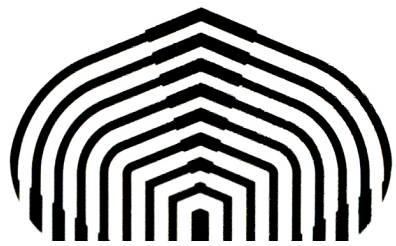
Linear Programming in Optimal Classical Planning

Blai Bonet

Universidad Simón Bolívar, Venezuela

UC3M. June 2019



Model for classical planning

Simplest model: full information and deterministic operators (actions):

- (finite) state space S
- (finite) operator space O
- initial state $s_{init} \in S$
- goal states $S_G \subseteq S$
- applicable operators $O(s) \subseteq A$
- **deterministic transition function** f such that $f(s, o)$ is state that results of applying $o \in O(s)$ in s
- operator cost $c(o)$ for each o

Solution is sequence of applicable operators that map initial state to goal

Solution $\langle o_0, o_1, \ldots, o_{n-1} \rangle$ is optimal if its cost $\sum_{0 \leq i < n} c(o_i)$ is minimum

Planning as search in the space of states

Computation of plans (solutions) as **search in space of states** of path that goes from initial state to a goal state

Search can be done efficiently in **explicit graphs**

Main obstacle: implicit model specified with factored language is typically of exponential size

Work around: search in **implicit graph** with guiding information

Algorithm: (in optimal classical planning) A* with **admissible heuristic**

Specification of models

Models specified using **representation language**

These languages are **factored languages** that permit specification of very large problems using few symbols

STRIPS: Propositional language

Representation language based on propositions

Propositions evaluate to true/false at each state (e.g. light is on, package is in Madrid, elevator is in second floor, etc)

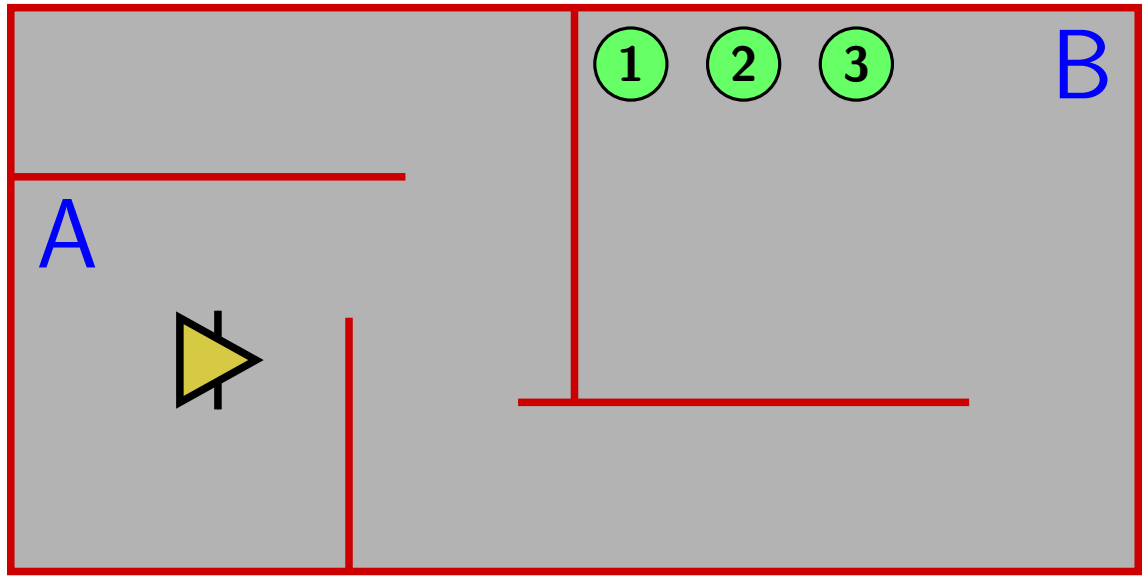
STRIPS task $P = (F, I, G, O)$:

- Set F of propositions used to describe states
- Initial state I is subset of propositions: those true at initial state
- Goal description G is subset of propositions: those we want to hold at goal
- Operators in O change truth-value of propositions

Each operator o characterized by three F-subsets:

- Precondition pre(o): things that need to hold for o to be "applicable"
- Positive effects add(o): things that become true when o is applied
- Negative effects del(o): things that become false when o is applied

