1 Definition: Transition System

A **planning task** is a 4-tuple $\Pi = \langle V, I, O, \gamma \rangle$

A **transition system** is a 6-tuple $T = \langle S, L, c, T, s_0, S^* \rangle$ where:

- S: finite set of states
- L: finite set of transition labels
- $c : L \mapsto \mathbb{R}_0^+$: label cost function
- $T \subseteq S \times L \times S$: transition relation
- $s_0 \in S$: initial state
- $S^* \subseteq S$: set of goal states

1.1 Forms and Properties

1.1.1 Heuristics

- Admissable: $h(s) \leq h^*(s)$
- Consistent: $h(s) \leq c(s, s') + h(s')$
- Goal aware: $h(s \in S^*) = 0$
- Safe: $h(s) = \infty \to h^*(s) = \infty$

1.1.2 Task forms and Misc.

- Positive normal form: All ops and goal are positive and flat
  - o is positive if pre(o) and eff(o) are positive
  - A logical proposition is positive if $\neg$ doesn't appear (including $\leftarrow$ and $\leftrightarrow$
  - o is flat, if eff(o) is flat (i.e. contains only atomics or $(x \rhd y)$)
- STRIPS: If all ops are STRIPS and goal follows: $\bigwedge_{v \in V} v$
  - o is STRIPS if pre(o) follows same form, and eff(o) is atomic.
- i-g Form: STRIPS form. $\{i, g\} \subseteq V$. $I := \{i\}$. $\gamma := \{g\}$. $\forall (o \in O)(|\text{pre}(o)| > 0)$
  - Any task can be made i-g form trivially, if already STRIPS.
- Transition Normal Form (TNF): $\forall (o \in O)(\text{vars}(\text{eff}(o)) = \text{vars}(\text{pre}(o)))$ and $\text{vars}(\gamma) = V$.
  - This can be achieved by 1) add auxiliary u to every dom(v)
  - 2) For each variable and value, add an operator than converts it to u for zero cost
  - 3) For all o, if a variable is in pre, but not in eff add it with the same value. If v in eff but not in pre, add $v = u$ in pre
- Algorithm is **sound** $\to$ plans are correct, and "unsolveable" answer is correct.

1.2 On-Set and Dominating states

- The on-set is the set of propositional variables that are true in a interpretation.
- Domainiting interpretations for $\text{on}(s) \subseteq \text{on}(s'), s, s' \in S$

1.3 Complexity

- $\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE}$
- (PlanEx)istance $\leq_p$ (B)ounded (C)ost PlanEx
- (PlanEx)istance $\in$ PSPACE
- True for both optimal and satisfycing

1.4 Search

- Uninformed: DFS, BFS, Iterative DFS
- Heuristic: Greedy BFS, A*, W-A*, IDA*
- Local Heuristic: Hill climbing, Sim. anneling, Beam

2 Satisfiability & Equivalence

φ satisfiable iff $\exists I : I \vDash \varphi$

φ valid iff $\forall I : I \vDash \varphi$

$\varphi \vDash \psi$ iff $\forall I : I \vDash \varphi \to I \vDash \psi$

$\varphi \equiv \psi$ iff $\varphi \vDash \psi \wedge \psi \vDash \varphi$

3 STRIPS Regression

Let $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_n$ be a conjunction of atoms, and o's add effects be $\{a_1, \ldots, a_k\}$, and delete effects $\{d_1, \ldots, d_l\}$

$\text{sregr}(\varphi, o) := \begin{cases} \bot \text{ if } \exists (i,j) \varphi_i = d_j \\ \text{pre}(o) \wedge (\{\varphi_1, \ldots, \varphi_n\} / \{a_1, \ldots, a_k\}) \text{ otherwise} \end{cases}$

4 SAT Planning style-algorithm

Algorithm 1: SAT Planning

procedure SATPLAN("Pi")
  for T in {0, 1, 2, ...} **do**
    $\varphi \leftarrow$ build_sat_formula(Π, T)
    $I \leftarrow$ sat_solver(φ)
    if I != none **then**
      return extract_plan(Π, T, I)
    end
  end
end

4.1 SAT: Operator Selection Clauses

- o_j^i (operator chosen at step i)
- $o_1^i \vee \ldots \vee o_n^i$ for $1 \leq i \leq T$
- $\neg o_j^i \vee \neg o_k^i$ for $1 \leq i \leq T, 1 \leq j < k \leq n$ (at most one operator per step)
  - This is equal to $\neg \left(o_j^i \wedge o_k^i \right)$

