Planning and Optimization

E8. Flow Heuristic

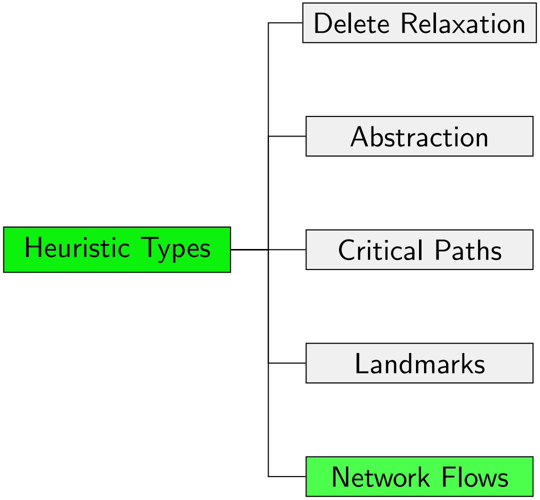
Malte Helmert and Gabriele Röger

Universität Basel

November 29, 2017

Introduction

Content of this Course: Heuristic Types



Reminder: SAS$^+$ Planning Tasks

For a SAS$^+$ planning task $\Pi = \langle V, I, O, \gamma \rangle$:

- V is a set of finite-domain state variables,
- Each atom has the form $v = d$ with $v \in V, d \in \mathrm{dom}(v)$.
- Operator preconditions and the goal formula γ are conjunctions of atoms.
- Operator effects are conjunctions of atomic effects, i.e., they have the form $v_1 := d_1 \wedge \cdots \wedge v_n := d_n$.

Example Task (1)

- One package, two trucks, two locations
- Variables:
  - $\mathit{pos\text{-}p}$ with $\mathrm{dom}(\mathit{pos\text{-}p}) = \{loc_1, loc_2, t_1, t_2\}$
  - $\mathit{pos\text{-}t\text{-}i}$ with $\mathrm{dom}(\mathit{pos\text{-}t\text{-}i}) = \{loc_1, loc_2\}$ for $i \in \{1, 2\}$
- The package is at location 1 and the trucks at location 2,
  - $I = \{\mathit{pos\text{-}p} \mapsto loc_1, \mathit{pos\text{-}t\text{-}1} \mapsto loc_2, \mathit{pos\text{-}t\text{-}2} \mapsto loc_2)$
- The goal is to have the package at location 2 and truck 1 at location 1.
  - $\gamma = (\mathit{pos\text{-}p} = loc_2) \wedge (\mathit{pos\text{-}t\text{-}1} = loc_1)$

Example Task (2)

- Operators: for $i, j, k \in \{1, 2\}$:

$$
\begin{aligned}
\mathit{load}(t_i, \mathit{loc}_j) &= \langle \mathit{pos\text{-}t\text{-}i} = \mathit{loc}_j \wedge \mathit{pos\text{-}p} = \mathit{loc}_j, \\
&\quad\ \mathit{pos\text{-}p} := t_i, 1\rangle \\
\mathit{unload}(t_i, \mathit{loc}_j) &= \langle \mathit{pos\text{-}t\text{-}i} = \mathit{loc}_j \wedge \mathit{pos\text{-}p} = t_i, \\
&\quad\ \mathit{pos\text{-}p} := \mathit{loc}_j, 1\rangle \\
\mathit{drive}(t_i, \mathit{loc}_j, \mathit{loc}_k) &= \langle \mathit{pos\text{-}t\text{-}i} = \mathit{loc}_j, \\
&\quad\ \mathit{pos\text{-}t\text{-}i} := \mathit{loc}_k, 1\rangle
\end{aligned}
$$

Example Task: Observations

Consider some atoms of the example task:

- $\mathit{pos\text{-}p} = \mathit{loc}_1$ initially true and must be false in the goal
  ▷ at location 1 the package must be loaded one time more often than unloaded.
- $\mathit{pos\text{-}p} = \mathit{loc}_2$ initially false and must be true in the goal
  ▷ at location 2 the package must be unloaded one time more often than loaded.
- $\mathit{pos\text{-}p} = t_1$ initially false and must be false in the goal
  ▷ same number of load and unload actions for truck 1.

Can we derive a heuristic from this kind of information?

Flow Heuristic

Example: Flow Constraints

Let π be some arbitrary plan for the example task and let $\#o$ denote the number of occurrences of operator o in π. Then the following holds:

- $pos\text{-}p = loc_1$ initially true and must be false in the goal
  ▷ at location 1 the package must be loaded one time more often than unloaded.
  $\#load(t_1, loc_1) + \#load(t_2, loc_1) =$
  $1 + \#unload(t_1, loc_1) + \#unload(t_2, loc_1)$
- $pos\text{-}p = t_1$ initially false and must be false in the goal
  ▷ same number of load and unload actions for truck 1.
  $\#unload(t_1, loc_1) + \#unload(t_1, loc_2) =$
  $\#load(t_1, loc_1) + \#load(t_1, loc_2)$

Network Flow Heuristics: General Idea

- Formulate flow constraints for each atom.
- These are satisfied by every plan of the task.
- The cost of a plan is $\sum_{o \in O} cost(o) \# o$
- The objective value of an integer program that minimizes this cost subject to the flow constraints is a lower bound on the plan cost (i.e., an admissible heuristic estimate).
- As solving the IP is NP-hard, we solve the LP relaxation instead.

