Planning for Artificial Intelligence

Lukáš Chrpa

LP-based Heuristics

Linear Programming

Linear Program

- A **Linear Program (LP)** consists of
  - A finite set of **real-valued variables** V
  - A finite set of **linear inequalities** over variables V
  - An **objective function** being a linear combination over V, which should be either **maximized** or **minimized**
- An **Integer Program (IP)** is the same except **integer-valued variables**

Complexity of LP

- An LP problem can be solved in **polynomial** time
- Solving IP is **NP-complete**

- Approximate IP solutions by corresponding LP ones (LP relaxation)

LP for Shortest Path in State Space

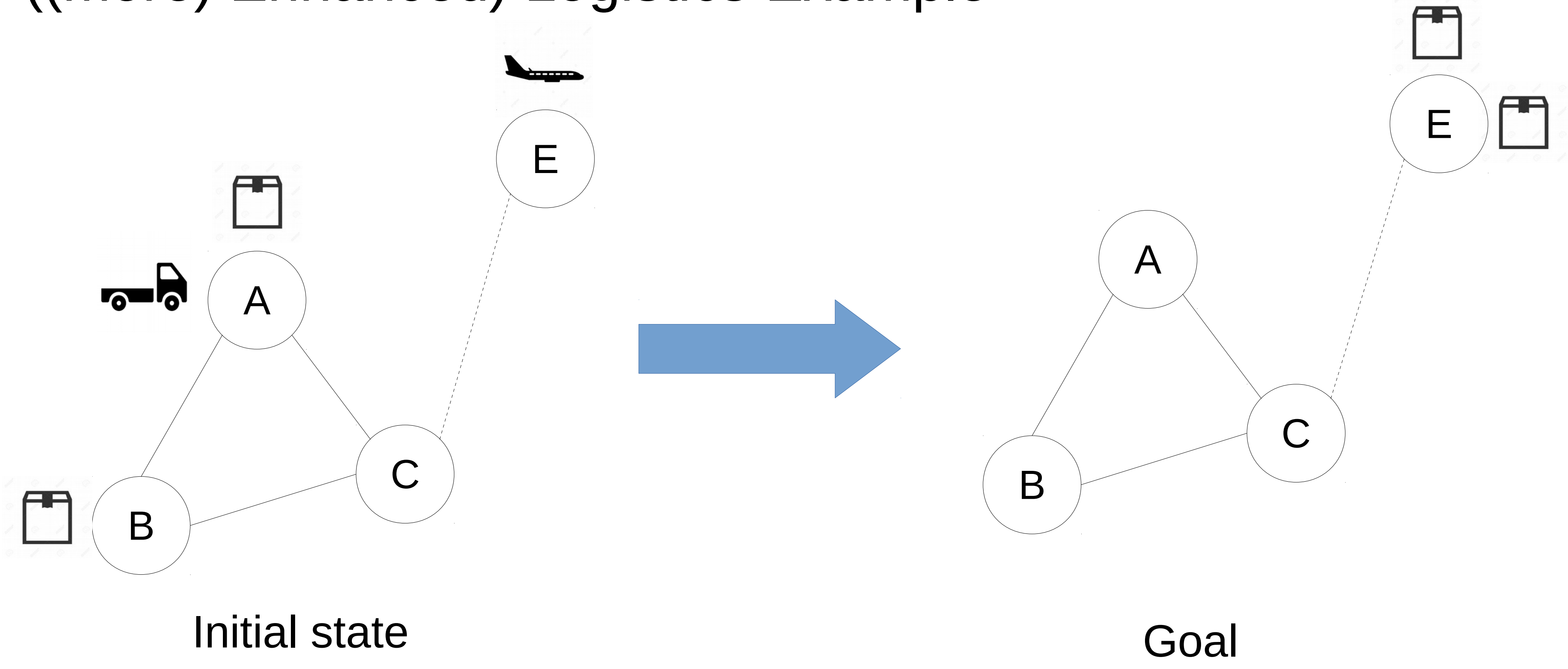
- Variables
  - $Dist_s$ for each state s
  - *GoalDist*
- **Maximize** *GoalDist*
- Subject to
  - $Dist_I = 0$
  - $Dist_{s'} \leq Dist_s + c(a)$ for each $\gamma(s,a)=s'$
  - $GoalDist \leq Dist_{sG}$ for each goal state s_G