4.2 Transition Clauses

Precondition:

- $\neg o^i \vee \text{pre}(o)^{i-1}$ for $1 \leq i \leq T, o \in O$

Positive/Negative Effects Clauses:

- $\neg o^i \vee \neg \alpha^{i-1} \vee v^i$
- $\neg o^i \vee \alpha^{i-1} \vee \neg \delta^{i-1} \vee \neg v^i$

Positive/Negative Frame Clauses:

- $\neg o^i \vee \neg v^{i-1} \vee \delta^{i-1} \vee v^i$
- $\neg o^i \vee \alpha^{i-1} \vee v^{i-1} \vee \neg v^i$

where $\alpha = \text{effcond}(v, \text{eff}(o))$ $\delta = \text{effcond}(\neg v, \text{eff}(o))$

5 BDD Complexity

	Hash table	Formula	BDD
$s \in S?$	$O(k)$	$O(\lVert S \rVert)$	$O(k)$
$S := S \cup \{s\}$	$O(k)$	$O(k)$	$O(k)$
$S := S \setminus \{s\}$	$O(k)$	$O(k)$	$O(k)$
$S \cup S'$	$O(k\lvert S \rvert + k\lvert S' \rvert)$	$O(1)$	$O(\lVert S \rVert \lVert S' \rVert)$
$S \cap S'$	$O(k\lvert S \rvert + k\lvert S' \rvert)$	$O(1)$	$O(\lVert S \rVert \lVert S' \rVert)$
$S \setminus S'$	$O(k\lvert S \rvert + k\lvert S' \rvert)$	$O(1)$	$O(\lVert S \rVert \lVert S' \rVert)$
$\overline{S}$	$O(k2^k)$	$O(1)$	$O(\lVert S \rVert)$
$\{s \mid s(v) = \mathbf{T}\}$	$O(k2^k)$	$O(1)$	$O(1)$
$S = \emptyset?$	$O(1)$	co-NP-complete	$O(1)$
$S = S'?$	$O(k\lvert S \rvert)$	co-NP-complete	$O(1)$
$\lvert S \rvert$	$O(1)$	#P-complete	$O(\lVert S \rVert)$

6 BDD Operators

6.1 Conditioning

Conditioning variable v in formula φ to T or F:

- $\varphi\left[\frac{T}{v}\right]$ or $\varphi\left[\frac{F}{v}\right]$: restrict v to a given value
- Time: $O(|\varphi|)$

6.2 Forgetting

Forgetting (existential abstraction): allow both $v = T$ and $v = F$ and eliminate v.

- On formulas: $\exists v \varphi = \varphi\left[\frac{T}{v}\right] \vee \varphi\left[\frac{F}{v}\right]$
- On sets: $\exists v S = S\left[\frac{T}{v}\right] \cup S\left[\frac{F}{v}\right]$
- Time: $O(|\varphi|)$

6.3 Renaming

Renaming X to Y in formula φ, written $\varphi[X \to Y]$: replace all X by Y in φ (Y not present in φ).

- Time: $O(|\varphi|)$

7 BDD Transitions

7.1 Transition BDD

$T_{V(O)} = \bigvee_{o \in O} t_{V(o)}$

$\cdot t_{V(O)} = \text{pre}(o) \wedge \bigwedge_{v \in V} (\text{effcond}(v,e) \vee (v \wedge \neg \text{ effcond}(\neg v,e)) \leftrightarrow v')$

7.2 Apply

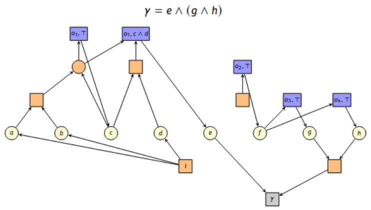
Algorithm 2: BDD Apply

procedure APPLY(reached, O)
  $B \leftarrow T_{V(O)}$
  $B \leftarrow$ bdd-intersection$(B, \text{reached})$
  for $v \in V$ **do**
    $B \leftarrow$ bdd-forget(B, v)
  end
  for $v \in V$ **do**
    $B \leftarrow$ bdd-rename(B, v', v)
  end
  return B
end

By then taking the union of the out and the previous reached, you get the reached for the following timestep.

