Introduction	Critical Path Heuristics	Dynamic Programming	Graphplan	FDR	Conclusion	References

# Al Planning 8. Critical Path Heuristics It's a Long Way to the Goal, But How Long Exactly? Part I: Honing In On the Most Critical Subgoals

## Álvaro Torralba, Cosmina Croitoru



#### Winter Term 2018/2019

Thanks to Prof. Jörg Hoffmann for slide sources

Álvaro Torralba, Cosmina Croitoru

AI Planning

Introduction 000	Critical Path Heuristics 000000	Dynamic Programming 000000000	Graphplan 0000	o PDR	Conclusion	References
Agenda						

- 1 Introduction
- 2 Critical Path Heuristics
- 3 Dynamic Programming Computation
- Graphplan Representation [for Reference]
- 5 What about FDR Planning?

## 6 Conclusion

Introduction ●00	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 000000	References
We Nee	ed Heuristic F	-unctions!				

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

## We cover the 4 different methods currently known:

- $\bullet$  Critical path heuristics:  $\rightarrow$  This Chapter
- $\bullet$  Delete relaxation:  $\rightarrow$  Chapters 9 and 10
- Abstractions:  $\rightarrow$  Chapters 11-13
- Landmarks:  $\rightarrow$  Chapter 14
- LP Heuristics:  $\rightarrow$  Chapter 16

 $\rightarrow$  Each of these have advantages and disadvantages. (We will do a formal comparison in Chapter 17.)

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion 000000	References
C						

# Critical Path Heuristics: Basic Idea



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

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

**But:** In "the most costly subgoal", 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.

Álvaro Torralba, Cosmina Croitoru

AI Planning

Introduction 00●	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion 000000	References
Our Ag	enda for This	Chapter				

- Critical Path Heuristics: Introduces and illustrates the formal definition.
- Oynamic Programming Computation: The straightforward method to compute critical path heuristics.
- Graphplan Representation: A slightly less straigtforward method to compute critical path heuristics. I mention this here only because, historically, it was there first, and its terminology is all over the planning literature.
- What about FDR Planning? The above uses STRIPS as this is a little easier to discuss in the examples. In this section, we point out on 1 slide that (almost) everything remains exactly the same for FDR.



**Definition**  $(r^*)$ . Let  $\Pi = (P, A, c, I, G)$  be a STRIPS planning task. The perfect regression heuristic  $r^*$  for  $\Pi$  is the function  $r^*(s) := r^*(s, G)$ where  $r^*(s, g)$  is the point-wise greatest function<sup>1</sup> that satisfies  $r^*(s, g) =$ 

 $\left\{ \begin{array}{ll} 0 & g \subseteq s \\ min_{a \in A, regr(g, a) \neq \bot} c(a) + r^*(s, regr(g, a)) & \textit{otherwise} \end{array} \right.$ 

(Reminder Chapter 6:  $regr(g, a) \neq \bot$  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 subgoal g is 0 if it is true in s; else, it is the minimum of using any action a to achieve g.

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

<sup>1</sup> "point-wise greatest" is needed here, and in the following, only to correctly handle 0-cost actions. We might bother you with an **Exercise** on this. Álvaro Torralba, Cosmina Croitoru Al Planning Chapter 8: Critical Path Heuristics 8/40



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

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

 $\rightarrow$  For singleton subgoals g, use regression as in  $r^*$ . For subgoal sets g, use the cost of the most costly singleton subgoal  $g' \in g$ .

 $\rightarrow \text{"Path"} = g_1 \xrightarrow{a_1} g_2 \dots g_{n-1} \xrightarrow{a_{n-1}} g_n \text{ where } g_1 \subseteq s, \ g_n \subseteq G, \ g_i \neq g_j, \text{ and } g_i \subseteq regr(g_{i+1}, a_i). \ |g_i| = 1 \text{ here, } |g_i| \leq m \text{ for } h^m \text{ (up next)}.$ 

 $\rightarrow$  "Critical path" = Cheapest path through the most costly subgoals  $g_i$ .

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 000000	References
The $h^1$	Houristic in	"TSP" in Au	stralia			



- 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  $h^1(I) = 5.5$ .

• The critical path is?  $at(Sy) \xrightarrow{drive(Sy,Ad)} at(Ad) \xrightarrow{drive(Ad,Da)} at(Da)$ .