Example: Gripper

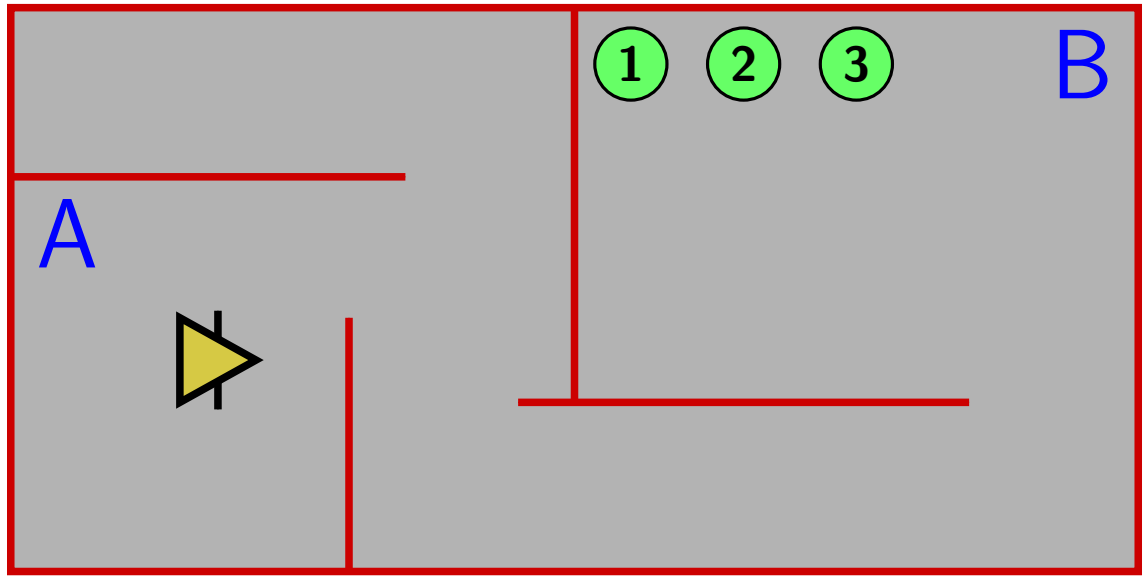


- Bunch of balls in room B
- Robot with left and right gripper, each one may hold a ball
- **Goal:** move all balls to room A

Robot may:

- move between rooms A and B; e.g. Move(A, B)
- use grippers to pick and drop balls from rooms; e.g. Pick(left, b_3, B)

Example: Gripper



Variables:

– robot's position: room A or B

– position of each ball b_i: either room A or B, or left or right gripper

States: **valuation for vars** (#states $> 2^{n+1}$ for problem with n balls)

Actions:

– **deterministic transition function:** from state to next state

– may have preconditions; e.g. can drop ① in A only if at A and holding it

Example: Gripper in PDDL

```
(define (domain gripper)
  (:predicates (room ?r) (ball ?b) (gripper ?g) (at-robby ?r) (at ?b ?r)
               (free ?g) (carry ?o ?g))

  (:action move
    :parameters (?from ?to)
    :precondition (and (room ?from) (room ?to) (at-robby ?from))
    :effect (and (at-robby ?to) (not (at-robby ?from))))

  (:action pick
    :parameters (?b ?r ?g)
    :precondition (and (ball ?b) (room ?r) (gripper ?g) (at ?b ?r) (at-robby ?r) (free ?g))
    :effect (and (carry ?b ?g) (not (at ?b ?r)) (not (free ?g))))

  (:action drop
    :parameters (?b ?r ?g)
    :precondition (and (ball ?b) (room ?r) (gripper ?g)
    (carry ?b ?g) (at-robby ?r))
    :effect (and (at ?b ?r) (free ?g) (not (carry ?b ?g))))
)

(define (problem p1)
  (:domain gripper)
  (:objects A B left right b1 b2 b3)
  (:init (room A) (room B) (gripper left) (gripper right) (ball b1) (ball b2) (ball b3)
         (at-robby A) (at b1 B) (at b2 B) (at b3 B) (free left) (free right))
  (:goal (and (at b1 A) (at b2 A) (at b3 A))))
```

Heuristic functions in search

Provide information to A* to make search more efficient

Difference in performance may be important (exponential speed up)

Heuristic is function h that for state s returns non-negative estimate $h(s)$ of cost to go from s to goal state

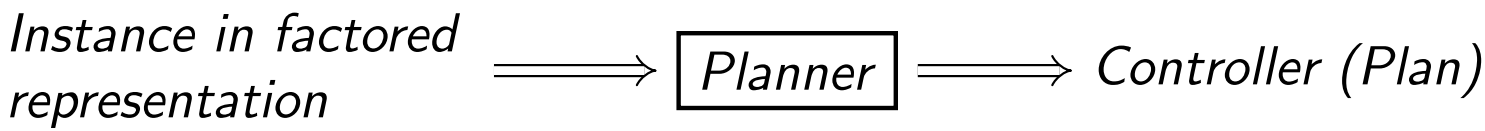
Properties:

- **Goal-aware:** $h(s) = 0$ if s is goal
- **Admissible:** $h(s) \leq$ "min cost to reach goal from s"
- **Consistent:** $h(s) \leq c(o) + h(f(s, o))$ where $o \in O(s)$ (triangular ineq.)

Basic facts about heuristics

1. Goal-aware + Consistent $\implies$ Admissible
2. A* returns **optimal path** if h is admissible
3. A* is **optimal algorithm** if h is consistent
4. If $h_1 \leq h_2$ and both consistent, A* with h_2 is "better" than A* with h_1

Domain-independent planning



Heuristic function must be computed automatically from input

- For effective planner, heuristic must be informative (i.e. must provide good guidance)
- For computing optimal plans, heuristic must be admissible
- This is the **main challenge** in optimal classical planning

Recipe for admissible heuristics

As proposed by Judea Pearl, best way to obtain admissible estimate $h(s)$ for task P:

- **Relax task** P from s into "simpler" task $P'(s)$
- Solve $P'(s)$ optimally to obtain **cost** $h^*_{P'}(s)$ of reaching goal in P' from s
- Set $h(s) := h^*_{P'}(s)$

Often, either

- $P'(s)$ is solved each time its value is needed, or
- P' is **solved entirely** and the estimates $h^*_{P'}(s)$ are stored in a table. Computing $h(s)$ is just a lookup operation into table (constant time)

Fundamental task: Combine multiple heuristics

Given **admissible** heuristics $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ for task P, how do we combine them into a new admissible heuristic?

