AI Planning for Autonomy

5. Critical Path Heuristics

It's a Long Way to the Goal, But How Long Exactly?
Part I: *Following the Most Critical Sub-Goals*

Nir Lipovetzky

THE UNIVERSITY OF MELBOURNE

Winter Term 2016

Agenda

Motivation

→ Critical path heuristics are a method to relax planning tasks, and thus automatically compute heuristic functions h.

We (almost) cover the 4 different methods currently known:

- Critical path heuristics. → **This Lecture**
- Delete relaxation.→ **Next Lecture**
- Abstractions.→ **Not Covered**
- Landmarks.→ **(Maybe) in Next Next Lecture**

→ Each of these have advantages and disadvantages. None strictly dominates any other, neither in practice nor in theory.

We introduce the method in STRIPS

Critical Path Heuristics: Basic Idea

"Approximate the cost of a goal set by the most costly sub-goal."

Assume uniform costs. Then $h(I)$ is? 2 (Perth or Darwin).

Assume $G = \{v(Br), v(Ad)\}$. Then $h(I)$ is? 1.

But: In "the most costly sub-goal", we may use size > 1!

$\rightarrow$ It is easiest to understand this approximation in terms of approximate versions of an equation characterizing h^* by regression.

A Regression-Based Characterization of h^*

Definition (r^*). *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task. The perfect regression heuristic r^* for Π is the function $r^*(s) := r^*(s, G)$ where $r^*(s, g)$ is the point-wise greatest function that satisfies $r^*(s, g) =$*

$$\begin{cases} 0 & g \subseteq s \\ \min_{a \in A, regr(g,a) \text{ is defined}} c(a) + r^*(s, regr(g, a)) & \text{otherwise} \end{cases}$$

(Reminder: $regr(g, a)$ is defined if $add_a \cap g \neq \emptyset$ and $del_a \cap g = \emptyset$; then, $regr(g, a) = (g \setminus add_a) \cup pre_a$.)

$\rightarrow$ The cost of achieving a sub-goal g is 0 if it is true in s; else, it is the minimum of using any action a to achieve g.

Proposition. *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task. Then $r^* = h^*$.* (Proof omitted.)

Critical Path Heuristics: h^1

Definition (h^1). *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task. The critical path heuristic h^1 for Π is the function $h^1(s) := h^1(s, G)$ where $h^1(s, g)$ is the point-wise greatest function that satisfies $h^1(s, g) =$*

$$\begin{cases} 0 & g \subseteq s \\ min_{a \in A, regr(g,a) \text{ is defined}}\, c(a) + h^1(s, regr(g, a)) & |g| = 1 \\ \max_{g' \in g} h^1(s, \{g'\}) & |g| > 1 \end{cases}$$

$\rightarrow$ For singleton sub-goals g, use regression as in r^*. For sub-goal sets g, use the cost of the most costly singleton sub-goal $g' \in g$.

$\rightarrow$ "Feasible path" = Path $g_1 \xrightarrow{a_1} g_2 \dots g_{n-1} \xrightarrow{a_{n-1}} g_n$ where $g_1 \subseteq s$, $g_n \subseteq G$, and for all i $g_i \subseteq regr(g_{i+1}, a_i)$ and $|g_i| = 1$ (resp. $|g_i| \leq m$).

$\rightarrow$ "Critical path" = *Cheapest* feasible path through *the most costly* sub-goals g_i.

The h^1 Heuristic in "TSP" in Australia

- P: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- A: $drive(x, y)$ where x, y have a road.

$$c(drive(x,y)) = \begin{cases} 1 & \{x,y\} = \{Sy, Br\} \\ 1.5 & \{x,y\} = \{Sy, Ad\} \\ 3.5 & \{x,y\} = \{Ad, Pe\} \\ 4 & \{x,y\} = \{Ad, Da\} \end{cases}$$

- I: $at(Sy), v(Sy)$; G: $at(Sy), v(x)$ for all x.