Álvaro Torralba, Cosmina Croitoru

Chapter 8: Critical Path Heuristics

10/40



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

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

 $\rightarrow$  For subgoal sets  $|g| \leq m$ , use regression as in  $r^*$ . For subgoal 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 the size of  $\Pi.$  (See next section.)

Álvaro Torralba, Cosmina Croitoru

11/40

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 000000	References
C 111 1		d n d				

# Critical Path Heuristics: Properties

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

**Proof Sketch.** Goal-awareness is obvious. We need to prove that  $h^m(s) \leq h^m(s') + c(a)$  for all transitions  $s \xrightarrow{a} s'$ . Since  $s \supseteq regr(s', a)$ , a critical path  $\vec{p}$  for  $h^m(s')$  can be pre-fixed by a to obtain an upper bound on  $h^m(s)$ : all subgoals at the start of  $\vec{p}$  are contained in s', and are achieved by a in s.

 $\rightarrow$  Intuition:  $h^m$  is admissible because it is always more difficult to achieve larger subgoals (so m-subsets can only be cheaper).

 $\rightarrow$  Any ideas about what happens when we compare  $h^{m+1}$  to  $h^m$ ? **Proposition (** $h^m$  gets more accurate as m grows).  $h^{m+1}$  dominates  $h^m$ . **Proof Intuition:** "It is always more difficult to achieve larger subgoals."

→ Any ideas about what happens when we let m go to  $\infty$ ? **Proposition (** $h^m$  is perfect in the limit). There exists m s.t.  $h^m = h^*$ . **Proof.** Setting m := |P|, the case |g| > m will never be used, so  $h^m = r^*$ .

Introduction 000	Critical Path Heuristics 00000●	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion 000000	References			
Questionnaire									
A	B C	● Initi ● Goa D ● Acti	al state $I: t$ G: t(A), p ons $A: drX$	t(A), p(D). b(D). XY, loX	C).				
Questio	n!								
In this	planning task,	what is the valu	le of $h^1({\mathbb Z})$	I) <b>?</b>					
(A): 2		(В	): 3						
(C): 4		(D	): 5						
ightarrow A criti	cal path is $t(A) \rightarrow$	$t(B) \to t(C) \to p(C)$	$(T) \to p(D)$	). (C) i	s correct.				

#### Question!

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

(A): 5

 $\rightarrow$  For all subgoals g generated, either  $|g|\leq 2$ , or g must request more than one position for either the truck or the package, which in this domain will be recognized i.e.  $h^2(I,g)=\infty.$  Thus  $h^2(I)=r^*(I)$  and (B) is correct.

(B): 8

Álvaro Torralba, Cosmina Croitoru

AI Planning

Introduction 000	Critical Path Heuristics	Dynamic Programming ●00000000	Graphplan 0000	FDR o	Conclusion 000000	References
Overvie	ew					

## Basic idea:

"Consider all size- $\leq m$  subgoals g. Initialize  $h^m(s,g)$  to 0 if  $g \subseteq s$ , and to  $\infty$  otherwise.

Then keep updating the value of each g based on actions applied to 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 dynamic programming algorithm that corresponds to this iterative definition.
- We point out the relation to general fixed point mechanisms.

Introduction 000	Critical Path Heuristics 000000	Oynamic Programming 0●0000000	Graphplan 0000	o FDR	Conclusion	References
Iterative	e Definition o	of $h^m$				

**Definition (Iterative**  $h^m$ ). Let  $\Pi = (P, A, c, I, G)$  be a STRIPS planning task, and let  $m \in \mathbb{N}$ . The iterative  $h^m$  heuristic  $h_i^m$  is defined by  $h_0^m(s,g) :=$ 

 $\left\{\begin{array}{ll} 0 & g \subseteq s \\ \infty & otherwise \end{array}\right.$ 

 $\begin{array}{l} \text{and } h^m_{i+1}(s,g) := \\ \left\{ \begin{array}{l} \min[h^m_i(s,g), \min_{a \in A, regr(g,a) \neq \bot} c(a) + h^m_i(s, regr(g,a))] & |g| \leq m \\ \max_{g' \subseteq g, |g'| = m} h^m_{i+1}(s,g') & |g| > m \end{array} \right. \end{array}$ 

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

**Proof Sketch:** (i) Convergence: If  $h_{i+1}^m(s,g) \neq h_i^m(s,g)$ , then  $h_{i+1}^m(s,g) < h_i^m(s,g)$ ; that can happen only finitely often because each decrease is due to a new path for g. (ii) If  $h_{i+1}^m = h_i^m$  then  $h_i^m$  satisfies the  $h^m$  equation (direct from definition). (iii) No function greater than  $h_i^m$  at any point can satisfy the  $h^m$  equation (easy by induction over i).