Cost Partitioning

Observations

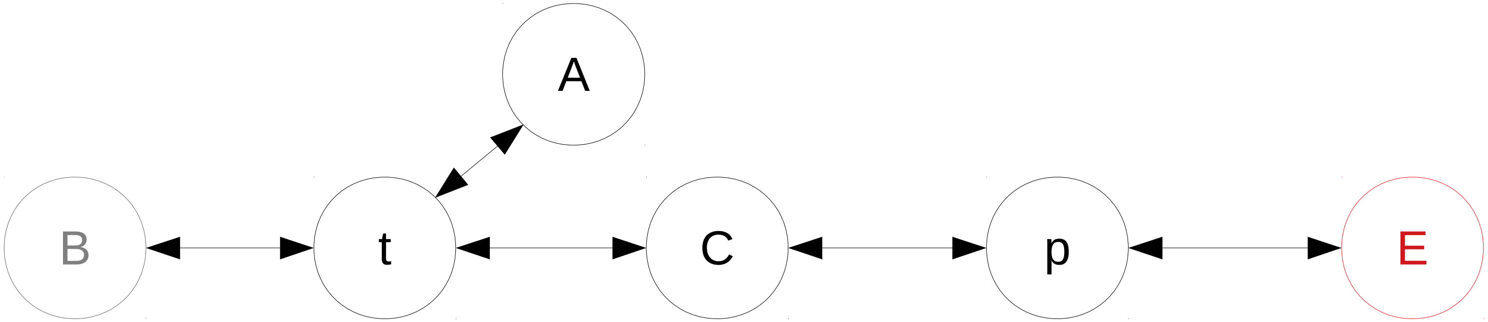
- Enumerating the state space is not a feasible option
- One option is to somehow split a problem into small subproblems
  - **cost partitioning** (action cost is divided into these subproblems)
- How ?
  - by **abstractions** (will be taught in a few weeks …)
  - DTG is a sort of abstraction

((more) Enhanced) Logistics Example

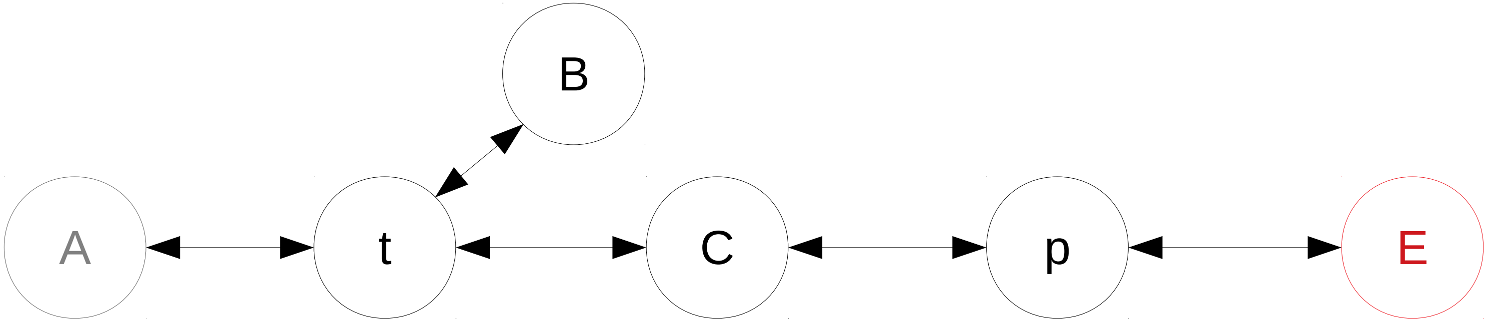


((more) Enhanced) Logistics Example – DTGs

DTG_{p1}



DTG_{p2}



Cost Partitioning

- Create **copies** $\Pi_1,\ldots,\Pi_n$ of a planning task Π
- Each copy has a different **action cost** function c_i (≥ 0)
- For each action, $c_1(a)+\ldots+c_n(a) \leq c(a)$

- We can derive that

$$h_1^*+\ldots+h_n^* \leq h^*$$

Optimal Cost Partitioning with LP

- Use variables for costs of each action in each task copy
- Express heuristic values with linear constraints (inequalities)
- Maximize the sum of these heuristic values (subject to the constraints)

LP for Optimal Cost Partitioning for Abstractions

- Variables
  - For each abstraction α
    - $Dist^{\alpha}_{s}$ for each state s (in α)
    - $c^{\alpha}(a)$ for each action a
    - $GoalDist^{\alpha}$
- **Maximize** $\sum_{\alpha} GoalDist^{\alpha}$

LP for Optimal Cost Partitioning for Abstractions cont.

- Subject to
  - for each action a
    - $c^{\alpha}(a) \geq 0$ (for each α)
    - $\sum_{\alpha} c^{\alpha}(a) \leq c(a)$
  - for each abstraction α
    - $Dist^{\alpha}_{I} = 0$
    - $Dist^{\alpha}_{s'} \leq Dist^{\alpha}_{s} + c^{\alpha}(a)$ for each $\gamma^{\alpha}(s,a)=s'$
    - $GoalDist^{\alpha} \leq Dist^{\alpha}_{sG}$ for each abstract goal state s_G

Operator Counting

Operator (action) Counting

- Reasoning about (solution) plans for deriving heuristics
- **Linear constraints** over variables representing the number of **action (operator) occurrence** in every plan
- For example
  - $Y_{a1} + Y_{a2} + Y_{a3} \geq 1$ – must apply a_1, a_2 or a_3 at least once (recall disjunctive action landmarks)
    - A package has to be loaded to some truck
  - $Y_{a4} - Y_{a5} \leq 0$ – cannot use a_4 more often than a_5
    - A package cannot be unloaded more often than loaded