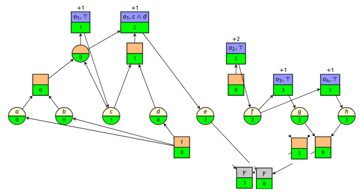
8 Relaxed Task Graph (RTG)

$o_1 = \langle c \vee (a \wedge b), c \wedge ((c \wedge d) \rhd e), 1 \rangle$
$o_2 = \langle \top, f, 2 \rangle$
$o_3 = \langle f, g, 1 \rangle$
$o_4 = \langle f, h, 1 \rangle$



8.1 $h^{\max}$ & h^{add}

- $h^{\max} \leq h^+ \leq h^{\text{FF}} \leq h^{\text{add}}$
- $h^{\max(s)} = \infty \leftrightarrow h^{+(s)} = \infty \leftrightarrow h^{\text{FF}(s)} = \infty \leftrightarrow h^{\text{add}(s)} = \infty$
- $h^{\max}$ and h^{add} $\to$ admissible and consistent
- h^+ and h^{FF} $\to$ NOT admissible and consistent
- All are safe and goal-aware.



Above, only nodes where $h^{\max}$ (left) and h^{add} (right) differ are recorded.

- $h^{\max}$: Pick the max predecessor at AND node, and the min at OR
- h^{add}: Add the predecessors at AND node, and pick the min at OR
- Both can be computed efficently by expanding the minimum/ newest node that CAN be updated

8.2 h^{FF} and Best Achiever Graphs (G)

- BAG can be achieved by removing all incoming edges into a OR node, except the minimum cost one
- h^{FF} can be achieved by adding all operators participating in the G^{add} for h^{add}
- G are also useful for analysis when h^{add} overapprox and when $h^{\max}$ under approx.

9 Invariant/Mutex/FDR

- Validating invariant is AS HARD as planning.
- Mutex group is a set of variables where AT MOST one can be true
- A Mutex cover is a set of mutex groups where each variable occurs in exactly one group
- A mutex group is positive if it contains no negations of variables

9.1 Mutex-based Reformulation of Propositional

Given a conflict-free propositional planning task Π w/ positive mutex cover $\{G_1, \ldots, G_N\}$

- In all condition where variable $v \in G_i$ occurs, replace v with $v_{G_i} := v$
- In all effects e where variable $v \in G_i$ occurs,
  - Replace all atomic add effects v with $v_{G_i} := v$
  - Replace all atomic delete effects $\neg v$ with:
    - $\left(v_{G_i} = v \wedge \neg \bigvee_{v' \in G_i \setminus \{v\}} \text{effcond}(v', e) \right) \rhd v_{G_i} := \text{none}$
    - Practically, this means, if v_{G_i} is being deleted AND IS NOT BEING SET TO ANOTHER VARIABLE, set it to none. This is keep it conflict-free.

The consistency condition consist(e) prohibits two simultaneous assignments to the same mutex group.
I.e. $\neg(\text{effcond}(v := d, e) \wedge \text{effcond}(v := d', e))$

9.2 SAS+

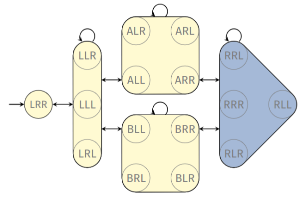
An operator of an FDR operator is a SAS+ operator if

- pre(o) is a satisfiable conjunction of atoms, and
- eff(o) is a conflict-free conjunction of atomic effects.

An FDR task is a SAS+ task if all operators are SAS+ and the goal is a satisfiable conjunction of atoms

10 Abstraction

- $s \in \gamma \to \alpha(s) \in \gamma_\alpha$
- $\langle s, o, s' \rangle \in \mathcal{T} \to \langle \alpha(s), o, \alpha(s') \rangle \in \mathcal{T}_\alpha$
- Abstraction are composable, i.e. $(\beta \circ \alpha)$ is a valid abstraction.
- Abstraction are surjective.
- Abstraction uses coarsening/refinement terminology.
- $h^{\beta \circ \alpha} \leq h^\alpha \leq h^*$
  - h^α is safe, goal-aware, admissible and consistent.



