PUI: Notes on Classical Planning

Daniel Fišer DANFIS@DANFIS.CZ
Department of Computer Science, Faculty of Electrical Engineering
Czech Technical University in Prague

1. Representations

Definition 1. A STRIPS **planning task** Π is specified by a tuple $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$, where $\mathcal{F} = \{f_1, ..., f_n\}$ is a set of facts, $\mathcal{O} = \{o_1, ..., o_m\}$ is a set of operators, and c is a cost function mapping each operator to a non-negative real number. A **state** $s \subseteq \mathcal{F}$ is a set of facts, $s_{init} \subseteq \mathcal{F}$ is an **initial state** and $s_{goal} \subseteq \mathcal{F}$ is a **goal** specification. An **operator** o is a triple $o = \langle \mathrm{pre}(o), \mathrm{add}(o), \mathrm{del}(o) \rangle$, where $\mathrm{pre}(o) \subseteq \mathcal{F}$ is a set of preconditions, and $\mathrm{add}(o) \subseteq \mathcal{F}$ and $\mathrm{del}(o) \subseteq \mathcal{F}$ are sets of add and delete effects, respectively. All operators are well-formed, i.e., $\mathrm{add}(o) \cap \mathrm{del}(o) = \emptyset$ and $\mathrm{pre}(o) \cap \mathrm{add}(o) = \emptyset$. An operator o is **applicable** in a state s if $\mathrm{pre}(o) \subseteq s$. The **resulting state** of applying an applicable operator o in a state s is the state $o[s] = (s \setminus \mathrm{del}(o)) \cup \mathrm{add}(o)$. A state s is a **goal state** iff $s_{goal} \subseteq s$.

A **sequence of operators** $\pi = \langle o_1, ..., o_n \rangle$ is applicable in a state s_0 if there are states $s_1, ..., s_n$ such that o_i is applicable in s_{i-1} and $s_i = o_i[s_{i-1}]$ for $1 \leq i \leq n$. The resulting state of this application is $\pi[s_0] = s_n$ and the cost of the plan is $\mathrm{c}(\pi) = \sum_{o \in \pi} \mathrm{c}(o)$. A sequence of operators π is called a **plan** iff $s_{goal} \subseteq \pi[s_{init}]$, and an **optimal plan** is a plan with the minimal cost over all plans.

Definition 2. An FDR planning task P is specified by a tuple $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$, where $\mathcal{V}$ is a finite set of **variables**. Each variable $V \in \mathcal{V}$ has a finite domain D_V. A (partial) **state** s is a (partial) variable assignement over $\mathcal{V}$. We write $\mathrm{vars}(s)$ for the set of variables defined in s and $s[V]$ for the value of V in s. The notation $s[V] = \perp$ means that $V \notin \mathrm{vars}(s)$. A partial state s is **consistent** with a partial state s' if $s[V] = s'[V]$ for all $V \in \mathrm{vars}(s')$. We say that **atom** $V = v$ is true in a (partial) state s iff $s[V] = v$. By c we denote a cost function mapping each operator to a non-negative real number. An **operator** $o \in \mathcal{O}$ is a pair $o = \langle \mathrm{pre}(o), \mathrm{eff}(o) \rangle$, where precondition $\mathrm{pre}(o)$ and effect $\mathrm{eff}(o)$ are partial assignements over $\mathcal{V}$. We require that $V = v$ cannot be both a precondition and an effect. The (complete) state s_{init} is the **initial state** of the task and the partial state s_{goal} describes its **goal**.

An operator o is **applicable** in a state s if s is consistent with $\mathrm{pre}(o)$. The **resulting state** of applying an applicabe operator o in the state s is the state $\mathrm{res}(o, s)$ with

$$\mathrm{res}(o, s) = \begin{cases} \mathrm{eff}(o)[V] & \text{if } V \in \mathrm{vars}(\mathrm{eff}(o)), \\ s[V] & \text{otherwise.} \end{cases}$$

A **sequence of operators** $\pi = \langle o_1, ..., o_n \rangle$ is applicable in a state s_0 if there are states $s_1, ..., s_n$ such that o_i is applicable in s_{i-1} and $s_i = \mathrm{res}(o_i, s_{i-1})$ for $1 \leq i \leq n$. The resulting state of this application is $\mathrm{res}(\pi, s_0) = s_n$ and the cost of the plan is $\mathrm{c}(\pi) = \sum_{o \in \pi} \mathrm{c}(o)$.

