heuristic functions for \( II \). If \( c_1, \ldots, c_n \) is a cost partitioning for \( II \), and if \( h_i[c_i] \) is consistent and goal-aware for all \( i \), then the partitioned sum \( \sum h[c] \) is consistent and goal-aware, and thus also admissible and safe.

\[
\rightarrow \text{Typical case: } h_i[c_i] \text{ is consistent and goal-aware because } h_i \in H_i \text{ where } H_i \text{ is a family of heuristics (a class of heuristics } h \text{ computed using the same framework, e.g. PDB heuristic) that are consistent and goal-aware.}
\]

Simple Again? Driving Two Cars

Planning task: Drive both cars from left to right, using actions \( \text{drive(carA, X, Y)} \) and \( \text{drive(carB, X, Y)} \) (unit costs).

\[
carA: \quad \text{car} \\
\]

\[
carB: \quad \text{car} \\
\]

Question!

What is the value of \( h^{\text{max}}(I) \) for this task?

(A): 3  (B): 6
Planning task: Drive both cars from left to right, using actions \( \text{drive}(\text{car}A, X, Y) \) and \( \text{drive}(\text{car}B, X, Y) \) (unit costs).

- carA:  
  -  
  -  
  -  
  -  
- carB:  
  -  
  -  
  -  

We can admissibly combine arbitrary heuristic functions.

→ But for the particular methods we have, is that any better than the admissible combinations we defined earlier?

Yes! (provided we manage to find the right cost partitionings)

- Given a collection \( L_1, \ldots, L_n \) of action sets, there always exists a cost partitioning that dominates the canonical (LM) heuristic, i.e., the best sum of orthogonal \( h_{LM}^{L_i} \).
- Given a pattern collection \( P_1, \ldots, P_n \), there always exists a cost partitioning that dominates the canonical (PDB) heuristic, i.e., the best sum of orthogonal \( h_{P_i} \).
- In both settings, there are cases where the domination is strict.
Dominating Orthogonal PDBs

Reminder:

An action $a$ affects a projection $\pi_P$ if there exists a variable $v \in P$ on which $\text{eff}_a$ is defined. Patterns $P_1, \ldots, P_n$ are orthogonal if every action affects at most one $P_i$. Then, $\sum_{i=1}^{n} h_{P_i}$ is admissible.

Theorem (Cost Partitionings Can Dominate the Sum of Orthogonal PDBs). Let II be a planning task, and let $\{P_1, \ldots, P_n\}$ be an orthogonal pattern collection. For each $i$ and $a \in A$, define $c_i(a) := c(a)$ if $a$ affects $\alpha_i$, and $c_i(a) := 0$ otherwise. Then $c_1, \ldots, c_n$ is a cost partitioning, and for all states $s$ we have $\sum_{i=1}^{n} h_{P_i}(s) = \sum h_{\{c_i\}}(s)$.

→ Orthogonality for PDBs is subsumed by “0/1” cost partitionings, putting the entire cost of each action into the PDB it affects.

(→ Yes, this works for arbitrary abstractions, not just for PDBs.)

Proof.

Corollary (Cost Partitionings Can Dominate the Canonical PDB Heuristic). Let II be a planning task, let $C$ be a pattern collection, and let $s$ be a state. Then there exists a cost partitioning for II so that $h^C(s) \leq \sum h_{\{c_i\}}(s)$.

Proof. “$\leq$”: Apply theorem to the additive $\{P_1, \ldots, P_k\} \subseteq C$ yielding the maximum in $s$. “$<$”: See next slide.

→ State-dependence: $h^C$ selects the maximum additive $\{P_1, \ldots, P_k\} \subseteq C$ depending on the state. Hence we have to select the cost partitioning depending on the state. That is, we can’t in general select a cost partitioning once and dominate $h^C$ on all states. Example see next slide.
Questionnaire

**Planning task:** Drive both cars from left to right.

- \( P_1 = \{\text{car} A\} \)
- \( P_2 = \{\text{car} B\} \)

**Question:** Which cost partitioning corresponds to the canonical PDB heuristic here?

**Questionnaire**

**Given:** A collection \( h_1, \ldots, h_n \) of admissible heuristics, and a state \( s \).

**Wanted:** A cost partitioning \( c_1, \ldots, c_n \).

**Number of candidates:** Infinite.

→ Do all of these yield a good overall lower bound on \( h^*(s) \)?

→ Many (most) cost partitionings are bad. Our challenge is to automatically find good ones.

→ The challenge is particularly vexing because ideally we want to do this for every search state \( s \)! (In particular, if we wish to dominate the canonical heuristics, cf. slides 21 and 18.)

**Definition (Optimal Cost Partitioning).** Let \( \Pi \) be a planning task, let \( h_1, \ldots, h_n \) be admissible heuristic functions for \( \Pi \), and let \( s \) be a state. An **optimal cost partitioning** for \( s \) and \( h_1, \ldots, h_n \) is any cost partitioning \( c_1, \ldots, c_n \) for which \( \sum h[c](s) \) is maximal.