$h^\alpha(\{p \mapsto L, t_A \mapsto R, t_B \mapsto R\}) = 3$

10.1 Additivity

- Orthogonal α_1 & α_2: If $\forall (t \in \mathcal{T})(\alpha_1(s) = \alpha_2(t)) \vee (\alpha_2(s) = \alpha_2(t))$, where $t = \langle s, \ell, t \rangle$
- Affect α for ℓ, if $\langle \alpha(s), \ell, \alpha(t) \rangle$, where $\alpha(s) \neq \alpha(t)$
- Also ortogonal if no label affects both abstractions.
- The sum of orthogonal h^α is safe, goal-aware, admissible and consistent.

10.2 Projections & Pattern Databases

- A projection (π_P) is a special kind of abstraction
- $\pi_P : S \to S'$ is defined as$\pi_{P(s)} := s|_P$ (where $s|_{P(v)} := s(v)$ for all v)
  - I.e. we condition a state on a single variable assignment.
- The heuristic induced by π_P, we call a PDB heuristic (h^P)
- Syntactic projections ($\Pi|_P$), gives the projected planning task, by practically, removing the variables in the projection, from $\langle P, I|_P, \{o|_P, o \in O\}, \gamma|_P \rangle$

$$\begin{tikzcd}
\Pi \arrow[r, "\text{induced TS}"] \arrow[dd, "\text{syntactic projection}"'] & \mathcal{T}(\Pi) \arrow[d, "\text{abstract TS}"] \\
 & \mathcal{T}(\Pi)^{\pi_P} \\
\Pi|_P \arrow[r, "\text{induced TS}"] & \mathcal{T}(\Pi|_P)
\end{tikzcd}$$

These two are isomorphic

10.3 PDB Lookup

- PDBs are precomputed before search.
- Is effective done via perfect hashing.
  - $N_i := \Pi_{j=1}^{i-1} |\text{dom}(v_j)|$
  - $\text{PDB-index}(s) := \Sigma_{i=1}^{k} N_i \cdot s(v_i)$

11 Merge and Shrink

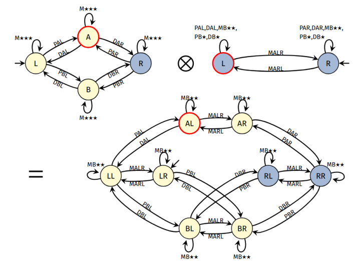
- Idea:
  - 1) Project Π to atomic projection
  - 2) Merge two of the resulting transition systems ($\mathcal{T}|_P$)
  - 3) Shrink combined $\mathcal{T}'$ by abstracting more states
  - 4) Pick the result as the first transition system and go to step (2)

Algorithm 3: Merge and Shrink

procedure MERGE-AND-SHRINK(Π, B)
  $F \leftarrow F(\Pi)$
  while $|F| > 1$ **do**
    "type" $\leftarrow$ pick-merge-or-shrink(F)
    if type $=$ merge **then**
      $\mathcal{T}_1 \leftarrow \text{pick}(F)$
      $\mathcal{T}_2 \leftarrow \text{pick}(F \setminus \{\mathcal{T}_1\})$
      $F \leftarrow (F \setminus \{\mathcal{T}_1, \mathcal{T}_2\}) \cup \{\mathcal{T}_1 \otimes \mathcal{T}_2\}$
    end
    if type $=$ shrink **then**
      $\mathcal{T} \leftarrow \text{pick}(F)$
      $\beta \leftarrow$ pick-abstraction(B)
      $F \leftarrow (F \setminus \{\mathcal{T}\}) \cup \{\mathcal{T}^\beta\}$
    end
  end
  return $F[0]$
end

$\mathcal{T}^{\pi_{\{\text{package}\}}} \otimes \mathcal{T}^{\pi_{\{\text{truck A}\}}}$:

$S_\otimes = S_1 \times S_2$



11.1 Factored Transition System (FTS or F)

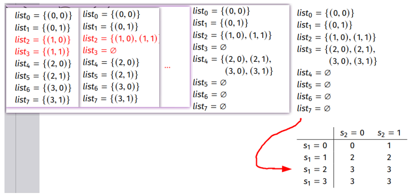
- A finite set $F = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, where all share ℓ and cost(s).
- FTS induced by Π is $F(\Pi) = \{\mathcal{T}^{\pi_v} \mid v \in V\}$
- $\bigotimes F \sim \mathcal{T}(\Pi)$ is the transition system that induced it.

11.2 Merge Strategies

- f-preserving strategy
  - Combine nodes with identical g and h value
    - *Rational*: Preserves h and overall graph shape
  - Tie-breaking criterion, prefer merging high $g + h$
    - *Rational*: High values heuristic estimates are less likely to be explored by A*, so it can be more inprecise.