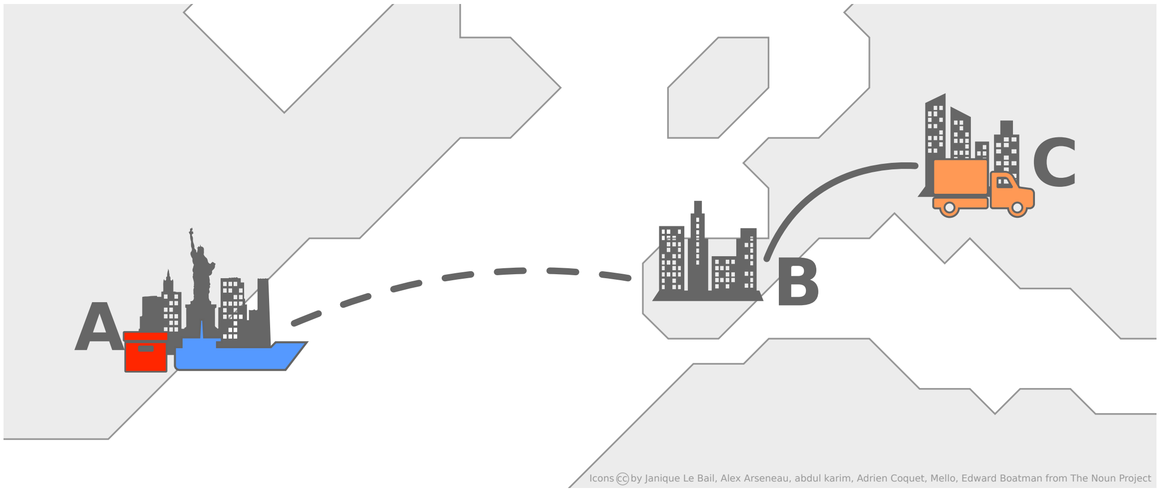

Figure 1: Example problem.

A sequence of operators π is called a **plan** iff $\mathrm{res}(\pi, s_{init})$ is consistent with s_{goal}, and an **optimal plan** is a plan with the minimal cost over all plans.

Exercises
Ex. 1.1 — Model the problem from Fig. 1 in STRIPS.

Ex. 1.2 — Model the problem from Fig. 1 in FDR.

2. $\mathrm{h}^{\max}$ Heuristic

Definition 3. Given a STRIPS planning task $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$, $\Pi^+ = \langle \mathcal{F}, \mathcal{O}^+, s_{init}, s_{goal}, \mathrm{c} \rangle$ denotes a **relaxed** STRIPS planning task, where $\mathcal{O}^+ = \{o_i^+ = \langle \mathrm{pre}(o_i), \mathrm{add}(o_i), \emptyset \rangle \mid o_i \in \mathcal{O}\}$.

Definition 4. Let $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$ denote a STRIPS planning task. The heuristic function $\mathrm{h}^{\mathrm{add}}(s)$ gives an estimate of the distance from s to a node that satisfies the goal s_{goal} as $\mathrm{h}^{\mathrm{add}}(s) = \Sigma_{f \in s_{goal}} \Delta_0(s, f)$, where:

$$\Delta_0(s, o) = \Sigma_{f \in \mathrm{pre}(o)} \Delta_0(s, f), \quad \forall o \in \mathcal{O},$$

and

$$\Delta_0(s, f) = \begin{cases} 0 & \text{if } f \in s, \\ \infty & \text{if } \forall o \in \mathcal{O} : f \notin \mathrm{add}(o), \\ \min\{\mathrm{c}(o) + \Delta_0(s, o) \mid o \in \mathcal{O}, f \in \mathrm{add}(o)\} & \text{otherwise.} \end{cases}$$