- $h^1(I) = h^1(I, G) = h^1(I, \{at(Sy), v(Sy), v(Ad), v(Br), v(Pe), v(Da)\}) = \max(h^1(I, \{at(Sy)\}), \dots, h^1(I, \{v(Da)\}))$.
- $h^1(I, \{at(Sy)\}) = h^1(I, \{v(Sy)\}) = 0$.
- $h^1(I, \{v(Da)\}) = 4 + h^1(I, regr(\{v(Da)\}, drive(Ad, Da))) = 4 + h^1(I, \{at(Ad)\})$.
- $h^1(I, \{at(Ad)\}) = \min(3.5 + h^1(I, \{at(Pe)\}), 4 + h^1(I, \{at(Da)\}), 1.5 + h^1(I, \{at(Sy)\})) = 1.5$.
- So $h^1(I, \{v(Da)\}) = 5.5$. Further, $h^1(I, \{v(Pe)\}) = 5$ and $h^1(I, \{v(Br)\}) = 1$, hence $h^1(I) = 5.5$.
- Critical path is? $at(Sy) \xrightarrow{drive(Sy,Ad)} at(Ad) \xrightarrow{drive(Ad,Da)} at(Da)$.

Critical Path Heuristics: The General Case

Definition (h^m). *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task, and let $m \in \mathbb{N}$. The critical path heuristic h^m for Π is the function $h^m(s) := h^m(s, G)$ where $h^m(s, g)$ is the point-wise greatest function that satisfies $h^m(s, g) =$*

$$\begin{cases} 0 & g \subseteq s \\ \min_{a \in A, regr(g,a) \text{ is defined}} c(a) + h^m(s, regr(g, a)) & |g| \leq m \\ \max_{g' \subseteq g, |g'| \leq m} h^m(s, g') & |g| > m \end{cases}$$

$\rightarrow$ For sub-goal sets $|g| \leq m$, use regression as in r^*. For sub-goal sets $|g| > m$, use the cost of the most costly m-subset g'.

$\rightarrow$ Like h^1, basically just replace "1" with "m".

$\rightarrow$ For fixed m, $h^m(s, g)$ can be computed in time polynomial in Π. (See next section.)

Critical Path Heuristics: Properties

Is h^m safe/goal-aware/admissible/consistent? Yes:

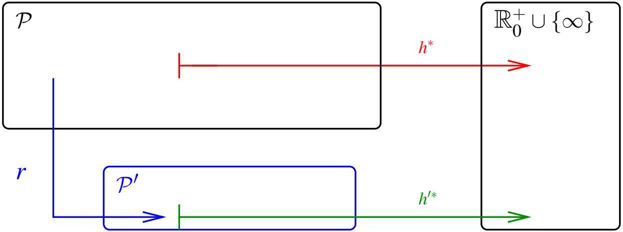
Proposition (h^m is Admissible). *h^m is consistent and goal-aware, and thus also admissible and safe.*

→ Intuition: h^m is admissible because it is always more difficult to achieve larger sub-goals.

Proposition (h^m is Perfect in the Limit). *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task. Then there exists $m \in \mathbb{N}$ so that $h^m = h^*$.*

Proof. Simply set $m := |F|$. Then the case $|g| > m$ will never be used, and thus $h^m = r^*$.

Critical Path Heuristics as Relaxations



where, for all $\Pi \in \mathcal{P}$, $h'^*(r(\Pi)) \leq h^*(\Pi)$.

For critical path heuristics h^m:

- Problem $\mathcal{P}$: All STRIPS planning tasks.
- Simpler problem $\mathcal{P}'$: Solving the h^m equations.
- Perfect heuristic h'^* for $\mathcal{P}'$: h^m. Note that $h^m \circ r(\Pi) \leq h^*(\Pi)$.
- Transformation r: Generate the equations.

$\rightarrow$ Is this a native relaxation? No.

$\rightarrow$ Is this relaxation efficiently constructible? Yes.

Dynamic Programming Computation

Basic idea:

"Initialize $h^m(s, g)$ to 0 if $g \subseteq s$, and to ∞ otherwise. Then keep updating the value of each g based on the values computed so far, until the values converge."