Introduction	Critical Path Heuristics	Dynamic Programming	Graphplan	FDR	Conclusion	References
		00000000				

# Dynamic Programming

#### Dynamic Programming Algorithm

 $\begin{array}{l} \text{new table } T_0^m(g), \text{ for } g \subseteq P \text{ with } |g| \leq m \\ \text{For all } g \subseteq P \text{ with } |g| \leq m \text{: } T_0^m(g) \coloneqq \left\{ \begin{array}{l} 0 & g \subseteq s \\ \infty & \text{otherwise} \end{array} \right. \\ \text{fn } Cost_i(g) \coloneqq \left\{ \begin{array}{l} T_i^m(g) & |g| \leq m \\ \max_{g' \subseteq g, |g'| = m} T_i^m(g') & |g| > m \end{array} \right. \\ \text{fn } Next_i(g) \coloneqq \min[Cost_i(g), \min_{a \in A, regr(g, a) \neq \bot} c(a) + Cost_i(regr(g, a))] \\ i \coloneqq 0 \end{array} \right. \end{array}$ 

#### do forever:

new table  $T_{i+1}^m(g)$ , for  $g \subseteq P$  with  $|g| \leq m$ For all  $g \subseteq P$  with  $|g| \leq m$ :  $T_{i+1}^m(g) := Next_i(g)$ if  $T_{i+1}^m = T_i^m$  then stop endif i := i + 1enddo

**Proposition.**  $h_i^m(s,g) = Cost_i(g)$  for all *i* and *g*. (Proof is easy.)

 $\rightarrow$  This is very inefficient! (Optimized for readability.) We can use "Generalized Dijkstra" instead, maintaining the frontier of cheapest m-tuples reached so far.

Introduction 000	Critical Path Heuristics	Dynamic Programming 000●00000	Graphplan 0000	PDR 0	Conclusion 000000	References
luct fo	r the Decord.	Eived Daint	Earmul	ation		

#### Fixed Point Algorithm – Template!

 $\begin{array}{l} \mbox{new table } T^m(g), \mbox{ for } g \subseteq P \mbox{ with } |g| \leq m \\ \mbox{For all } g \subseteq P \mbox{ with } |g| \leq m ; \ T^m(g) := \left\{ \begin{array}{ll} 0 & g \subseteq s \\ \infty & \mbox{ otherwise} \end{array} \right. \\ \mbox{ fn } Cost(g) := \left\{ \begin{array}{ll} T^m(g) & |g| \leq m \\ \max_{g' \subseteq g, |g'| = m} T^m(g') & |g| > m \end{array} \right. \\ \mbox{ fn } Next(g) := \min[Cost(g), \min_{a \in A, regr(g,a) \neq \bot} c(a) + Cost(regr(g,a))] \\ \mbox{ while } \exists g \subseteq P, |g| \leq m : T^m(g) \neq Next(g) \mbox{ do:} \\ \mbox{ select one such } g \\ T^m(g) := Next(g) \\ \mbox{ endwhile} \end{array} \right.$ 

**Proposition.** Once the algorithm stops,  $h^m(s,g) = Cost(g)$  for all g. **Proof Sketch:** Similar to that for convergence of  $h_i^m$  to  $h^m$ .

 $\rightarrow$  This algorithm is not fully specified (hence "template"): How to select g s.t.  $T^m(g) \neq Next(g)$ ? We will use dynamic programming for simplicity.