Definition 5. Let $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$ denote a STRIPS planning task. The heuristic function $\mathrm{h}^{\max}(s)$ gives an estimate of the distance from s to a node that satisfies the goal s_{goal} as $\mathrm{h}^{\max}(s) = \max_{f \in s_{goal}} \Delta_1(s, f)$, where:

$$\Delta_1(s, o) = \max_{f \in \mathrm{pre}(o)} \Delta_1(s, f), \quad \forall o \in \mathcal{O},$$

and

$$\Delta_1(s,f) = \begin{cases} 0 & \text{if } f \in s, \\ \infty & \text{if } \forall o \in \mathcal{O} : f \notin \text{add}(o), \\ \min\{\text{c}(o) + \Delta_1(s,o) \mid o \in \mathcal{O}, f \in \text{add}(o)\} & \text{otherwise.} \end{cases}$$

Algorithm 1: Algorithm for computing $\text{h}^{\text{max}}(s)$.

Input: $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \text{c} \rangle$, state s
Output: $\text{h}^{\text{max}}(s)$

for each $f \in s$ **do** $\Delta_1(s,f) \leftarrow 0$;
for each $f \in \mathcal{F} \setminus s$ **do** $\Delta_1(s,f) \leftarrow \infty$;
for each $o \in \mathcal{O}$ **do** $U(o) \leftarrow |\text{pre}(o)|$;
$C \leftarrow \emptyset$;
while $s_{goal} \not\subseteq C$ **do**
    $c \leftarrow \arg\min_{f \in \mathcal{F} \setminus C} \Delta_1(s,f)$;
    $C \leftarrow C \cup \{c\}$;
    for each $o \in \mathcal{O}, c \in \mathit{pre}(o)$ **do**
        $U(o) \leftarrow U(o) - 1$;
        if $U(o) = 0$ **then**
            for each $f \in \mathit{add}(o)$ **do**
                $\Delta_1(s,f) \leftarrow \min\{\Delta_1(s,f), \text{c}(o) + \Delta_1(s,c)\}$;
            end
        end
    end
end
$\text{h}^{\text{max}}(s) = \max_{f \in s_{goal}} \Delta_1(s,f)$;

Exercises

Ex. 2.1 — Modify Algorithm 1 to compute h^{add} instead of h^{max}.

Ex. 2.2 — Compute $\text{h}^{\text{max}}(s_{init})$, $\text{h}^{\text{add}}(s_{init})$, $\text{h}^{+}(s_{init})$, and $\text{h}^{\star}(s_{init})$ for the following problem $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \text{c} \rangle$:

$\mathcal{F} = \{a, b, c, d, e, f, g\}$

$\mathcal{O} =$

	pre	add	del	c
o_1	$\{a\}$	$\{c,d\}$	$\{a\}$	1
o_2	$\{a,b\}$	$\{e\}$	$\emptyset$	1
o_3	$\{b,e\}$	$\{d,f\}$	$\{a,e\}$	1
o_4	$\{b\}$	$\{a\}$	$\emptyset$	1
o_5	$\{d,e\}$	$\{g\}$	$\{e\}$	1

$s_{init} = \{a, b\}, s_{goal} = \{f, g\}$

3. LM-Cut Heuristic

Definition 6. A **disjunctive operator landmark** $L \subseteq \mathcal{O}$ is a set of operators such that every plan contains at least one operator from L.

Definition 7. Let $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$ denote a planning task, let Δ_1 denote the function from Definition 5 for Π, and let $\mathrm{supp}(o) = \arg\max_{f \in \mathrm{pre}(o)} \Delta_1(f)$ denote a function mapping each operator to its **supporter**.

A **justification graph** $G = (N, E)$ is a directed labeled multigraph with a set of nodes $N = \{n_f \mid f \in \mathcal{F}\}$ and a set of edges $E = \{(n_s, n_t, o) \mid o \in \mathcal{O}, s = \mathrm{supp}(o), t \in \mathrm{add}(o)\}$, where the triple (a, b, l) denotes an edge from a to b with the label l.