11.3 Merge and Shrink - Effective Shrink

- This is done by first converting the combined table (which is a cross product of two abstractions) into a linked list. And then...



11.4 Merge and Shrink Lookup

- Looking up the heuristic value in a MAS system requires looking up from first single variables and then into the larger merge, i.e.

At the end, our heuristic is represented by six tables:

- three one-dimensional tables for the atomic abstractions:

T_{package}	L	R	A	B
	0	1	2	3

$T_{\text{truck A}}$	L	R
	0	1

$T_{\text{truck B}}$	L	R
	0	1

- two tables for the two merge and subsequent shrink steps:

$T^1_{\text{m\&s}}$	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	0	1
$s_1 = 1$	2	2
$s_1 = 2$	3	3
$s_1 = 3$	3	3

$T^2_{\text{m\&s}}$	$s_2 = 0$	$s_2 = 1$
$s_1 = 0$	1	1
$s_1 = 1$	1	0
$s_1 = 2$	2	2
$s_1 = 3$	3	3

- one table with goal distances for the final transition system:

T_h	$s = 0$	$s = 1$	$s = 2$	$s = 3$
$h(s)$	3	2	0	1

11.5 Label Reduction

11.5.1 Definition

- A label reduction $\langle \lambda : L \to L', c' : L \to \mathbb{R}^+ \rangle$, such that $\ell \in L, c'(\lambda(\ell)) \leq c(\ell)$
- The label-reduced transition system $\mathcal{T}^{\langle \lambda, c' \rangle} = \langle S, L', c', \{\langle s, \lambda(\ell), t \rangle \mid \langle s, \ell, t, \rangle \in T\}, s_0, S_* \rangle$

11.5.2 Properties

- ℓ is alive in F if all $\mathcal{T}' \in F$ have ℓ, dead otherwise
  - ℓ locally subsumes ℓ' in $\mathcal{T}$ if for all transition $\langle s, \ell', t \rangle$ there is also $\langle s, \ell, t \rangle$
  - It also globally subsumes if this is true in all $\mathcal{T} \in F$
  - ℓ and ℓ' are locally equivilant if ℓ locally subsumes ℓ' and vice versa.
  - ℓ and ℓ' are $\mathcal{T}$-combinable if there are locally equivilant in all transition systems $\mathcal{T}' \in F \setminus \{\mathcal{T}\}$

11.5.3 Exact Label Reduction

- The label reduction is exact (No loss in information), if for all $\ell_1, \ell_2 \in L$:
  - Either ℓ_1 or ℓ_2 globally subsumes the other
  - ℓ_1 and ℓ_2 are $\mathcal{T}$-combinable for some $\mathcal{T} \in F$

12 Landmarks

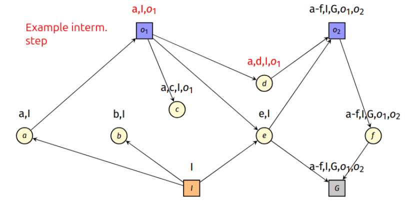
- Fact landmark: This must be visited at least once
- Disjunctive Action: One action from the set must be performed
- Network flow: Fact consumption should be balanced.

12.1 Relaxed Task Graph Landmarks

- Causal Landmark λ: for I if $\gamma \models \lambda$ OR if for all plans $\langle o_1, \ldots, o_n \rangle$ at least one $\text{pre}(o_i) \models \lambda$
- Causal fact landmark: Same as above, but $\lambda := v$
- To calculate the RTG landmarks, first instantiate all nodes with all potential landmarks. Then perform this on RTG

$$\text{LM}(n) = \{n\} \bigcap_{n' \to n \in A} \text{LM}(n') \quad \text{if type}(n) = \vee$$

$$\text{LM}(n) = \{n\} \bigcup_{n' \to n \in A} \text{LM}(n') \quad \text{if type}(n) = \wedge$$



$\text{LM}(d) = \{d\} \cup \text{LM}(o_1)$

12.2 Minimum Hitting Set

- The minimum hitting set, the minimal cost set of operators such that a for a set of set of operators, each set contains one of the operators selected.
- This is relevant for combining disjunctive action landmarks, which is not admissible if additively combined.