- Pick one (fixed or random): $\mathcal{H}(s) = h_i(s)$
- Take maximum: $h^{\max}_{\mathcal{H}}(s) = \max\{h_1(s), h_2(s), \ldots, h_n(s)\}$
- Take sum: $h^{\text{sum}}_{\mathcal{H}}(s) = h_1(s) + h_2(s) + \cdots + h_n(s)$

First two guarantee admissibility, last doesn't. However, $h^{\max}_{\mathcal{H}} \leq h^{\text{sum}}_{\mathcal{H}}$

We would like to use $h^{\text{sum}}_{\mathcal{H}}$ but need admissibility

Cost relaxation

Given:

- Task P (either STRIPS or other) with operator costs c, denoted by P_c
- Method to relax P_c into P'_c

Additional relaxation:

- Before calculating relaxation P'_c, change cost function from c to c'
- Relaxed task is $P'_{c'}$ of original task P_c

Result:

- If relaxation method yields admissible (resp. consistent) estimates, relaxed task $P_{c'}$ also yields admissible (resp. consistent) estimates when $c' \leq c$
- That is, $h^*_{P''}(s) \leq h^*_{P'}(s) \leq h^*_P(s)$ for $P'' = P'_{c'}$ when $c' \leq c$

Cost partitioning

A task P with costs $c(\cdot)$ can be decomposed into $\mathcal{P} = \{P_{c_1}, P_{c_2}, \ldots, P_{c_n}\}$ where each cost function $c_i(\cdot)$ satisfies $c_i(o) \leq c(o)$ for all operators o

Given heuristics $\mathcal{H} = \{h_1, h_2, \ldots, h_n\}$ where h_i is for problem P_{c_i}

$$h^{\max}_{\mathcal{H}}(s) \;=\; \max\{h_1(s), h_2(s), \ldots, h_n(s)\} \;\leq\; h^*(s)$$

If $c_1(o) + c_2(o) + \cdots + c_n(o) \leq c(o)$ for each operator o,

$$h^{\mathsf{sum}}_{\mathcal{H}}(s) \;=\; h_1(s) + h_2(s) + \cdots + h_n(s) \;\leq\; h^*(s)$$

We say that $\{c_1, c_2, \ldots, c_n\}$ is a **cost partitioning**. The **optimal cost partitioning (OCP)** maximizes $h^{\mathsf{sum}}_{\mathcal{H}}(s)$ (it depends on s)

Linear programming

LP (or linear optimization) is method to optimize **linear objective** (function) subject to **linear constraints** on variables

Standard forms:

Minimize $c^T x$

subject to

$$Ax \geq b$$
$$x \geq 0$$

Maximize $c^T x$

subject to

$$Ax \leq b$$
$$x \geq 0$$

Pseudo-LP for optimal cost partitioning

Decision variables: (heuristic value) $h_i(s)$, (cost partition) $c_i(o)$

Maximize $\sum_{1 \leq i \leq n} h_i(s)$

subject to

[linear constraints that "calculate" $h_i(s)$]

$\sum_{1 \leq i \leq n} c_i(o) \leq c(o)$ (for each operator o)

$0 \leq c_i(o)$ **(non-negative operator costs)**

Exact LP will depend on the relaxation method. Optimal cost-partitioning heuristic for state s denoted by $h^{\mathsf{OCP}}_{\mathcal{H}}(s)$ or $h^{\mathsf{OCP}}_{\mathcal{C}}(s)$

(Action) Landmarks

(Disjunctive action) landmark for task $P(s)$ is subset $L \subseteq O$ of operators such that any plan for state s **must execute** at least some operators in L

STRIPS Task $P = (F, I, G, O)$ where:

- $F = \{i, p, q, r, g\}$, $I = \{i\}$, $G = \{g\}$, $O = \{o_1, o_2, o_3, o_4\}$
- $o_1[3] : i \rightarrow p, q$
- $o_2[4] : i \rightarrow p, r$
- $o_3[5] : i \rightarrow q, r$
- $o_4[0] : p, q, r \rightarrow g$

Optimal plan: (o_1, o_2, o_4) with cost 7

Landmarks for I: $L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}, \ldots$

Non-landmarks for I: $\{o_1\}, \{o_2\}, \{o_3\}$

There are efficient methods to compute landmarks

Landmark heuristic

Given landmark $L = \{o_1, o_2, \ldots\}$ for state s, $h^L(s) = \min\{c(o) : o \in L\}$

In example, $\mathcal{L} = \{L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}\}$ is collection of landmarks for initial state. The associated heuristics are $\mathcal{H} = \{h^{L_1}, h^{L_2}, h^{L_3}, h^{L_4}\}$

- $h^{\max}_{\mathcal{H}}(I) = \max\{h^{L_1}(I), h^{L_2}(I), h^{L_3}(I), h^{L_4}(I)\} = \max\{3, 3, 4, 0\} = 4$
- $h^{\text{sum}}_{\mathcal{H}}(I) = 3 + 3 + 4 + 0 = 10$ **(non-admissible since $h^*(I) = 7$)**
- For cost partitioning given by

	c_1	c_2	c_3	c_4	$\sum$
$o_1[3]$	1	2			3
$o_2[4]$	1		3		4
$o_3[5]$		2	3		5
$o_4[0]$					0

cost-partitioning for $h^{\text{sum}}_{\mathcal{H}}$ yields $1 + 2 + 3 + 0 = 6$ **(admissible)**

Optimal cost-partitioning for landmarks

Optimal cost-partitioning $h^{\text{OCP}}_{\mathcal{L}}$ for collection $\mathcal{L}$ may be computed efficiently:

Decision variables: (heuristic value) h_i, (cost partition) $c_i(o)$

Maximize $\sum_{L_i \in \mathcal{L}} h_i$

subject to

$$h_i \leq c_i(o) \qquad \text{(for each } L_i \in \mathcal{L} \text{ and } o \in L_i\text{)}$$

$$\textstyle\sum_i [\![o \in L_i]\!] c_i(o) \leq c(o) \qquad \text{(for each operator } o\text{)}$$

$$0 \leq c_i(o) \qquad \text{(for each } L_i \in \mathcal{L} \text{ and } o \in L_i\text{)}$$

$$0 \leq h_i \qquad \text{(for each } L_i \in \mathcal{L}\text{)}$$

Optimal cost-partitioning for landmarks

Optimal cost-partitioning $h^{\text{OCP}}_{\mathcal{L}}$ for collection $\mathcal{L}$ may be computed efficiently:

Decision variables: (heuristic value) h_i

Maximize $\sum_{L_i \in \mathcal{L}} h_i$

subject to

$\sum_i [\![o \in L_i]\!] h_i \leq c(o)$ (for each operator o)

$0 \leq h_i$ (for each $L_i \in \mathcal{L}$)

Another way to exploit landmarks: Hitting sets

From $\mathcal{L} = \{L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}\}$:

- any plan must execute **at least two** operators in $U = \{o_1[3], o_2[4], o_3[5]\}$
- **cheapest 2-subset** is to select o_1 and o_2 for cost of 7
- Therefore, 7 is admissible estimate on cost of any plan (exact in example)
- Formally, we say that $\{o_1, o_2\}$ is **minimum-cost hitting set** for U

Hitting sets:

- Classical problem in CS (in Karp's original list of NP-Complete problems)
- **Task:** Given family $\mathcal{F}$ of subsets of universe U, costs $c(u)$ for each $u \in U$, and bound K, determine whether there is subset $H \subseteq U$ such that:
  - $S \cap H \neq \emptyset$ for every $S \in \mathcal{F}$
  - $c(H) \leq K$ where $c(H) = \sum_{u \in H} c(u)$

Hitting sets as (integer) LP

Calculation of hitting sets can be done with LP with **integer variables**

Task: universe U with costs $c(u)$, and family $\mathcal{F}$ of U-subsets

Decision variables: (binary variable) x_u for each $u \in U$

$$\text{Minimize} \sum_{u \in U} c(u) \cdot x_u$$

subject to

$$\sum_{u \in S} x_u \geq 1 \qquad \text{(for each } S \in \mathcal{F}\text{)}$$

$$x_u \in \{0, 1\} \qquad \text{(for each } u \in U\text{)}$$

Unfortunately, we don't know if ILPs can be solved in **polynomial time**

Duality

For any linear optimization problem, called **primal problem**, there is a related LP problem called **dual problem:**

- objective in dual is maximization (resp. minimization) if objective in primal is minimization (resp. maximization)
- each variable in dual corresponds to constraint in primal, and vice versa
- each constraint in dual corresponds to variable in primal, and vice versa

Many algorithms, like Simplex, work with both problems simultaneously

Often theoretical insight is gained by studying the dual problem

Primal and dual problems

Dual of minimization primal in standard form:

Primal

Minimize $c^T x$

subject to

$$Ax \geq b$$
$$x \geq 0$$

Dual

Maximize $b^T y$

subject to

$$A^T y \leq c$$
$$y \geq 0$$

Weak duality: primal objective is **lower bounded** by dual objective

Strong duality: both problems have same objective value if primal is feasible and bounded

Primal and dual problems

Dual of maximization primal in standard form:

Primal

Maximize $c^T x$

subject to

$$Ax \leq b$$
$$x \geq 0$$

Dual

Minimize $b^T y$

subject to

$$A^T y \geq c$$
$$y \geq 0$$

Weak duality: primal objective is **upper bounded** by dual objective

Strong duality: both problems have same objective value if primal is feasible and bounded

Hitting sets vs. optimal cost partitionings

For $\mathcal{L} = \{L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}\}$

Let us begin with LP for hitting sets for $\mathcal{L}$

Minimize $\sum_{o_i \in O} c(o_i) \cdot x_i$

subject to

$\sum_{o_i \in L} x_i \geq 1$ (for each $L \in \mathcal{L}$)

$x_i \in \{0, 1\}$ (for each $o_i \in O$)

Hitting sets vs. optimal cost partitionings

For $\mathcal{L} = \{L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}\}$

Let us begin with LP for hitting sets for $\mathcal{L}$

Minimize $3x_1 + 4x_2 + 5x_3 + 0x_4$

subject to

$$
\begin{aligned}
x_1 + x_2 &\geq 1 && \text{(for landmark } L_1\text{)} \\
x_1 + x_3 &\geq 1 && \text{(for landmark } L_2\text{)} \\
x_2 + x_3 &\geq 1 && \text{(for landmark } L_3\text{)} \\
x_4 &\geq 1 && \text{(for landmark } L_4\text{)} \\
x_i \in \{0, 1\} & && \text{(for each } o_i \in O\text{)}
\end{aligned}
$$

Hitting sets vs. optimal cost partitionings

For $\mathcal{L} = \{L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}\}$

Relaxation of integer variables

Minimize $3x_1 + 4x_2 + 5x_3 + 0x_4$

subject to

$$
\begin{aligned}
x_1 + x_2 \quad\quad\quad\quad &\geq 1 && \text{(for landmark } L_1\text{)} \\
x_1 \quad\quad + x_3 \quad\quad &\geq 1 && \text{(for landmark } L_2\text{)} \\
x_2 + x_3 \quad\quad &\geq 1 && \text{(for landmark } L_3\text{)} \\
x_4 &\geq 1 && \text{(for landmark } L_4\text{)} \\
0 \leq x_1, x_2, x_3, x_4 &\leq 1 && \text{(for each } o_i \in O\text{)}
\end{aligned}
$$