An **s-t-cut** $\mathcal{C}(G, s, t) = (N^0, N^\star \cup N^b)$ is a partitioning of nodes from the justification graph $G = (N, E)$ such that $N^\star$ contains all nodes from which t can be reached with a zero-cost path, N^0 contains all nodes reachable from s without passing through any node from $N^\star$, and $N^b = N \setminus (N^0 \cup N^\star)$.

Algorithm 2: Algorithm for computing $\mathrm{h}^{\text{lm-cut}}(s)$.

Input: $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$, state s
Output: $\mathrm{h}^{\text{lm-cut}}(s)$
$\mathrm{h}^{\text{lm-cut}}(s) \leftarrow 0$;
$\Pi_1 = \langle \mathcal{F}' = \mathcal{F} \cup \{I, G\}, \mathcal{O}' = \mathcal{O} \cup \{o_{init}, o_{goal}\}, s'_{init} = \{I\}, s'_{goal} = \{G\}, \mathrm{c}_1 \rangle$, where $\mathrm{pre}(o_{init}) = \{I\}$, $\mathrm{add}(o_{init}) = s_{init}$, $\mathrm{del}(o_{init}) = \emptyset$, $\mathrm{pre}(o_{goal}) = s_{goal}$, $\mathrm{add}(o_{goal}) = \{G\}$, $\mathrm{del}(o_{goal}) = \emptyset$, $\mathrm{c}_1(o_{init}) = 0$, $\mathrm{c}_1(o_{goal}) = 0$, and $\mathrm{c}_1(o) = \mathrm{c}(o)$ for all $o \in \mathcal{O}$;
$i \leftarrow 1$;
while $\mathrm{h}^{\max}(\Pi_i, s'_{init}) \neq 0$ **do**
    Construct a justification graph G_i from Π_i;
    Construct an s-t-cut $\mathcal{C}_i(G_i, n_I, n_G) = (N_i^0, N_i^\star \cup N_i^b)$;
    Create a landmark L_i as a set of labels of edges that cross the cut $\mathcal{C}_i$, i.e., they lead from N_i^0 to $N_i^\star$;
    $m_i \leftarrow \min_{o \in L_i} \mathrm{c}_i(o)$;
    $\mathrm{h}^{\text{lm-cut}}(s) \leftarrow \mathrm{h}^{\text{lm-cut}}(s) + m_i$;
    Set $\Pi_{i+1} = \langle \mathcal{F}', \mathcal{O}', s'_{init}, s'_{goal}, \mathrm{c}_{i+1} \rangle$, where $\mathrm{c}_{i+1}(o) = \mathrm{c}_i(o) - m_i$ if $o \in L_i$, and $\mathrm{c}_{i+1}(o) = \mathrm{c}_i(o)$ otherwise;
    $i \leftarrow i + 1$;
end

Exercises

Ex. 3.1 — Modify Algorithm 1 to compute $\mathrm{h}^{\max}$ and to find supporters from Definition 7 at the same time.

Ex. 3.2 — Compute $\mathrm{h}^{\text{lm-cut}}(s_{init})$ for the following problem $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$:
$\mathcal{F} = \{s, t, q_1, q_2, q_3\}$

$\mathcal{O} =$

	pre	add	del	c
o_1	$\{s\}$	$\{q_1, q_2\}$	$\emptyset$	1
o_2	$\{s\}$	$\{q_1, q_3\}$	$\emptyset$	1
o_3	$\{s\}$	$\{q_2, q_3\}$	$\emptyset$	1
fin	$\{q_1, q_2, q_3\}$	$\{t\}$	$\emptyset$	0

$s_{init} = \{s\}, s_{goal} = \{t\}$

Ex. 3.3 — Compute $\mathrm{h}^{\max}(s_{init})$, $\mathrm{h}^{\text{lm-cut}}(s_{init})$, $\mathrm{h}^{+}(s_{init})$, and $\mathrm{h}^{\star}(s_{init})$ for the following problem $\Pi = \langle \mathcal{F}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c}\rangle$:
$\mathcal{F} = \{a, b, c, d, e, i, g\}$