Álvaro Torralba, Cosmina Croitoru

Chapter 8: Critical Path Heuristics

18/40

Introduction 000	Critical Path Heuristics	Dynamic Programming 0000●0000	Graphplan 0000	FDR o	Conclusion 000000	References
Examp	le: $m = 1$ in	"Logistics"				
		Easte D: t(	m) m c [ A E	ות מא		



- Facts  $P: t(x) x \in \{A, B, C, D\};$  $p(x) x \in \{A, B, C, D, T\}.$
- Initial state I:  $\{t(A), p(C)\}$ .
- Goal  $G: \{t(A), p(D)\}.$
- Actions A (unit costs): drive(x, y), load(x), unload(x).

E.g.: load(x): pre t(x), p(x); add p(T); del p(x).

## Content of Tables $T_i^1$ :

i	t(A)	t(B)	t(C)	t(D)	p(T)	p(A)	p(B)	p(C)	p(D)
0	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	$\infty$
1	0	1	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	0	$\infty$
2	0	1	2	$\infty$	$\infty$	$\infty$	$\infty$	0	$\infty$
3	0	1	2	3	3	$\infty$	$\infty$	0	$\infty$
4	0	1	2	3	3	4	4	0	4
5	0	1	2	3	3	4	4	0	4

 $\rightarrow$  So  $h^1(I) = 4$ . (Cf. slide 13)

**Note:** This table computation always first finds the *shortest* path to achieve a subgoal g. Hence, with unit action costs, the value of g is fixed once it becomes  $< \infty$ , and equals the i where that happens. With non-unit action costs, neither is true.

Álvaro Torralba, Cosmina Croitoru

AI Planning

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000●000	Graphplan 0000	PDR 0	Conclusion 000000	References
-						

# Example: m = 2 in "Dompteur"



- $P = \{alive, haveTiger, tamedTiger, haveJump\}.$ Short:  $P = \{A, hT, tT, J\}.$
- Initial state I: alive.
- Goal G: alive, haveJump.

 Actions A: getTiger: pre alive; add haveTiger tameTiger: pre alive, haveTiger; add tamedTiger jumpTamedTiger: pre alive, tamedTiger; add haveJump jumpTiger: pre alive, haveTiger; add haveJump; del alive

Content of Tables  $T_i^2$ :

i	A	hT	tT	J	A,	A,	A,	hT,	hT,	tT,
					hT	tT	J	tT	J	J
0	0	$\infty$								
1	0	1	$\infty$	$\infty$	1	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
2	0	1	2	2	1	2	$\infty$	2	2	$\infty$
3	0	1	2	2	1	2	3	2	2	3

 $\rightarrow$  So  $h^2(I) = 3$ , in contrast to  $h^1(I) = 2$ .

Note reg A, J in step 2: Each of A and J is reached, but not both together: jumpTiger deletes A so we can't regress this subgoal over that action; jumpTamedTiger yields the regressed subgoal  $\{A, tT\}$  whose value at 1 is  $\infty$ .

Álvaro Torralba, Cosmina Croitoru

AI Planning

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 000000	References
Examp	le $m-1$ in	"TSP" in Au	stralia			



- 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*.

i	at(Sy)	at(Ad)	at(Br)	at(Pe)	at(Da)	v(Sy)	v(Ad)	v(Br)	v(Pe)	v(Da)
0	0	$\infty$	$\infty$	$\infty$	$\infty$	0	$\infty$	$\infty$	$\infty$	$\infty$
1	0	1.5	1	$\infty$	$\infty$	0	1.5	1	$\infty$	$\infty$
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.

Content of Tables  $T_i^1$ :

Example	•• ••••	0 = 1	laur Cimanla	"TCD"	: Λ.	مالمسخم	
000	000000		000000000	0000		000000	
Introduction	Critical Path H	euristics	Dynamic Programming	Graphplan	FDR	Conclusion	References

# Example: m = 2 in Very Simple "TSP" in Australia



## Content of Tables $T_i^2$ :

- Facts P: at(Sy), at(Br), v(Sy), v(Br).
- Initial state I: at(Sy), v(Sy).
- Goal G: at(Sy), v(Sy), v(Br).
- Actions A: drive(Sy, Br), drive(Br, Sy); both cost 1. drive(Sy, Br):

pre at(Sy); add at(Br), v(Br); del at(Sy). drive(Br, Sy):

pre at(Br); add at(Sy), v(Sy); del at(Br).

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

 $\rightarrow$  So  $h^2(I) = 2$ , in contrast to  $h^1(I) = 1$ .