Hitting sets vs. optimal cost partitionings

For $\mathcal{L} = \{L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}\}$

Calculate dual LP (it has same objective value)

Maximize $y_1 + y_2 + y_3 + y_4$

subject to

$\sum_j [\![o \in L_j]\!] y_j \leq c(o)$ (for each operator o)

$0 \leq y_j$ (for each $L_j \in \mathcal{L}$)

Hitting sets vs. optimal cost partitionings

For $\mathcal{L} = \{L_1 = \{o_1, o_2\}, L_2 = \{o_1, o_3\}, L_3 = \{o_2, o_3\}, L_4 = \{o_4\}\}$

Calculate dual LP (it has same objective value)

Maximize $y_1 + y_2 + y_3 + y_4$

subject to

$$
\begin{aligned}
y_1 + y_2 &\leq 3 && \text{(operator } o_1\text{)} \\
y_1 + y_3 &\leq 4 && \text{(operator } o_2\text{)} \\
y_2 + y_3 &\leq 5 && \text{(operator } o_3\text{)} \\
y_4 &\leq 0 && \text{(operator } o_4\text{)}
\end{aligned}
$$

$0 \leq y_1, y_2, y_3, y_4$

This is LP for optimal-cost partitioning!

SAS$^+$: Finite-domain variables

Representation language based on finite-domain variables rather than propositional variables

SAS$^+$ task is tuple $\Pi = (\mathcal{V}, O, s_I, s_\star, \text{cost})$ where

- $\mathcal{V}$ is set of variables; each variable V has **finite domain** $\text{Dom}(V)$
- s_I is **complete valuation** of variables defining initial state
- $s_\star$ is **partial valuation** of variables defining (set of) goal states
- O is set of operators; each operator given by (partial valuation) precondition $\text{pre}(o)$ and (partial valuation) effect $\text{eff}(o)$
- operator costs given by function $\text{cost} : O \to \mathbb{R}^{\geq 0}$

Transition Normal Form (TNF)

General format of SAS$^+$ problems in which:

- task $\Pi = (\mathcal{V}, O, s_I, s_\star, \text{cost})$
- $s_\star$ is complete state (i.e. unique goal state)
- $\text{vars}(\text{pre}(o)) = \text{vars}(\text{eff}(o))$ for every operator o

A SAS$^+$ task may be transformed into TNF in **linear time** (small overhead)

Practical transformation implemented in planners (e.g. FastDownward)

From now on, assume tasks in TNF as it makes presentation simpler

Transition systems and abstractions

Task $\Pi = (\mathcal{V}, O, s_I, s_\star, \text{cost})$ induces transition system $TS = (S, T, s_I, s_G)$:

- S is set of states
- s_I is initial state
- $s_G = \{s_\star\}$
- transitions (s, o, s') where o is applicable in s, and s' is resulting state

Abstraction $\alpha : S \to S^\alpha$ maps Π into **abstract** $\Pi^\alpha = (S^\alpha, T^\alpha, s_I^\alpha, s_G^\alpha)$:

- $S^\alpha = \{\alpha(s) : s \in S\}$
- $T^\alpha = \{(\alpha(s), o, \alpha(s')) : (s, o, s') \in T\}$
- $s_I^\alpha = \alpha(s_I)$
- $s_G^\alpha = \{\alpha(s) : s \in G\}$. If TNF, $G = \{s_\star\}$ and $s_G^\alpha = \{\alpha(s_\star)\}$

Heuristic $h^\alpha(s) =$ "cost of **optimal path** from $\alpha(s)$ to $\alpha(s_\star)$"

DTGs and SEQ heuristic

DTG for variable V is **directed graph** with vertices v for each value v of V, and edges (v, o, v') for each o such that $\text{pre}(o)[V] = v$ and $\text{eff}(o)[V] = v'$

State-equation (SEQ) heuristic $h^{\text{SEQ}}(s)$ for state s is defined by:

Decision variables: Count_o (#times o is applied)

$$\text{Minimize} \sum_o \text{cost}(o) \cdot \text{Count}_o$$

subject to

$$\sum_{v' \xrightarrow{o} v} \text{Count}_o - \sum_{v \xrightarrow{o} v'} \text{Count}_o = \Delta_s(V, v) \qquad \text{(for each } \langle V, v \rangle\text{)}$$

$$0 \leq \text{Count}_o \qquad \text{(for each } o\text{)}$$

where $\Delta_s(V, v) = [\![s_\star[V] = v]\!] - [\![s[V] = v]\!]$ (net change)

SEQ vs. OCP heuristics

Abstraction α is **atomic projection** if there is variable V such that $\alpha(s) = (V, s[V])$ (i.e. s is projected into V). Such α is denoted by α_V

Let Atom $= \{\alpha_V : V \in \mathcal{V}\}$ be all the atomic projections, and let $h^{\text{OCP}}_{\text{Atom}}$ denote the optimal cost-partitioning for the collection $\{h^{\alpha_V} : V\}$

Then, $h^{\text{OCP}}_{\text{Atom}}(s) \leq h^{\text{SEQ}}(s)$

We can close the gap by:

– removing **non-negativity constraints**, $0 \leq c_i(a)$, from OCP model (i.e., allow negative operator costs in the cost partitioning)

– removing "dead-end" states in the atomic transitions systems

Network flows

Consider transition system $TS = (S, T, s_I, \{s_\star\})$ with single goal state.
Following LP is standard formulation of **min-cost network flow problem:**