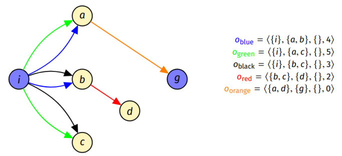
12.3 LM-Cut

12.3.1 Justification Graph

- Precondition choice function: $P : O \to V$, maps any operator in a task Π to one of it's preconditions.
- Justification graph: $\langle V, E := \{\langle P(o), a \rangle \mid o \in O, a \in \text{add}(o)\} \rangle$

Example (Precondition Choice Function)

$P(o_{\text{blue}}) = P(o_{\text{green}}) = P(o_{\text{black}}) = i$, $P(o_{\text{red}}) = b$, $P(o_{\text{orange}}) = a$

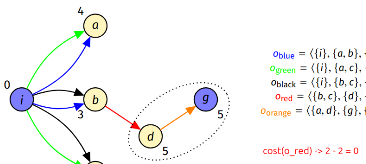


- **Cut** ($C \subseteq E$): A subset of edges, such that ALL paths $i \xrightarrow{*} g$ contains $e \in C$. Doesn't have to include all.

12.4 LM-Cut Algorithm

Initialize $h^{\text{LM-Cut}}(I) := 0$. Then iterate:

- Compute $h^{\max}$ using a RTG. Stop if $h^{\max}(g) = 0$
- Compute Justification graph for the P/pcf that chooses the precondition with the MAXIMAL $h^{\max}$ value.
- Determine goal zone V_g (i.e. all nodes with a zero cost path to g).
- Compute the cut L that contains labels of all edges $\langle v, o, v' \rangle$ such that $v \notin V_g, v' \in V_g$, and v CAN be reached from i without traversing V_g. *It is guaranteed that* $\text{cost}(L) > 0$
- Increase $h^{\text{LM-Cut}}(I)$ by $\text{cost}(L)$ (i.e. the cost of the cut).
- Decrease $\text{cost}(o)$ by $\text{cost}(L)$ for all $o \in L$.



round	$P(o_{\text{orange}})$	$P(o_{\text{red}})$	landmark	cost
1	d	b	$\{o_{\text{red}}\}$	2
			$h^{\text{LM-cut}}(I)$	2

13 Integer and Linear Programs

13.1 Integer Programs (IP)

Solving is an NP-hard problem.

Consists of:

- finite set of integer variables V
- finite set of linear inequalities (constraints) over V
- an objective function, which is a linear combination of V
- Whether it should be minimized or maximized.

Example:

- minimize $3X_{O_1} + 4X_{O_2} + 5X_{O_3}$ subject to
  - $X_{O_4} \geq 1$
  - $X_{O_1} + X_{O_2} \geq 1$
  - $X_{O_1} + X_{O_3} \geq 1$
  - $X_{O_2} + X_{O_3} \geq 1$
  - $X_{O_1}, X_{O_2}, X_{O_3}, X_{O_4} \geq 0$

13.2 Linear Programs (LP)

Consist of the same as Integer programs, but with real valued variables and constraints.

Can be solved in polynomial time wrt. the number of constraints with SIMPLEX.

An LP can serve as an upper bound to an IP (LP relaxation).problem

13.3 Standard Maximization/Minimization Problem

- Given a vector of objective coefficience $c = \mathbb{R}^{N \times 1}$, bounds $b = \mathbb{R}^{N \times 1}$, and coefficients $A = \mathbb{R}^{M \times N}$

Optimize: Maximize $c^T x$ subject to $Ax \leq b$ and $x \geq 0$

- The **dual** to the maximization problem is the minimization (These are equal).

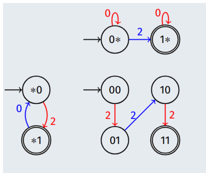
Optimize: Minimize $b^T x$ subject to $A_T x \geq c$ and $x \geq 0$

14 Cost Partitioning

- Is admissible.
- Principal: Distribute cost of operators between h such that $\sum_n^{i=1} \text{cost}_{i(o)} \leq \text{cost}(o)$ for all $o \in O$
- Also called the **cost partitioning constraint**
- A general cost partitioning the upholds this constraint is admissible.