- We start with an iterative definition of h^m that makes this approach explicit.
- We define a generalization of the Bellman-Ford algorithm that corresponds to that iterative definition.
- We point out the relation to general fixed point mechanisms.

Iterative Definition of h^m

Definition (Iterative h^m). *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task, and let $m \in \mathbb{N}$. The iterative h^m heuristic h_i^m is defined by*

$$h_0^m(s, g) := \begin{cases} 0 & g \subseteq s \\ \infty & \text{otherwise} \end{cases}$$

and

$$h_{i+1}^m(s, g) := \begin{cases} \min[h_i^m(s, g), \min_{a \in A, regr(g,a) \text{ is defined}} c(a) + h_i^m(s, regr(g, a))] & |g| \leq m \\ \max_{g' \subseteq g, |g'| \leq m} h_{i+1}^m(s, g') & |g| > m \end{cases}$$

Proposition. *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task. Then the series $\{h_i^m\}_{i=0,\dots}$ converges to h^m.*

Generalized Bellman-Ford

Generalized Bellman-Ford (adds maximization to standard algorithm)

new table $T_0^m(g)$, for $g \subseteq F$ with $|g| \leq m$
For all $g \subseteq F$ with $|g| \leq m$: $T_0^m(g) := \begin{cases} 0 & g \subseteq s \\ \infty & \text{otherwise} \end{cases}$
fn $c_i(g) := \begin{cases} T_i^m(g) & |g| \leq m \\ \max_{g' \subseteq g, |g'| \leq m} T_i^m(g') & |g| > m \end{cases}$
fn $f_i(g) := \min[c_i(g), \min_{a \in A, regr(g,a) \text{ is defined}} c(a) + c_i(regr(g,a))]$
$i := 0$
do forever:
    new table $T_{i+1}^m(g)$, for $g \subseteq F$ with $|g| \leq m$
    For all $g \subseteq F$ with $|g| \leq m$: $T_{i+1}^m(g) := f_i(g)$
    if $T_{i+1}^m = T_i^m$ **then** stop **endif**
    $i := i + 1$
enddo

Proposition. $h_i^m(s,g) = c_i(g)$ *for all* i *and* g. (Easy.)

$\rightarrow$ If we want to know only the converged h^m, it is of course not necessary to allocate a new table for each i. Presented this way here only for simplicity.

Bellman-Ford for $m = 1$ in "TSP" in Australia

- P: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- A: $drive(x, y)$ where x, y have a road.

$$c(drive(x,y)) = \begin{cases} 1 & \{x,y\} = \{Sy, Br\} \\ 1.5 & \{x,y\} = \{Sy, Ad\} \\ 3.5 & \{x,y\} = \{Ad, Pe\} \\ 4 & \{x,y\} = \{Ad, Da\} \end{cases}$$

- I: $at(Sy), v(Sy)$; G: $at(Sy), v(x)$ for all x.

Content of Tables T_i^1:

i	$at(Sy)$	$at(Ad)$	$at(Br)$	$at(Pe)$	$at(Da)$	$v(Sy)$	$v(Ad)$	$v(Br)$	$v(Pe)$	$v(Da)$
0	0	∞	∞	∞	∞	0	∞	∞	∞	∞
1	0	1.5	1	∞	∞	0	1.5	1	∞	∞
2	0	1.5	1	5	5.5	0	1.5	1	5	5.5
3	0	1.5	1	5	5.5	0	1.5	1	5	5.5

$\rightarrow$ So what is $h^1(I)$? 5.5.

Bellman-Ford for $m = 2$ in Very Simple "TSP" in Australia

- P: $at(Sy)$, $at(Br)$, $v(Sy)$, $v(Br)$.
- A: $drive(Sy, Br)$, $drive(Br, Sy)$; cost 1.
- I: $at(Sy)$, $v(Sy)$; G: $at(Sy)$, $v(Sy)$, $v(Br)$.