**NOTE reg** at(Sy), v(Br)) in step 1: Each of at(Sy) and v(Br) is reached, but not both together: drive(Sy, Br) deletes at(Sy) so we can't regress this subgoal over that action; drive(Br, Sy) yields the regressed subgoal  $\{at(Br), v(Br)\}$  whose value at iteration 0 is  $\infty$ .

Álvaro Torralba, Cosmina Croitoru

AI Planning

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000●	Graphplan 0000	FDR o	Conclusion 000000	References
Runtim	е					

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

**Proof Sketch.** With fixed m, the number of size-m fact sets is polynomial in the size of  $\Pi$ , so obviously each iteration of the algorithm runs in time polynomial in that size. The fixed point is reached at the latest at  $i + 1 = |P|^m + 1$ , as each path has length at most  $|P|^m$ .

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

### Remarks:

- In practice, only m = 1, 2 are used; higher values of m are infeasible.
- However! Instead of considering all "atomic subgoals" of size  $\leq m$ , one can select an arbitrary set C of atomic subgoals!

 $\rightarrow h^{C}$ , currently investigated in FAI, great results in learning to recognize dead-ends [Steinmetz and Hoffmann (2016)].

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan ●000	FDR o	Conclusion 000000	References
Graphp	lan Represen	tation: The C	Case $m$	= 1		

#### 1-Planning Graphs

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

Rings a bell? This was called "relaxed planning graph" in Al'18 Chapter 14. Slide 33 explains why.

000	Critical Path Heuristics 000000	Overammic Programming	Graphplan 0●00	o O	Conclusion	References
1-Plan	ning Graph fo	r "Logistics"				

o



- Initial state I: t(A), p(C).
- Goal G: t(A), p(D).
- Actions A: dr(X, Y), lo(X), ul(X).

#### **Content of Fact Sets** *F*<sub>*i*</sub>**:**

 $\mathbf{a}$ 

i	t(A)	t(B)	t(C)	t(D)	p(T)	p(A)	p(B)	p(C)	p(D)
0	yes	no	no	no	no	no	no	yes	no
1	yes	yes	no	no	no	no	no	yes	no
2	yes	yes	yes	no	no	no	no	yes	no
3	yes	yes	yes	yes	yes	no	no	yes	no
4	yes								
5	yes								

 $\rightarrow$  Rings a bell? We got a "yes" for i, g if and only if  $T_i^1(g) \neq \infty$ , cf. slide 19.

Álvaro Torralba, Cosmina Croitoru

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 00●0	FDR o	Conclusion 000000	References
1-Plann	ning Graphs v	rs. $h^1$				

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

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

**Proof Sketch:** Induction over the value i of  $h^1(s)$ . Trivial for base case i = 0. For the step case, assume that  $h^1_{\mathsf{PG}}(s) = h^1(s)$  for all s where  $h^1(s) \leq i$ , and show the same property for all s with  $h^1(s) \leq i + 1$ .  $h^1_{\mathsf{PG}}(s) < i + 1$  directly contradicts the assumption. To show  $h^1_{\mathsf{PG}}(s) \leq i + 1$ , it suffices to observe that  $h^1(pre_a) \leq i$  implies  $h^1_{\mathsf{PG}}(pre_a) \leq i$  by assumption.

 $\rightarrow$  Intuition: A 1-planning graph is like our dynamic programming algorithm for m=1, 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.

Álvaro Torralba, Cosmina Croitoru

AI Planning

Craphy	Jan Dankasan	tation. The (	Comoral	Casa		
			0000			
Introduction	Critical Path Heuristics	Dynamic Programming	Graphplan	FDR	Conclusion	References

## Graphplan Representation: The General Case

## *m*-Planning Graphs

$$\begin{split} F_0 &:= s; \ M_0 := \emptyset; \ i := 0 \\ \text{fn} \ Reached_i(g) &:= \left\{ \begin{array}{cc} True & g \subseteq F_i, \ \nexists g' \in M_i : g' \subseteq g \\ False & \text{otherwise} \end{array} \right. \\ \text{while not} \ Reached_i(G) \ \text{do} \\ A_i &:= \left\{ a \in A \mid Reached_i(pre_a) \right\} \\ F_{i+1} &:= F_i \cup \bigcup_{a \in A_i} add_a \\ M_{i+1} &:= \left\{ g \subseteq P \mid |g| \leq m, \forall a \in A_i : \text{not} \ Reached_i(regr(g, a)) \right\} \\ \text{if} \ F_{i+1} = F_i \ \text{and} \ M_{i+1} = M_i \ \text{then stop endif} \\ i &:= i+1 \\ \text{endwhile} \end{split}$$