→ Optimal cost partitionings distribute costs in a way that yields the best possible lower bound, for a given state.

**Question:** Does this definition sound completely impractical?

(A): Yes  (B): No
Optimal Cost Partitioning for Landmarks

**Theorem (Polynomial-Time Optimal Cost Partitioning for Landmarks).** Let \( \Pi \) be a planning task, let \( s \) be a state, and let \( L_1, \ldots, L_n \) be disjunctive action landmarks for \( s \). Then an optimal cost partitioning for \( s \) and \( h^{L_1}, \ldots, h^{L_n} \) can be computed in time polynomial in \( |\Pi| \) and \( n \).

**Proof Sketch.** The problem of finding an optimal cost partitioning \( c_1, \ldots, c_n \) can be formulated as a Linear Programming (LP) problem. We use LP variables \( c_{i,a} \) for encoding the partitioned costs, and variables \( h_{L_i} \) encoding the weight the final heuristic will count for the landmark \( L_i \). Simple constraints ensure that \( c_{i,a} \) is indeed a cost partitioning, and that the weights \( h_{L_i} \) are not larger than allowed. Maximizing \( \sum_{i=1}^{n} h_{L_i} \) results in an optimal cost partitioning.

→ Selection of the cheapest action from a landmark \( L_i \) can be encoded into LP, giving a weight to each \( L_i \). An optimal cost partitioning corresponds to an LP solution maximizing the summed-up weights.

→ Note: We assume here that the \( L_i \) are LMs for \( s \). Corresponds to standard methods determining a set of LMs for each search state (cf. Chapter 14).

Alvaro Torralba, Cosmina Croitoru  AI Planning  Chapter 15: Combining Heuristic Functions  29/49

Optimal Cost Partitioning for PDBs

**Theorem (Polynomial-Time Optimal Cost Partitioning for PDBs).** Let \( \Pi \) be a planning task, let \( s \) be a state, and let \( \{P_1, \ldots, P_n\} \) be a pattern collection. Then an optimal cost partitioning for \( s \) and \( h^{P_1}, \ldots, h^{P_n} \) can be computed in time polynomial in \( |\Pi| \) and \( |\Theta^{P_1}|, \ldots, |\Theta^{P_n}| \).

**Proof Sketch.** LP formulation: Constraints \( \sum_{i=1}^{n} c_{i,a} \leq c(a) \) ensure that we get a cost partitioning. For each \( i \), constraints \( c_{i,a} = 0 \) and \( c_{i,t'} \leq c_{i,t} + c_{i,a} \) for each transition \( t \xrightarrow{a} t' \) in \( \Theta^{P_i} \) ensure that \( c_{i,t} \) for any state \( t \) in \( \Theta^{P_i} \) is at most the abstract cost to reach \( t \) from \( s \) (using the partitioned costs \( c_{i,a} \)). Constraints \( h_{P_i} \leq c_{i,t} \) for all abstract goal states \( t \) in \( \Theta^{P_i} \) ensure that the weight \( h_{P_i} \) counted for each \( P_i \) is at most the real abstract remaining cost of \( s \).

Maximizing \( \sum_{i=1}^{n} h_{P_i} \) results in an optimal cost partitioning.

→ Cheapest paths in abstract state spaces can be encoded into LP, giving a weight to each PDB. An optimal cost partitioning corresponds to an LP solution maximizing the summed-up weights.

(→ Yes, this works for arbitrary abstractions, not just for PDBs.)
**Theorem (Polynomial-Time Optimal Cost Partitioning).** Let $\Pi$ be a planning task, let $s$ be a state, and let $h_1, \ldots, h_n$ be heuristic functions for $\Pi$ such that each $h_i$ either is given by $h_i = h_i^{LM}$ for a disjunctive action landmark for $s$, or is given by $h_i = h_i^{P}$ for a pattern $P_i$ with abstract state space $\Theta_{P_i}$. Then an optimal cost partitioning for $s$ and $h_1, \ldots, h_n$ can be computed in time polynomial in $|\Pi|$ and the size of the representation of $h_1, \ldots, h_n$.

**Proof Sketch.** Simply put all the LP variables and constraints described previously into a single formulation.

→ Selection of the cheapest action from a landmark $L_i$ can be encoded into LP, giving a weight to each $L_i$. Cheapest paths in abstract state spaces can be encoded into LP, giving a weight to each PDB. An optimal cost partitioning corresponds to an LP solution maximizing the summed-up weights. (→ Yes, this works for arbitrary abstractions, not just for PDBs.)

**Optimal Cost Partitionings for Landmarks and PDBs**
Where Cost Partitioning Fails

**Theorem.** Let \( s \) be a state, and let \( L_1, \ldots, L_n \) be the collection of all delete relaxation disjunctive action landmarks for \( s \). Let \( H \) be a minimum-cost hitting set. Then \( \sum_{a \in H} c(a) = h^+(s) \).