Content of Tables T_i^2:

i	$at(Sy)$	$at(Br)$	$v(Sy)$	$v(Br)$	$at(Sy)$, $at(Br)$	$at(Sy)$, $v(Sy)$	$at(Sy)$, $v(Br)$	$at(Br)$, $v(Sy)$	$at(Br)$, $v(Br)$	$v(Sy)$, $v(Br)$
0	0	∞	0	∞	∞	0	∞	∞	∞	∞
1	0	1	0	1	∞	0	∞	1	1	1
2	0	1	0	1	∞	0	2	1	1	1
3	0	1	0	1	∞	0	2	1	1	1

$\rightarrow$ So what is $h^2(I)$? $2 = h^*(I)$. And what is $h^1(I)$? 1.

$\rightarrow$ Note that $h^2(\{at(Sy), at(Br)\}) = \infty$: we recognize the invariant that "the same variable" can only have one value at a time.

Bellman-Ford Algorithm: Runtime

Proposition. *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task, and let $m \in \mathbb{N}$ be fixed. Then the generalized Bellman-Ford algorithm runs in time polynomial in the size of Π.*

[**Proof Sketch.** With fixed m, the number of size-m fact sets is polynomial in the size of Π, so obviously each iteration of generalized Bellman-Ford runs in time polynomial in that size. The number of iterations until convergence is bounded by $|A| + 1$: by that time, all feasible paths are captured by the tables.]

$\rightarrow$ For any fixed m, the critical path heuristic h^m can be computed in polynomial time.

$\rightarrow$ In other words, for any fixed m the underlying relaxation is efficiently computable.

$\rightarrow$ In practice, only $m = 1, 2$ are used; higher values of m are typically infeasible.

Graphplan Representation: The Case $m = 1$

1-Planning Graphs

$F_0 := s$; $i := 0$
while $G \not\subseteq F_i$ **do**
    $A_i := \{a \in A \mid pre_a \subseteq F_i\}$
    $F_{i+1} := F_i \cup \bigcup_{a \in A_i} add_a$
    if $F_{i+1} = F_i$ **then** stop **endif**
    $i := i + 1$
endwhile

1-Planning Graph for "TSP" in Australia

- P: $at(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$; $v(x)$ for $x \in \{Sy, Ad, Br, Pe, Ad\}$.
- A: $drive(x, y)$ where x, y have a road.

$$c(drive(x,y)) = \begin{cases} 1 & \{x,y\} = \{Sy, Br\} \\ 1.5 & \{x,y\} = \{Sy, Ad\} \\ 3.5 & \{x,y\} = \{Ad, Pe\} \\ 4 & \{x,y\} = \{Ad, Da\} \end{cases}$$

- I: $at(Sy), v(Sy)$; G: $at(Sy), v(x)$ for all x.

Content of Fact Sets F_i:

i	$at(Sy)$	$at(Ad)$	$at(Br)$	$at(Pe)$	$at(Da)$	$v(Sy)$	$v(Ad)$	$v(Br)$	$v(Pe)$	$v(Da)$
0	yes	no	no	no	no	yes	no	no	no	no
1	yes	yes	yes	no	no	yes	yes	yes	no	no
2	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes
3	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes

$\rightarrow$ Rings a bell? We got a "yes" for i, g if and only if $T^1_i(g) \neq \infty$, cf. slide 17.

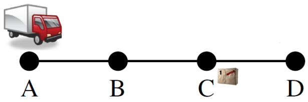
1-Planning Graphs vs. h^1

Definition. *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task. The 1-planning graph heuristic h^1_{PG} for Π is the function $h^1_{\text{PG}}(s) := \min\{i \mid s \subseteq F_i\}$, where F_i are the fact sets computed by a 1-planning graph, and the minimum over an empty set is ∞.*

Proposition. *Let $\Pi = (F, A, c, I, G)$ be a STRIPS planning task with uniform costs. Then $h^1_{\text{PG}} = h^1$.*

$\rightarrow$ Intuition: A 1-planning graph is like Bellman-Ford, except that it represents not all facts but only those that have been reached (value $\neq \infty$), and instead of a fact-value table it only remembers that set.

Questionnaire



- Initial state I: $t(A), p(C)$.
- Goal G: $t(A), p(D)$.
- Actions A: $drXY, loX, ulX$.

Question!

In this planning task, what is the value of $h^1(I)$?