 $\rightarrow$  Intuition: All *m*-subsets *g* of *F*<sub>i</sub> are reachable within *i* steps, except for those *g* listed in *M*<sub>i</sub> (the "mutexes").

 $\rightarrow$  Instead of listing the reached *m*-subsets, represent those that are not reached (and hope that there are fewer of those).

Álvaro Torralba, Cosmina Croitoru

AI Planning

000	000000	00000000	0000	•	000000			
Introduction	Critical Path Heuristics	Dynamic Programming	Graphplan	FDR	Conclusion	References		

# Critical Path Heuristics in FDR

... are exactly the same!

 $\rightarrow$  All definitions, results, and proofs apply, exactly as stated, also to FDR planning tasks. (See the single exception below.)

 $\rightarrow$  Remember (cf.  $\rightarrow$  Chapter 2): We refer to pairs (v, d) of variable and value as facts. We identify partial variable assignments with sets of facts.

The single non-verbatim-applicable statement, adapted to FDR:

**Proposition** ( $h^m$  is Perfect in the Limit). There exists m s.t.  $h^m = h^*$ .

**Proof.** Given the definition of regr(g, a) for FDR ( $\rightarrow$  Chapter 6), it is easy to see by induction that every subgoal g contains at most one fact for each variable  $v \in V$ . Thus, if we set m := |V|, then the case |g| > m will never be used, so  $h^m = r^*$ .

 $\rightarrow$  In FDR, it suffices to set m to the number of variables, as opposed to the number of variable values i.e. STRIPS facts, compare slide 12!

Álvaro Torralba, Cosmina Croitoru

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion •00000	References
Summa	ry					

- The critical path heuristics  $h^m$  estimate the cost of reaching a subgoal g by the most costly m-subset of g.
- This is admissible because it is always more difficult to achieve larger subgoals.
- $h^m$  can be computed using dynamic programming, i.e., initializing true *m*-subsets *g* to 0 and false ones to  $\infty$ , 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 unit costs, using a particular representation of reached/unreached *m*-subsets *g*.

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 00000	References
Historic	al Remarks					

- The first critical path heuristic was introduced in the Graphplan system [Blum and Furst (1997)], which uses  $h^2$  computed by a 2-planning graph.<sup>2</sup>
- 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 and Nebel (2001)].
- Graphplan spawned a huge amount of follow-up work [e.g., Kambhampati *et al.* (1997); Koehler *et al.* (1997); Koehler (1998); Kambhampati (2000)]; in particular, it was my personal "kindergarden planner".
- Nowadays, h<sup>m</sup> is not in wide use anymore; its most prominent application right now is in modified forms that allow to use arbitrary sets of atomic subgoals (see slide 36), or to compute improved delete-relaxation heuristics (→ Chapter 10).

<sup>&</sup>lt;sup>2</sup>Actually, Graphplan does parallel planning (a simplistic form of temporal planning), and uses a version of 2-planning graphs reflecting this. I omit the details since parallel planning is not relevant in practice.

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 000000	References
A Tech	nical Remark					

Reminder: Search Space for Progression

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

 $\rightarrow$  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
- $\operatorname{succ}(g) = \{(a,g') \mid g' = \operatorname{regr}(g,a)\}$

 $\rightarrow$  Need to compute  $h^m(I,g) = \max_{g' \subseteq g, |g'|=m} h^m(I,g') \Rightarrow$  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) subgoals in every search state s. For m=2 already, this is completely infeasible; all systems using  $h^2$  do regression search, where all subgoals can be evaluated relative to the dynamic programming outcome for I.

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion 000●00	References
Reading	Ţ					

• Admissible Heuristics for Optimal Planning [Haslum and Geffner (2000)]. Available at:

•  $h^m(P) = h^1(P^m)$ : Alternative Characterisations of the Generalisation from  $h^{\max}$  to  $h^m$  [Haslum (2009)].