How do we get the flow constraints?

How to Derive Flow Constraints?

- The constraints formulate how often an atom can be produced or consumed.
- "Produced" (resp. "consumed") means that the atom is false (resp. true) before an operator application and true (resp. false) in the successor state.
- For general SAS$^+$ operators, this depends on the state where the operator is applied: effect $v := d$ only produces $v = d$ if the operator is applied in a state s with $s(v) \neq d$.
- For general SAS$^+$ tasks, the goal does not have to specify a value for every variable.
- All this makes the definition of flow constraints somewhat involved and in general such constraints are inequalitites.

Good news: easy for tasks in transition normal form

Reminder: Transition Normal Form

Definition (Transition Normal Form)

A SAS^+ planning task $\Pi = \langle V, I, O, \gamma \rangle$ is in transition normal form (TNF) if

- for all $o \in O$, $vars(pre(o)) = vars(eff(o))$, and
- $vars(\gamma) = V$.

In words, an operator in TNF must mention the same variables in the precondition and effect, and a goal in TNF must mention all variables (= specify exactly one goal state).

TNF for Example Task (1)

The example task is not in transition normal form:

- Load and unload operators have preconditions on the position of some truck but no effect on this variable.
- The goal does not specify a value for variable *pos-t-2*.

TNF for Example Task (2)

Operators in transition normal form: for $i, j, k \in \{1, 2\}$:

$$\begin{aligned}
\mathit{load}(t_i, \mathit{loc}_j) &= \langle \mathit{pos\text{-}t\text{-}i} = \mathit{loc}_j \land \mathit{pos\text{-}p} = \mathit{loc}_j, \\
&\qquad \mathit{pos\text{-}p} := t_i \land \mathit{pos\text{-}t\text{-}i} := \mathit{loc}_j, 1\rangle \\
\mathit{unload}(t_i, \mathit{loc}_j) &= \langle \mathit{pos\text{-}t\text{-}i} = \mathit{loc}_j \land \mathit{pos\text{-}p} = t_i, \\
&\qquad \mathit{pos\text{-}p} := \mathit{loc}_j \land \mathit{pos\text{-}t\text{-}i} := \mathit{loc}_j, 1\rangle \\
\mathit{drive}(t_i, \mathit{loc}_j, \mathit{loc}_k) &= \langle \mathit{pos\text{-}t\text{-}i} = \mathit{loc}_j, \\
&\qquad \mathit{pos\text{-}t\text{-}i} := \mathit{loc}_k, 1\rangle
\end{aligned}$$

TNF for Example Task (3)

To bring the goal in normal form,

- add an additional value **u** to dom(*pos-t-2*)
- add zero-cost operators
  $o_1 = \langle \textit{pos-t-2} = \textit{loc}_1, \textit{pos-t-2} := \mathbf{u}, 0\rangle$ and
  $o_2 = \langle \textit{pos-t-2} = \textit{loc}_2, \textit{pos-t-2} := \mathbf{u}, 0\rangle$
- Add $\textit{pos-t-2} = \mathbf{u}$ to the goal:
  $\gamma = (\textit{pos-p} = \textit{loc}_2) \wedge (\textit{pos-t-1} = \textit{loc}_1) \wedge (\textit{pos-t-2} = \mathbf{u})$

Notation

- In SAS$^+$ tasks, states are variable assignments, conditions are conjunctions over atoms, and effects are conjunctions of atomic effects.
- In the following, we use a unifying notation to express that an atom is true in a state/entailed by a condition/ made true by an effect.
- For state s, we write $(v = d) \in s$ to express that $s(v) = d$.
- For a conjunction of atoms φ, we write $(v = d) \in \varphi$ to express that φ has a conjunct $v = d$ (or alternatively $\varphi \models v = d$).
- For effect e, we write $(v = d) \in e$ to express that e contains the atomic effect $v := d$.

Flow Constraints (1)

A flow constraint for an atom relates how often it can be produced to how often it can be consumed.

Let o be an operator in transition normal form. Then:

- o produces atom a iff $a \in \mathit{eff}(o)$ and $a \notin \mathit{pre}(o)$.
- o consumes atom a iff $a \in \mathit{pre}(o)$ and $a \notin \mathit{eff}(o)$.
- Otherwise o is neutral wrt. atom a.

$\rightsquigarrow$ State-independent

Flow Constraints (2)

A flow constraint for an atom relates how often it can be produced to how often it can be consumed.

The constraint depends on the current state s and the goal γ.
If γ mentions all variables (as in TNF), the following holds:

- If $a \in s$ and $a \in \gamma$ then atom a must be equally often produced and consumed.
- Analogously for $a \notin s$ and $a \notin \gamma$.
- If $a \in s$ and $a \notin \gamma$ then a must be consumed one time more often than it is produced.
- If $a \notin s$ and $a \in \gamma$ then a must be produced one time more often than it is consumed.