Operator-counting Heuristics

- Y_a represents the **number of occurrences** of a
- Hence, for each action a $Y_a \geq 0$
- **Minimize** $\sum_a Y_a c(a)$
  - this is also the value of the heuristic
- Additional constraints (inequalities) over Y_a variables can be considered
  - e.g. those in the previous slide

Properties of Operator-counting Heuristics

- Operator-counting heuristics are **admissible**

- Operator-counting heuristics can be calculated in **polynomial time** (solving LP)

- Adding **more constraints** makes operator-counting heuristics **more informed**

State-Equation Heuristic (SEQ)

- Facts (variable assignments) can be **produced** and **consumed** by actions
- Number of producing and consuming actions must **be balanced** for each fact (depends on the **current state** and the **goal**)
  - e.g in the Logistic example, let pkg=B be true in the current state and pkg=E be the goal
    - pkg=B has to be **consumed** (i.e., there has to be one more consumer of pkg=B)
    - pkg=E has to be **produced** (i.e., there has to be one more producer of pkg=E)
    - for pkg=X ($X \neq B,E$)**,** there has to be the **same number** of producers and consumers

State-Equation Heuristic (SEQ)

- The set of actions **producing** *f*, $prod(f)=\{a \mid f \in eff(a), f \notin pre(a)\}$
- The set of actions **consuming** *f*, $cons(f)=\{a \mid vars(f) \cap vars(pre(a)) \cap vars(eff(a)) \neq \emptyset, f \in pre(a), f \notin eff(a)\}$
- For each fact (variable assignment) over variables mentioned in G:

$$G(f) - s(f) \leq \Sigma_{a \in prod(f)} Y_a - \Sigma_{a \in cons(f)} Y_a$$

- Note that we assume that variables mentioned in the effects of all actions are also mentioned in the preconditions of these actions
  - can be adapted for cases in which it doesn't hold
- Note that s(f)=1 if f is true in s, otherwise s(f)=0, and G(f)=1 if f is part of the goal, otherwise G(f)=0

Potential Heuristics

Potential Heuristics

- A **state feature** is a function $f{:}S \rightarrow \mathbb{R}$
- A **potential heuristic** for a set of state features $f_1, f_2, \dots, f_n$ is a heuristic function h defined as a **linear combination** of the state features

$$h(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

with real-valued weights (**potentials**) w_i

Obtaining Ingredients for Potential Heuristics

- Computing **state features** should be fast (i.e. in **constant time**)
- Determining **potentials**
  - LP to rescue !
  - Ideally, **potentials are computed only once** (not in every visited state as e.g. SEQ)
- Potential heuristic should be
  - admissible
  - consistent

Atomic Potential Heuristics

- An **atomic state feature** tests if an atom (fact) is true in a given state
- Let X=y be an atom (fact) and s be a state. The **atomic feature** $f_{X=y}(s)$ is defined as:

$$f_{X=y}(s) = 1, \text{ if } (X=y) \in s$$

$$f_{X=y}(s) = 0, \text{ otherwise}$$

- We take into account **all** the atomic features
- Complexity ?

Atomic Potential Heuristics

- An **atomic state feature** tests if an atom (fact) is true in a given state
- Let X=y be an atom (fact) and s be a state. The **atomic feature** $f_{X=y}(s)$ is defined as:

$$f_{X=y}(s) = 1, \text{ if } X=y \in s$$
$$f_{X=y}(s) = 0, \text{ otherwise}$$

- We take into account **all** the atomic features
- Complexity ?
  - Constant

Computing Potentials

- Constraints on potentials characterize **admissible** and **consistent** atomic potential heuristics
- **Goal awareness** (for each goal state s_G)

$$\sum_{f \in s_G} w_f = 0$$

- **Consistency** (for each action a)

$$\sum_{f \in pre(a)} w_f - \sum_{f \in eff(a)} w_f \leq c(a)$$

- Again, we assume that variables mentioned in preconditions and effects of all actions are the same

Are we missing something ?

Are we missing something ? Objective function

- Well informed heuristics should be close to the perfect one
- Some examples of objective functions
  - maximize **heuristic value** of **the initial state**
  - maximize **average heuristic value** of **all states**
  - maximize **average heuristic value** of **sample states**
  - ….

A Little Bit of Theory

- Let $h^{maxpot}(s)$ represent the maximum value across all admissible and consistent atomic potential heuristics in s
- Let $h^{SEQ}(s)$ represent the state-equation heuristic value in s (in literature the SEQ heuristic is also called the *flow* heuristic)
- Let $h^{gOCP}(s)$ represent the optimal general cost partitioning (omits non-negativity cost constraints) of atomic projections (similar to DTGs)
- **Theorem**

$$h^{maxpot}(s) = h^{SEQ}(s) = h^{gOCP}(s)$$