Available at: http://users.cecs.anu.edu.au/~patrik/publik/pm4p2.pdf Content: Shows 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  $h_i^m$  characterization of  $h^m$ . Relevance more generally: yields an alternative computation of  $h^m$ . This is not per se useful, but variants thereof have been shown to allow the computation of powerful partial-delete-relaxation heuristics ( $\rightarrow$  Chapter 10).

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 0000●0	References
Readin	g, ctd.					

• Explicit Conjunctions w/o Compilation: Computing  $h^{FF}(\Pi^C)$  in Polynomial Time [Hoffmann and Fickert (2015)].

#### Available at:

http://fai.cs.uni-saarland.de/hoffmann/papers/icaps15b.pdf

Content: Introduces the  $h^C$  heuristic (cf. slide 23), which allows to select an arbitrary set C of atomic subgoals, and thus strictly generalizes  $h^m$ .

This is only a side note in the paper though, the actual concern is with defining and computing partial-delete-relaxation heuristics on top of  $h^C$ .

Introduction 000	Critical Path Heuristics	Dynamic Programming 000000000	Graphplan 0000	FDR o	Conclusion 00000●	References
Readin	g, ctd.					

• Towards Clause-Learning State Space Search: Learning to Recognize Dead-Ends [Steinmetz and Hoffmann (2016)].

#### Available at:

http://fai.cs.uni-saarland.de/hoffmann/papers/aaai16.pdf

Content: Specifies how to "learn" the atomic subgoals C based on states s where the search already knows that  $h^*(s) = \infty$ , yet where  $h^C(s) \neq \infty$ . The learning process adds new conjunctions into C, in a manner guaranteeing that  $h^C(s) = \infty$  afterwards.

Doing this systematically in a depth-first search, we obtain a framework that approaches the elegance of clause learning in SAT, finding and analyzing conflicts to learn knowledge that generalizes to other search branches.

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion 000000	References
Referer	nces I					

- Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1–2):279–298, 1997.
- Patrik Haslum and Hector Geffner. Admissible heuristics for optimal planning. In S. Chien, R. Kambhampati, and C. Knoblock, editors, *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, pages 140–149, Breckenridge, CO, 2000. AAAI Press, Menlo Park.
- Patrik Haslum.  $h^m(P) = h^1(P^m)$ : Alternative characterisations of the generalisation from  $h^{\max}$  to  $h^m$ . In Alfonso Gerevini, Adele Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 354–357. AAAI Press, 2009.
- Jörg Hoffmann and Maximilian Fickert. Explicit conjunctions w/o compilation: Computing  $h^{FF}(\Pi^C)$  in polynomial time. In Ronen Brafman, Carmel Domshlak, Patrik Haslum, and Shlomo Zilberstein, editors, *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press, 2015.

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion 000000	References
Referen	ces II					

- Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.
- Subbarao Kambhampati, Eric Parker, and Eric Lambrecht. Understanding and extending Graphplan. In S. Steel and R. Alami, editors, *Proceedings of the 4th European Conference on Planning (ECP'97)*, pages 260–272. Springer-Verlag, 1997.
- Subbarao Kambhampati. Planning graph as a (dynamic) CSP: Exploiting EBL, DDB and other CSP search techniques in graphplan. *Journal of Artificial Intelligence Research*, 12:1–34, 2000.
- Jana Koehler, Bernhard Nebel, Jörg Hoffmann, and Yannis Dimopoulos. Extending planning graphs to an ADL subset. In S. Steel and R. Alami, editors, *Proceedings of the 4th European Conference on Planning (ECP'97)*, pages 273–285. Springer-Verlag, 1997.
- Jana Koehler. Planning under resource constraints. In H. Prade, editor, Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98), pages 489–493, Brighton, UK, August 1998. Wiley.

Introduction 000	Critical Path Heuristics	Dynamic Programming 00000000	Graphplan 0000	FDR o	Conclusion 000000	References	
References III							

Marcel Steinmetz and Jörg Hoffmann. Towards clause-learning state space search: Learning to recognize dead-ends. In Dale Schuurmans and Michael Wellman, editors, *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (AAAI'16). AAAI Press, February 2016.