Decision variables: Count_t (#times transition t is traversed)

$$\text{Minimize} \sum_{o} \sum_{t \text{ has label } o} \mathsf{cost}(o) \cdot \mathsf{Count}_t$$

subject to

$$\sum_{t \in \mathsf{IN}(s)} \mathsf{Count}_t - \sum_{t \in \mathsf{OUT}(s)} \mathsf{Count}_t = \Delta(s') \quad \text{(for each } s' \in S\text{)}$$

$$0 \leq \mathsf{Count}_t \quad \text{(for each } t \in T\text{)}$$

where $\Delta(s') = [\![s' = s_\star]\!] - [\![s = s]\!]$ (net change)

For abstraction TS^α, $f^\alpha(s)$ denotes value of this LP for TS^α

Known: $h^{\mathsf{OCP}}_{\{f^{\alpha_V} : V\}}(s) \leq h^{\mathsf{SEQ}}(s)$ (but may bridge gap with simple transf.)

Strengthening of heuristics

LP-based heuristics can be strengthened by adding more constraints, often coming from different ideas or known heuristics

For example, we may combine **SEQ constraints** with **landmark constraints** to obtain an improved heuristic function

It can be shown:

$$h^{\mathsf{LP}}_{\mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_n} = h^{\mathsf{OCP}}_{\mathcal{H}}$$

where $\mathcal{H} = \{h^{\mathsf{LP}}_{\mathcal{C}_1}, h^{\mathsf{LP}}_{\mathcal{C}_2}, \ldots, h^{\mathsf{LP}}_{\mathcal{C}_n}\}$

For example, for landmark collection $\mathcal{L}$:

$$h^{\mathsf{LP}}_{\mathsf{SEQ} \cup \mathcal{L}} = h^{\mathsf{OCP}}_{\{h^{\mathsf{SEQ}}, h^{\mathsf{LP}}_{\mathcal{L}}\}} = h^{\mathsf{OCP}}_{\{h^{\mathsf{SEQ}}, h^{\mathsf{OCP}}_{\mathcal{L}}\}}$$

Potential heuristics: Motivation

Previous heuristics require solving an LP for each evaluation $h(s)$

It requires polynomial time, yet we'd like more efficient computation

Recall that after simple transformations $h^{\mathsf{OCP}}_{\mathsf{Atom}}(s)$ becomes $h^{\mathsf{SEQ}}(s)$. The former is an optimal cost partitioning for $\mathcal{H} = \{h^{\alpha_V} : V \in \mathcal{V}\}$

$h^{\alpha_V}(s)$ is cost of **optimal path** in DTG from V-value in s to V-value $s_\star$

New idea:

- Compute OCP for $\mathcal{H}$ for initial state s_I (yet any state will do)
- For each V, using the costs in OCP, **compute and store in table** $T[V, v]$ optimal costs to go from any V-value v to V-value in $s_\star[V]$
- During search, let $h(s) = \sum_{V \in \mathcal{V}} T[V, s[V]]$. Observe $h(s) \leq h^*(s)$ (why?)

Potential heuristics

A potential heuristic is a non-negative state function of the form

$$\varphi(s) \;=\; \sum_{f \in \mathcal{F}} w_f [\![s \vDash f]\!]$$

where $\mathcal{F}$ is **set of features** and w_f is weight assigned to feature f

We want to have such heuristic functions because:

– "Small" $\mathcal{F}$ and fast-evaluation of $[\![s \vDash f]\!]$ yields fast computation of $\varphi(s)$

– Potential heuristics may be quite informative

– How do we guarantee admissibility / consistency of φ?

Use LP to define admissible, consistent, and informative heuristics!

Fact-based features

Consider task $\Pi = (\mathcal{V}, O, s_I, s_\star, \text{cost})$ in TNF and let $TS = (S, T, s_I, \{s_\star\})$ be its transition system

A fact is a pair $\langle V, v \rangle$. It holds in state s iff $s[V] = v$.

Facts may be used to construct features (n is number of facts):

- **One-dimensional** feature is fact (there are $\mathcal{O}(n)$ such features)
- **Two-dimensional** feature is conjunction of two facts (incl. 1d) ($\mathcal{O}(n^2)$)
- **Higher-dimensional** feature is conjunction of three or more facts ($\mathcal{O}(n^d)$)

Goal-aware and consistent potential heuristics

Potential heuristic φ is goal-aware and consistent (and thus admissible) iff

$$\begin{aligned} \varphi(s_\star) &\leq 0 \\ \varphi(s) - \varphi(s') &\leq \mathsf{cost}(o) \qquad \text{(for each } s \xrightarrow{o} s' \text{ in } T\text{)} \end{aligned}$$

Substituting $\varphi(s)$ by its definition:

$$\begin{aligned} \textstyle\sum_{f \in \mathcal{F}} w_f [\![s_\star \models f]\!] &\leq 0 \\ \textstyle\sum_{f \in \mathcal{F}} w_f \big([\![s \models f]\!] - [\![s' \models f]\!] \big) &\leq \mathsf{cost}(o) \qquad \text{(for each } s \xrightarrow{o} s' \text{ in } T\text{)} \end{aligned}$$

This LP has **exponential number of constraints:** one for each transition

We'll circumvent this obstacle for one- and two-dimensional features!