$\mathcal{O} =$

	pre	add	del	c
o_1	$\{i\}$	$\{a, b\}$	$\emptyset$	2
o_2	$\{i\}$	$\{b, c\}$	$\emptyset$	3
o_3	$\{a, c\}$	$\{d\}$	$\{c\}$	1
o_4	$\{b, d\}$	$\{e\}$	$\{b\}$	3
o_5	$\{a, c, e\}$	$\{g\}$	$\{c, d\}$	1
o_6	$\{a\}$	$\{e\}$	$\{a, c\}$	5

$s_{init} = \{i\}, s_{goal} = \{g\}$

Ex. 3.4 — Decide dominance for the following cases: $\mathrm{h}^{\max} \succcurlyeq \mathrm{h}^{\mathrm{add}}$, $\mathrm{h}^{\max} \succcurlyeq \mathrm{h}^{\text{lm-cut}}$, $\mathrm{h}^{\max} \succcurlyeq \mathrm{h}^{+}$, $\mathrm{h}^{\text{lm-cut}} \preccurlyeq \mathrm{h}^{+}$, $\mathrm{h}^{\text{lm-cut}} \succcurlyeq \mathrm{h}^{\max}$.

4. Merge And Shrink Heuristic

Definition 8. A **transition system** is a tuple $\mathcal{T} = \langle S, L, T, I, G\rangle$, where S is a finite set of **states**, L is a finite set of **labels**, each label has **cost** $\mathrm{c}(l) \in \mathbb{R}^+_0$, $T \subseteq S \times L \times S$ is a **transition relation**, $I \subseteq S$ is a set of initial states, and $G \subseteq S$ is a set of goal states.

Definition 9. Given an FDR planning task $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c}\rangle$, $\mathcal{T}(P) = \langle S, L, T, I, G\rangle$ denote a **transition system of** P, where S is a set of states over $\mathcal{V}$, $L = \mathcal{O}$, $T = \{(s, o, t) \mid \mathrm{res}(o, s) = t\}$, $I = \{s_{init}\}$, and $G = \{s \mid s \in S, s \text{ is consistent with } s_{goal}\}$.

Definition 10. Let $\mathcal{T}^1 = \langle S^1, L, T^1, I^1, G^1\rangle$ and $\mathcal{T}^2 = \langle S^2, L, T^2, I^2, G^2\rangle$ denote two transition systems with the same set of labels, and let $\alpha : S^1 \mapsto S^2$. We say that S^2 is an **abstraction of** S^1 with **abstraction function** α if for every $s \in I^1$ it holds that $\alpha(s) \in I^2$ and for every $s \in G^1$ it holds that $\alpha(s) \in G^2$ and for every $(s, l, t) \in T^1$ it holds that $(\alpha(s), l, \alpha(t)) \in T^2$.

Definition 11. Let P denote an FDR planning task, let $\mathcal{A}$ denote an abstraction of a transition system $\mathcal{T}(P) = \langle S, L, T, I, G\rangle$ with the abstraction function α. The **abstraction heuristic** induced by $\mathcal{A}$ and α is the function $h^{\mathcal{A},\alpha}(s) = \mathrm{h}^{\star}(\mathcal{A}, \alpha(s))$ for all $s \in S$.

Definition 12. Given two transition systems $\mathcal{T}^1 = \langle S^1, L, T^1, I^1, G^1\rangle$ and $\mathcal{T}^2 = \langle S^2, L, T^2, I^2, G^2\rangle$ with the same set of labels, the **synchronized product** $\mathcal{T}^1 \otimes \mathcal{T}^2 = \mathcal{T}$ is a transition system $\mathcal{T} = \langle S, L, T, I, G\rangle$, where $S = S^1 \times S^2$, $T = \{((s_1, s_2), l, (t_1, t_2)) \mid (s_1, l, s_2) \in T^1, (s_2, l, t_2) \in T^2\}$, $I = I^1 \times I^2$, and $G = G^1 \times G^2$.