(A): 0 (B): 2

(C): 4 (D): 5

→ A critical path is $t(A) \rightarrow t(B) \rightarrow t(C) \rightarrow p(T) \rightarrow p(D)$. (C) is correct.

Question!

In this planning task, what is the value of $h^2(I)$?

(A): 5 (B): 8

Summary

- The critical path heuristics h^m estimate the cost of reaching a sub-goal g by the most costly m-subset of g.
- This is admissible because it is always more difficult to achieve larger sub-goals.
- h^m can be computed using dynamic programming, i.e., initializing true m-subsets g to 0 and false ones to ∞, then applying value updates until convergence.
- This computation is polynomial in the size of the planning task, given fixed m. In practice, $m = 1, 2$ are used; $m > 2$ is typically infeasible.
- Planning graphs correspond to dynamic programming with uniform costs, using a particular representation of reached/unreached m-subsets g.

Historical Remarks

- The first critical path heuristic was introduced in the Graphplan system [*Blum and Furst, AI-97*], which uses h^2 computed by a 2-planning graph.[1] Graphplan's success can mainly be traced to the detection of invariants as on slide 18.
- 1-planning graphs are commonly referred to as relaxed planning graphs. This is because they're identical to Graphplan's 2-planning graphs when ignoring the delete lists [*Hoffmann, JAIR-01*].
- Graphplan spawned a huge amount of follow-up work.
- Nowadays, h^m is not in wide use anymore; its most prominent application right now is in a modified form that allows to compute improved delete-relaxation heuristics, cf. slide 30.

[1] Actually, Graphplan does parallel planning (a simple form of temporal planning), and uses a version of 2-planning graphs reflecting this. I'm sparing you the details since parallel planning is generally considered to not be very relevant in practice.

An (Important) Technical Remark

Reminder: Search Space for Progression

- start() $= I$
- succ(s) $= \{(a, s') \mid \Theta_\Pi$ has the transition $s \xrightarrow{a} s'\}$

$\rightarrow$ Need to compute $h^m(s) = h^m(s, G) \Rightarrow$ one call of dynamic programming for every different search state s!

Reminder: Search Space for Regression

- start() $= G$
- succ(g) $= \{(a, g') \mid g' = regr(g, a)\}$

$\rightarrow$ Need to compute $h^m(I, g) = \max_{g' \subseteq g, |g'| \leq m} h^m(I, g') \Rightarrow$ a single call of dynamic programming, for $s = I$ before search begins!

$\rightarrow$ For $m = 1$, it is feasible to use progression and recompute the cost of the (singleton) sub-goals in every search state s. For $m = 2$ already, this is completely infeasible; all systems using h^2 do regression search, where all sub-goals can be evaluated relative to the dynamic programming outcome for I.

Reading

- *Admissible Heuristics for Optimal Planning* [*Haslum and Geffner, AIPS-00*].

  Available at:

  `http://www.dtic.upf.edu/~hgeffner/html/reports/admissible.ps`

  Content: The original paper defining the h^m heuristic function, and comparing it to the techniques previously used in Graphplan.

Reading, ctd.

- $h^m(P) = h^1(P^m)$: *Alternative Characterisations of the Generalisation from $h^{\max}$ to h^m* [*Haslum, ICAPS-09*].

  Available at: `http://users.cecs.anu.edu.au/~patrik/publik/pm4p2.pdf`

  Content: A recent paper showing how to characterize h^m in terms of h^1 in a compiled planning task that explicitly represents size-m conjunctions.

  Relevance here: this contains the only published account of the iterative h^m_i characterization of h^m.

  Relevance more generally: this yields another alternative computation of h^m. That alternative is not per se very useful, but variants thereof have been shown to allow the computation of powerful semi-delete relaxation heuristics (see next; not covered in this course).

- *Semi-Relaxed Plan Heuristics* [*Keyder, Hoffmann and Haslum, ICAPS-12*]. Best Paper Award at ICAPS'12.

  Available at: `http://fai.cs.uni-saarland.de/hoffmann/papers/icaps12a.pdf`

  Content: The semi-delete relaxation heuristics mentioned above.