One-dimensional features

Let us consider a transition $s \xrightarrow{o} s'$ in T:

- Let $\Delta_o(f, s) = [\![s \vDash f]\!] - [\![s' \vDash f]\!]$
- Each feature is of format $f = \langle V, v \rangle$, let $\text{var}(f) = V$
- (For operator o) partition $\mathcal{F}$ into $\mathcal{F}^{\text{irr}}$=$\{f : \text{var}(f) \notin \text{vars}(o)\}$ and $\mathcal{F}^{\text{ind}} = \mathcal{F} \backslash \mathcal{F}^{\text{irr}}$:

$$\sum_{f \in \mathcal{F}} w_f \, \Delta_o(f, s) \;=\; \sum_{f \in \mathcal{F}^{\text{irr}}} w_f \, \Delta_o(f, s) + \sum_{f \in \mathcal{F}^{\text{ind}}} w_f \, \Delta_o(f, s)$$

- For feature f in $\mathcal{F}^{\text{irr}}$: $\Delta_o(f, s) = 0$ since o doesn't change its value
- For feature f in $\mathcal{F}^{\text{ind}}$: $[\![s \vDash f]\!] = [\![\text{pre}(o) \vDash f]\!]$ and $[\![s' \vDash f]\!] = [\![\text{eff}(o) \vDash f]\!]$
- Thus, $\Delta_o(f, s) = [\![s \vDash f]\!] - [\![s' \vDash f]\!] = [\![\text{pre}(o) \vDash f]\!] - [\![\text{eff}(o) \vDash f]\!] = \Delta_o(f)$

For transition $s \xrightarrow{o} s'$,

$$\sum_{f \in \mathcal{F}} w_f \, \Delta_o(f, s) \leq \text{cost}(o) \quad \textbf{is equivalent to} \quad \sum_{f \in \mathcal{F}^{\text{ind}}} w_f \, \Delta_o(f) \leq \text{cost}(o)$$

LP for one-dimensional potential heuristics

Decision variables: w_f (weight of feature f)

[*your choice of linear objective function*]

subject to

$$\sum_{f \in \mathcal{F}} w_f \, [\![s_\star \models f]\!] \;\leq\; 0$$

$$\sum_{f \in \mathcal{F}^{\text{ind}}} w_f \, \Delta_o(f) \;\leq\; \text{cost}(o) \qquad \text{(for each operator } o\text{)}$$

where $\Delta_o(f) \in \{-1, 0, 1\}$ and $[\![s_\star \models f]\!] \in \{0, 1\}$ are constants

- $\mathcal{F}$ is **any subset** of one-dimensional features **(includes atomic proj.)**
- LP size $= |\mathcal{F}|$ decision variables, and $1 + |O|$ linear constraints

Choice of objective function

Room for different ideas:

– Max value at initial state s_I: Maximize $\sum_{f \in \mathcal{F}} w_f \, [\![s_I \models f]\!]$

– Max average value in sample M of states of size k:

$$\text{Maximize } \frac{1}{k} \sum_{s \in M} \sum_{f \in \mathcal{F}} w_f \, [\![s \models f]\!]$$

– Max min value in sample M of states:

$$\text{Maximize } z$$

subject to

$$z \;\leq\; \sum_{f \in \mathcal{F}} w_f \, [\![s \models f]\!] \qquad \text{(for each } s \in M\text{)}$$

– . . . others . . .

LP for two-dimensional potential heuristics

Decision variables: w_f (weight of feature f), z^o_V **(new auxiliary vars)**

[*your choice of linear objective function*]

subject to

$$\sum_{f \in \mathcal{F}} w_f \, [\![s_\star \models f]\!] \;\leq\; 0$$

$$\Delta_o + \sum_{V \in \mathcal{V}_{\overline{o}}} z^o_V \;\leq\; \mathsf{cost}(o) \qquad \text{(for each operator } o\text{)}$$

$$\sum_{f \in \mathcal{F}^{\mathsf{ctx}}, f = f_o \wedge \langle V, v \rangle} w_f \Delta_o(f_o) \;\leq\; z^o_V \qquad \text{(for each } o,\ V \in \mathcal{V}_{\overline{o}},\ v \in \mathsf{Dom}(V)\text{)}$$

where $\mathcal{V}_{\overline{o}} = \mathcal{V} \setminus \mathsf{vars}(o)$, $\Delta_o = \sum_{f \in \mathcal{F}^{\mathsf{ind}}} w_f \Delta_o(f)$, $\mathcal{F} = \mathcal{F}^{\mathsf{irr}} \cup \mathcal{F}^{\mathsf{ind}} \cup \mathcal{F}^{\mathsf{ctx}}$

- $\mathcal{F}$ is **any subset** of two-dimensional features
- LP size $= |\mathcal{F}| + |\mathcal{V}| \times |O|$ decision variables, and **at most** $1 + |O|(1 + |\mathcal{V}|d)$ linear constraints where d bounds domain size of variables

Higher-dimensional features

Intractable: reduction of non-3-colorability into testing if given φ is consistent

Let $G = (V, E)$ be undirected graph: the one we'd like to test for non 3-colorability

Need construct Π and φ in p-time such that φ is consistent iff G is non 3-colorable

---

For the task $\Pi = (\mathcal{V}, O, s_I, s_\star)$ in TNF:

- $|V + 1|$ variables: one C_v for color of vertex v (rgb), and one master M (binary)
- For vertex v and $c \neq c'$, there is operator $chg(v, c, c')$ of **zero cost** to change C_v from c to c' when $M = 0$ (effects include $M = 0$ as well)
- For M, there is operator o_M of **zero cost** to change M from 0 to 1
- $s_I[C_v] = s_\star[C_v] = red$ for all vertices v, $s_I[M] = 0$, and $s_\star[M] = 1$

For potential $\varphi(s)$ over 3-dimensional features f, **all weights are zero except**:

- $w_f = -1$ if vars(f)=$\{M, C_u, C_v\}$, $\{u, v\}$ is edge, $f[M]$=1, and $f[C_u] \neq f[C_v]$
- $w_{f_M} = |E| - 1$ for feature $f_M = \langle M, 1 \rangle$ of dimension 1

Claim: φ is consistent iff G is non 3-colorable (e.g. no efficiently constructible LP)

Analysis of reduction

Analysis of values of φ at states s:

- If $s[M] = 0$, then $\varphi(s) = 0$ since $w_f = 0$ when $f[M] = 0$
- If $s[M] = 1$, then $\varphi(s) \geq -1$ since $w_{f_M} = |E| - 1$ and $w_f = -1$ for features f with $\text{vars}(f) = \{M, C_u, C_v\}$ such that $\{u, v\}$ is edge, $f[M] = 1$ and $f[C_u] \neq f[C_v]$ (there are at most $|E|$ such features f with $s \models f$)
- $\varphi(s) = -1$ iff $s[M]=1$ and s is 3-coloring: f_M contributes $|E| - 1$, and each edge $\{u, v\}$ contributes -1 via $f = \langle M, 1\rangle \wedge \langle C_u, s[C_u]\rangle \wedge \langle C_v, s[C_v]\rangle$

G is 3-colorable $\Leftrightarrow$ there is operator sequence that achieves 3-coloring

$\Leftrightarrow$ there is $s \xrightarrow{o} s'$ where s' is 3-coloring and $s'[M]=1$

$\Leftrightarrow$ there is $s \xrightarrow{o} s'$ where $\varphi(s) = 0$ and $\varphi(s') = -1$

$\Leftrightarrow$ **φ isn't consistent** (since all operators have zero cost)

Theorem: Checking whether given φ is consistent is **coNP-Complete**

Higher-dimensional features: Parametrized tractability

Let $\mathcal{F}$ be a set of features, each of **arbitrary dimension**

We want LP for constructing goal-aware and consistent $\mathcal{F}$-potentials

Theorem: There is set $\mathcal{C}$ of linear constraints that **characterize** the goal-aware and consistent $\mathcal{F}$-potential heuristics on task Π where

– number of decision variables is $\mathcal{O}(|O|(|\mathcal{F}| + nd^{w^*}))$

– number of constraints is $\mathcal{O}(|O|nd^{1+w^*})$

where d bounds domain size of variables in Π, n is number of variables in Π, and w^* is maximum **treewidth of context-dependency graphs**

Computing linear constraints that characterize goal-aware and consistent $\mathcal{F}$-potentials is **fixed-parameter tractable** with parameter $\max\{w^*, d\}$

Context-dependency graph and treewidth

Number of variables and constraints in LP for arbitrary feature set $\mathcal{F}$ is **exponential** in maximum **treewidth** of context-dependency (CD) graphs

CD graph $G(\Pi, \mathcal{F}, o)$ for task Π, feature set $\mathcal{F}$, and operator o has:

- one vertex for each variable V in Π
- one (undirected) edge $e = \{V, V'\}$, $V \neq V'$, iff there is f in $\mathcal{F}$ such that $\text{vars}(f) \cap \text{vars}(o) \neq \emptyset$ and $e \subseteq \text{vars}(f) \setminus \text{vars}(o)$

Treewidth of (undirected) graph G:

- parameter that measures how "complex" are cycles in G (larger means more complex)
- treewidth of tree is 1, and treewidth of clique K_n is $n - 1$
- often appears in analysis of combinatorial algorithms

Wrap up

- Linear programming is a powerful practical and theoretical tool for analysis, design, and implementation of heuristics for planning
- Most powerful and advanced state-of-the-art heuristics are formulated and computed using LP (either during preprocessing before search starts or during search)
- LP-based heuristics may go beyond delete-relaxation heuristics (like SEQ)
- Potential heuristics may be even more powerful and they offer interesting computational tradeoffs
- LP should be considered a **declarative language** for heuristic functions rather than just a computational model

References

Relevant references in chronological order:

1. M. van den Briel, J. Benton, S. Kambhampati and T. Vossen. *An LP-based heuristic for optimal planning.* CP 2007.
2. B. Bonet and M. Helmert. *Strengthening Landmark Heuristics via Hitting Sets*. ECAI 2010.
3. M. Katz and C. Domshlak. *Optimal admissible composition of abstraction heuristics.* AIJ 2010.
4. H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning.* Morgan & Claypool Publishers. 2013.
5. B. Bonet. *An admissible heuristic for SAS+ planning obtained from the state equation.* IJCAI 2013.
6. B. Bonet and M. van den Briel, *Flow-based heuristics for optimal planning: Landmarks and merges.* ICAPS 2014.

References

Relevant references in chronological order:

7. F. Pommerening, G. Röger, M. Helmert and B. Bonet. *LP-based Heuristics for Cost-optimal Planning*. ICAPS 2014.

8. F. Pommerening and M. Helmert. *A normal form for classical planning tasks.* ICAPS 2015.

9. F. Pommerening, M. Helmert, G. Röger and J. Seipp. *From non-negative to general operator cost partitioning.* AAAI 2015.

10. J. Seipp, F. Pommerening and M. Helmert. *New optimization functions for potential heuristics.* ICAPS 2015.

11. F. Pommerening, M. Helmert and B. Bonet. *Abstraction Heuristics, Cost Partitioning and Network Flows*. ICAPS 2017.

12. F. Pommerening, M. Helmert and B. Bonet. *Higher-Dimensional Potential Heuristics for Optimal Classical Planning*. AAAI 2017.