Algorithm 3: Algorithm for computing merge-and-shrink.

Input: $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$
Output: Abstraction $\mathcal{M}$

$\mathcal{A} \leftarrow$ Set of (atomic) abstractions $(\alpha_i, \mathcal{T}_i)$ of $\mathcal{T}(P)$;
while $|\mathcal{A}| > 1$ **do**
  $A_1 = (\alpha_1, \mathcal{T}_1), A_2 = (\alpha_2, \mathcal{T}_2) \leftarrow$ Select two abstractions from $\mathcal{A}$;
  Shrink A_1 and/or A_2 until they are "small enough";
  $\mathcal{A} \leftarrow (\mathcal{A} \setminus \{A_1, A_2\}) \cup (A_1 \otimes A_2)$ `// Merge`
end
$\mathcal{M} \leftarrow$ The only element of $\mathcal{A}$;

Exercises

Ex. 4.1 — Compute the synchronized product of $\mathcal{T}^1 = \langle S^1, L, T^1, I^1, G^1 \rangle$ and $\mathcal{T}^2 = \langle S^2, L, T^2, I^2, G^2 \rangle$, where $L = \{a, b, c, d, e\}$, $S^1 = \{A, B, C, D\}$, $T^1 = \{(A, a, B), (B, b, C), (C, c, A), (A, d, A), (A, e, D)\}$, $I^1 = \{A, B\}$, $G^1 = \{A, C\}$, $S^2 = \{X, Y, Z\}$, $T^2 = \{(X, a, Y), (X, a, Z), (Y, b, Z), (Z, c, Y), (Z, d, Y), (Z, e, Z)\}$, $I^2 = \{X\}$, and $G^2 = \{X\}$.

Ex. 4.2 — Study merge and shrink strategies proposed by Helmert, Haslum, and Hoffmann (2007) and compute $\mathrm{h}^{\mathrm{m\&s}}(s_{init})$ for the problem in Fig. 1 (Ex. 1.2).

5. LP-Based Heuristics

Definition 13. Let $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$ denote an FDR planning task. The **domain transition graph** for a variable $V \in \mathcal{V}$ is a tuple $\mathcal{A}_V = (N_V, L_V, T_V)$, where $N_V = \{n_v \mid v \in D_V\} \cup \{n_\perp\}$ is a set of nodes, $L_V = \{o \mid o \in \mathcal{O}, V \in \mathrm{vars}(\mathrm{pre}(o)) \cup \mathrm{vars}(\mathrm{eff}(o))\}$ is a set of labels, and $T_V \subseteq N_V \times L_V \times N_V$ is a set of transitions $T_V = \{(n_u, o, n_v) \mid o \in L_V, V \in \mathrm{vars}(\mathrm{eff}(o)), \mathrm{pre}(o)[V] = u, \mathrm{eff}(o)[V] = v\} \cup \{(n_v, o, n_v) \mid o \in L_V, V \notin \mathrm{vars}(\mathrm{eff}(o)), \mathrm{pre}(o)[V] = v\}$.

Definition 14. Let $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$ denote an FDR planning task, $\mathcal{A}_V = (N_V, L_V, T_V)$ a domain transition graph for each variable $V \in \mathcal{V}$, and s a state reachable from s_{init}. Given the following linear program with real-valued variables x_o for each operator $o \in \mathcal{O}$:

$$\begin{aligned} &\text{minimize} && \sum_{o \in \mathcal{O}} c(o) x_o \\ &\text{subject to} && LB_{V,v} \leq \sum_{(v', o, v) \in T_V} x_o - \sum_{(v, o, v') \in T_V} x_o \quad \forall V \in \mathcal{V}, \forall v \in D_V, \end{aligned}$$