Iverson Bracket

The dependency on the current state and the goal can concisely be expressed with Iverson brackets:

Definition (Iverson Bracket)

Let P be a logical proposition ($=$ some statement that can be evaluated to true or false). Then

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{if } P \text{ is false.} \end{cases}$$

Example: $[2 \neq 3] = 1$

Flow Constraints (3)

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

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

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

Flow Heuristic

Definition (Flow Heuristic)

Let $\Pi = \langle V, I, O, \gamma \rangle$ be a SAS$^+$ task in transition normal form and let $A = \{(v = d) \mid v \in V, d \in dom(v)\}$ be the set of atoms of Π.

The flow heuristic $h^{\text{flow}}(s)$ is the objective value of the following LP or ∞ if the LP is infeasible:

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

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

$$\text{Count}_o \geq 0 \quad \text{for all } o \in O$$

Flow Heuristic on Example Task

$\rightsquigarrow$ Blackboard

Flow Heuristic: Properties (1)

Theorem

The flow heuristic h^{flow} is goal-aware, safe, consistent and admissible.

Proof.

We prove goal-awareness and consistency, the other properties follow from these two.

Goal-awareness: If $s \models \gamma$ then $\text{Count}_o = 0$ for all $o \in O$ is feasible and the objective function has value 0. As $\text{Count}_o \geq 0$ for all variables and operator costs are nonnegative, the objective value cannot be smaller. ...

Flow Heuristic: Properties (2)

Proof (continued).

Consistency: Let o be an operator that is applicable in state s and let $s' = s[\![o]\!]$. Consider an optimal feasible vector $\mathbf{y}'$ for the LP for s' and let $y_{o'}$ denote the value of $\text{Count}_{o'}$ in this vector. Let $\mathbf{y}$ be the vector that assigns Count_o the value $y_o + 1$ and all other variables $\text{Count}_{o'}$ $(o' \neq o)$ the value $y_{o'}$. We show that $\mathbf{y}$ is feasible for the LP for s.

Let $a = (v = d)$ be an atom. The flow constraint for a in state s is

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

We consider how the flow constraint for a is affected by a change from s' to s and from $\mathbf{y}'$ to $\mathbf{y}$. ...

Flow Heuristic: Properties (3)

Proof (continued).

If $v \notin vars(pre(o))$, the constraint is not affected and stays satisfied as it is satisfied by $\mathbf{y}'$. Otherwise, we distinguish four cases:

- $a \in pre(o), a \notin eff(o)$: Then $a \in s$ and $a \notin s'$, increasing the left-hand side by one. Count_o only occurs on the right-hand side and increases by one, so the change is balanced.
- $a \notin pre(o), a \in eff(o)$: Then $a \notin s$ and $a \in s'$, decreasing the left-hand side by one. Count_o only occurs on the left-hand side and increases by one, so the change is balanced.
- $a \in pre(o), a \in eff(o)$: Then $a \in s$ and $a \in s'$ and Count_o occurs on both sides, so the equation stays balanced.
- $a \notin pre(o), a \notin eff(o)$: Then $a \notin s$ and $a \notin s'$ and Count_o does not occur on either side of the equation.

...

Flow Heuristic: Properties (4)

Proof (continued).

As $\mathbf{y} \geq \mathbf{y}' \geq \mathbf{0}$, also the constraints that require the LP variables to be non-negative are satisfied.

The value of the objective function with $\mathbf{y}$ is $h^{\text{flow}}(s') + cost(o)$. Since $\mathbf{y}$ is feasible for the LP for state s, this is an upper bound on $h^{\text{flow}}(s)$, so in total $h^{\text{flow}}(s) \leq h^{\text{flow}}(s') + cost(o)$. □

Summary

Summary

- A flow constraint for an atom describes how the number of producing operator applications is linked to the number of consuming operator applications.
- The flow heuristic computes a lower bound on the cost of each operator sequence that satisfies these constraints for all atoms.
- The heuristic only considers the number of occurrences of each operator, but ignores their order.

Literature

Literature (1)

References on the flow heuristic:

Menkes van den Briel, J Benton, and Rao Kambhampati.
An LP-based Heuristic for Optimal Planning.
Proc. CP 2007, pp. 651–665, 2007.
Introduces the flow heuristic.

Blai Bonet.
An Admissible Heuristic for SAS+ Planning Obtained from the State Equation.
Proc. IJCAI 2013, pp. 2268–2274, 2013.
Rediscovery of flow heuristic plus some improvements.

Literature (2)

Blai Bonet and Menkes van den Briel.
Flow-based Heuristics for Optimal Planning: Landmarks and Merges.
Proc. ICAPS 2014, pp. 47–55, 2014.
More on improvements.

Florian Pommerening and Malte Helmert.
A Normal Form for Classical Planning Tasks.
Proc. ICAPS 2015, pp. 188–192, 2015.
Formulation for transition normal form.