Example:

- zero-one cost partitioning: Set cost to one in one abstraction and zero everywhere else
- uniform cost partitioning: Distribute the cost equally among abstractions



Heuristic value: 2 + 2 = 4

14.1 Saturated Cost Partitioning

- mscf: minimum saturated cost function

$$\text{mscf}(o) = \max\left(0, \max_{\alpha(s) \xrightarrow{o} \alpha(t)} (h^\alpha(s) - h^\alpha(t))\right)$$

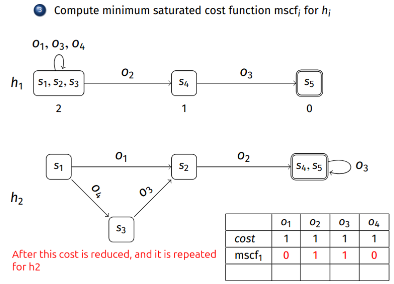
Algorithm

Iterate

- Pick a heuristic h_i that hasn't been picked. Terminate if none is left.
- Compute h_i given current cost
- Compute scf_i (ideally mscf_i) for h_i
- Decrease $\text{cost}(o)$ by $\text{scf}_i(o)$ for all o

Example:

Compute minimum saturated cost function mscf_i for h_i



	o_1	o_2	o_3	o_4
cost	1	1	1	1
mscf_1	0	1	1	0

14.2 Relation to LM-Cut

- LM-Cut computes SCP over disjunctive action landmarks

abstraction

- Let Π be a planning task and $\mathcal{L}$ be a disjunctive action landmark. The minimum saturated cost function for $\mathcal{L}$ is:

$$\text{mscf} = \begin{cases} \min\limits_{o' \in \mathcal{L}} \text{cost}(o') & \text{if } o \in \mathcal{L} \\ 0 & \text{otherwise} \end{cases}$$

14.3 Optimal Cost Partitioning

- The Optimal Cost Partitioning can be calculated as an LP for disjunctive action landmarks.
- The bound is the $\text{cost}(o)$ as defined by the cost partitoning constraint.
- The variables are the cost of the landmark.
- The goal is to maximize their sum.

Dual

- This can be converted to the **dual**.
- Instead the variables is whether an operator has been applied.
- The goal is to minimize the sum of applied operators times their cost
  - $\sum_o \text{Applied}_o \cdot \text{cost}(o)$
- The constrain is that all landmarks must be hit.
  - $\sum_{o \in L} \text{Applied}_o \geq 1$ for all landmarks L.

14.4 Post-hoc Optimization (h^{PhO})

- $h^G \leq h^{\text{PhO}} \leq h^{\text{OCP}}$

Assuming that you have a PDB, you can optimize them even further with LPs. Operators can be included if they affect an abstraction α i.e., the ℓ moves into a new abstract state $(s \neq t)$

Linear Program

Minimize $X_A + X_B + X_C$ subject to

abst1	$X_A + X_B \geq h^{\{A,B\}}(s) = 6$
abst2	$X_A + X_C \geq h^{\{A,C\}}(s) = 6$
abst3	$X_B + X_C \geq h^{\{B,C\}}(s) = 6$
non-negative constraints	$X_A \geq 0, X_B \geq 0, X_C \geq 0$

$\Rightarrow$ any plan has at least cost 9.

15 Network Flow Heuristics

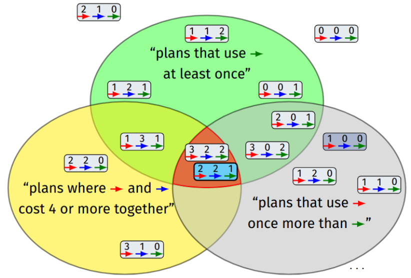
Definition (Flow Constraint)

Let $\Pi = \langle V, I, O, \gamma \rangle$ be a task in transition normal form.
The flow constraint for atom a in state s is

$$[a \in s] + \sum_{o \in O: a \in \text{eff}(o)} \text{Count}_o = [a \in \gamma] + \sum_{o \in O: a \in pre(o)} \text{Count}_o$$

- Count_o: LP variable for the number of occurrences of operator o.
- Neutral operators either appear on both sides or on none.

16 Operator Counting



Definition (Operator-counting IP/LP Heuristic)

The operator-counting integer program IP_C for a set C of operator-counting constraints for state s is

$$\text{Minimize} \sum_{o \in O} \text{cost}(o) \cdot \text{Count}_o \quad \text{subject to}$$

C and $\text{Count}_o \geq 0$ for all $o \in O$,

where O is the set of operators.

The IP heuristic h^{IP}_C is the objective value of IP_C,
the LP heuristic h^{LP}_C is the objective value of its LP-relaxation.

If the IP/LP is infeasible, the heuristic estimate is ∞.