where

$$LB_{V,v} = \begin{cases} 0 & \text{if } V \in \mathrm{vars}(s_{goal}) \text{ and } s_{goal}[V] = v \text{ and } s[V] = v, \\ 1 & \text{if } V \in \mathrm{vars}(s_{goal}) \text{ and } s_{goal}[V] = v \text{ and } s[V] \neq v, \\ -1 & \text{if } (V \notin \mathrm{vars}(s_{goal}) \text{ or } s_{goal}[V] \neq v) \text{ and } s[V] = v, \\ 0 & \text{if } (V \notin \mathrm{vars}(s_{goal}) \text{ or } s_{goal}[V] \neq v) \text{ and } s[V] \neq v, \end{cases}$$

then the value of the **flow heuristic** $\mathrm{h}^{\mathrm{flow}}(s)$ for the state s is

$$\mathrm{h}^{\mathrm{flow}}(s) = \begin{cases} \left\lceil \sum_{o \in \mathcal{O}} c(o) x_o \right\rceil & \text{if the solution is feasible,} \\ \infty & \text{if the solution is not feasible.} \end{cases}$$

(Bonet, 2013; Bonet & van den Briel, 2014)

Definition 15. Let $P = \langle \mathcal{V}, \mathcal{O}, s_{init}, s_{goal}, \mathrm{c} \rangle$ denote an FDR planning task and s a state reachable from s_{init}. Given the following linear program with real-valued variables $P_{V,v}$ for each variable $V \in \mathcal{V}$ and each value $v \in D_V$, and real-valued variables M_V for each variable $V \in \mathcal{V}$:

$$\begin{aligned} \text{maximize} \quad & \sum_{V \in \mathcal{V}} P_{V,s[V]} \\ \text{subject to} \quad & P_{V,v} \leq M_V && \forall V \in \mathcal{V}, \forall v \in D_V \\ & \sum_{V \in \mathcal{V}} maxpot(V, s_{goal}) \leq 0 \\ & \sum_{V \in \mathrm{vars}(\mathrm{eff}(o))} (maxpot(V, \mathrm{pre}(o)) - P_{V,\mathrm{eff}(o)[V]}) \leq \mathrm{cost}(o) && \forall o \in \mathcal{O}, \end{aligned}$$

where

$$maxpot(V, p) = \begin{cases} P_{V,p[V]} & \text{if } V \in \mathrm{vars}(p), \\ M_V & \text{otherwise .} \end{cases}$$

then the value of the **potential heuristic** $\mathrm{h}^{\mathrm{pot}}(s)$ for the state s is

$$\mathrm{h}^{\mathrm{pot}}(s) = \begin{cases} \left\lceil \sum_{V \in \mathcal{V}} P_{V,s[V]} \right\rceil & \text{if the solution is feasible,} \\ \infty & \text{if the solution is not feasible.} \end{cases}$$

(Pommerening, Helmert, Röger, & Seipp, 2015; Seipp, Pommerening, & Helmert, 2015)

Exercises

Ex. 5.1 — Compute the $\mathrm{h}^{\mathrm{flow}}(s_{init})$ and $\mathrm{h}^{\mathrm{pot}}(s_{init})$ for the problem from Fig. 1.

Ex. 5.2 — How can be flow heuristic improved with landmarks (e.g., from the LM-Cut heuristic)?

Ex. 5.3 — How can we modify objective of the LP for the potential heuristic so we still obtain admissible estimate for all reachable states?

References

Bonet, B. (2013). An admissible heuristic for SAS$^+$ planning obtained from the state equation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2268–2274.

Bonet, B., & Helmert, M. (2010). Strengthening landmark heuristics via hitting sets. In *19th European Conference on Artificial Intelligence, ECAI*, pp. 329–334.

Bonet, B., & van den Briel, M. (2014). Flow-based heuristics for optimal planning: Landmarks and merges. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 47–55.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*.

Helmert, M., Haslum, P., & Hoffmann, J. (2007). Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, (ICAPS)*, pp. 176–183.

Pommerening, F., Helmert, M., Röger, G., & Seipp, J. (2015). From non-negative to general operator cost partitioning. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (AAAI)*, pp. 3335–3341.

Seipp, J., Pommerening, F., & Helmert, M. (2015). New optimization functions for potential heuristics. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 193–201.