**Proof.** “\( \leq \)”: Every relaxed plan must hit every \( L_i \). For “\( \geq \)”, we prove that any hitting set \( H \) contains a relaxed plan. With \( RH := \{ p | p \text{ can be reached in delete relaxation using only } H \} \), assume to the contrary that \( G \not\subseteq RH \). Choose 1 fact from the goal and each action precondition, using a fact outside \( RH \) where possible. Consider the graph over facts with arcs \( (p, a, q) \) where \( p \in \text{pre}_a \) and \( q \in \text{eff}_a \), and consider the cut \( L \) defined by \( RH, \overline{RH} \):

\[
\begin{align*}
  & s \\
  & a(1) \\
  & a(2) \\
  & L \\
  & RH \\
  & G \\
\end{align*}
\]

\( L \) is a LM for \( s \): We cannot reach the goal without using one of these actions. However, consider any \( a \in H \). Case (1): If \( \text{pre}_a \subseteq RH \), then \( \text{add}_a \subseteq RH \) because \( a \in H \). So \( a \notin L \). Case (2): If \( \text{pre}_a \not\subseteq RH \), then we selected \( p \in \text{pre}_a \setminus RH \). So, again, \( a \notin L \). Altogether, \( H \) does not hit \( L \), in contradiction.

### So What?

- Hitting sets over LMs were first proposed by [Bonet and Helmert (2010)].
- Hitting sets over LMs dominate the optimal cost partitioning. This is because, for any action \( a \), the total weight (after cost partitioning) of all LMs \( a \) participates in is bounded by \( c(a) \). So if we hit all LMs then we get an upper bound on the cost-partitioning heuristic.
- There are constructive methods to find “complete” sets of landmarks, i.e., methods which guarantee that the minimum-cost hitting set will deliver \( h^+ \). This is nowadays the state-of-the-art method to compute \( h^+ \) [Haslum et al. (2012)].
- A similar result does not hold for \( h^* \) even if we somehow found all (non-delete relaxed) disjunctive action LMs. This is because, in the original planning task, we may have to apply the same action more than once.
- In practice, hitting sets over LMs tend to be computationally too expensive (for every state, apart from finding all the LMs we have to solve the NP-hard minimum-cost hitting set problem . . . ).

### Summary

- A cost partitioning distributes the cost of each action across \( n \) otherwise identical planning tasks. This can be used to admissibly sum up any ensemble of admissible heuristic functions.
- For every state and ensemble of PDB heuristics, there exists a cost partitioning that dominates the canonical PDB heuristic; the domination can be strict.
- The same is true of the canonical LM heuristic.
- Optimal cost partitionings distribute action costs such that the lower bound for a given state is maximal.
- For PDBs and LMs, and for their combination, optimal cost partitionings can be computed in polynomial time by Linear Programming.
- In practice, computing optimal cost partitionings for every search state typically is too costly, and we need to approximate.
The admissible combination of lower bounds has a long history. Famous instances pertain to additive PDBs in Game playing [Felner et al. (2004)].

In planning, this story also started with additive PDBs [Edelkamp (2001); Haslum et al. (2007)], then was extended to $h^n$ among others [Haslum et al. (2005)]. The intuition always was to design the heuristics in a way making them independent.

When I was in some project meeting somewhere in about 2005, someone from outside the area said “But what if we count each move only half in each of the heuristics?” The remark was received with confusion, then forgotten about.

Then Michael & Carmel [Katz and Domshlak (2008)] suddenly came along and told us we’d been looking at 0/1 cost partitionings all the time, and how to find optimal general ones efficiently using LP.

Since then, various works towards making this practical, cf. slide 35.

Cost partitioning is not specific to planning, can be applied anywhere!

### Reading

- **Optimal Additive Composition of Abstraction-Based Admissible Heuristics** [Katz and Domshlak (2008)].
  
  **Available at:**
  

  **Content:** Original paper proposing cost partitioning, and showing that, for certain classes of heuristics, optimal cost partitionings can be computed in polynomial time using Linear Programming. Specifically, the paper established this for abstractions as handled in this course, as well as for implicit abstractions represented through planning task fragments identified based on the causal graph.

- **Cost-Optimal Planning with Landmarks** [Karpas and Domshlak (2009)].
  
  **Available at:**
  
  [http://iee3.technion.ac.il/~dcarmel/Papers/Sources/ijcai09a.pdf](http://iee3.technion.ac.il/~dcarmel/Papers/Sources/ijcai09a.pdf)

  **Content:** The “alarm clock” waking LMs up to the modern age of cost-optimal planning (cf. Chapter 14). Introduces cost partitioning for elementary landmarks heuristics, and the computation of optimal cost partitionings for such heuristics using Linear Programming. Introduces uniform cost partitioning, which is used in the experiments due to being more runtime-effective.

- **Diverse and Additive Cartesian Abstraction Heuristics** [Seipp and Helmert (2014)].
  
  **Available at:**
  

  **Content:** Introduces the current state of the art technique for cost partitioning with abstraction heuristics, saturated cost partitioning, which partitions costs according to what is actually needed to preserve the abstraction heuristic.
References I


References II


References III


