AIND: Planning Search – Heuristics Analysis

Markus Mayer

June 11, 2017

1 Problems and optimal plans

The three planning problems take place in the Air Cargo domain that consists of the object literals Cargo, Plane and Airport, the propositions At and In as well as the actions defined in listing 1.

Listing 1: Air Cargo Action Schema

```
Action(Load(c, p, a),
    Precond: At(c, a) ∧ At(p, a)
           ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    Effect: ¬ At(c, a) ∧ In(c, p))

Action(Unload(c, p, a),
    Precond: In(c, p) ∧ At(p, a)
           ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
    Effect: At(c, a) ∧ ¬ In(c, p))

Action(Fly(p, from, to),
    Precond: At(p, from)
           ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
    Effect: ¬ At(p, from) ∧ At(p, to))
```

Problem 1: The first planning problem using two cargo items, planes and airports is defined in listing 2. An optimal plan to it consists of 6 actions and is shown in listing 3. Since the planner does not know about durations, it cannot reason about the order of Fly and Unload operations that are indipendent across airplanes: It might be more efficient to schedule all Fly actions before all Unloads, because there is no need to wait for a plane to unload for another to start flying; however, none of that is encoded in the problem domain. Metrics for the different planning strategies are shown in table 1 and fig. 1 on page 5.

Listing 2: Problem 1 initial state and goal

```
Init(At(C1, SFO) ∧ At(C2, JFK)
   ∧ At(P1, SFO) ∧ At(P2, JFK)
   ∧ Cargo(C1) ∧ Cargo(C2)
   ∧ Plane(P1) ∧ Plane(P2)
   ∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

Listing 3: Problem 1 optimal plan

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
```

```
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

Problem 2: The second planning problem uses three cargo items, planes and airports and is defined in listing 4. An optimal plan to it consists of 9 actions and is shown in listing 5. Metrics for the different planning strategies are shown in table 2 and fig. 2 on page 6.

Listing 4: Problem 2 initial state and goal

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
   ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
   ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
   ∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
   ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

Listing 5: Problem 2 optimal plan

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
```

Problem 3: The third and last planning problem uses four cargo items and airports with only two planes; it is defined in listing 6. An optimal plan to it consists of 12 actions and is shown in listing 7. In this plan, plane P2 is loaded with both cargo C2 and C4 at the airports JFK (line 2) and ORD (line 4) respectively, and both are then flown to SFO (line 10). This is a valid plan, as an airplane does not have to be empty in order to be Loaded, according to the preconditions of that action. If required, the behavior of flying only single cargos could be achieved by placing a virtual Empty cargo into the plane, similar to the Empty block of the 8-Puzzle domain. Metrics for the different planning strategies are shown in table 3 and fig. 3 on page 7.

Listing 6: Problem 3 initial state and goal

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
   ∧ At(P1, SFO) ∧ At(P2, JFK)
   ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
   ∧ Plane(P1) ∧ Plane(P2)
   ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
```

Listing 7: Problem 3 optimal plan

```
Load(C1, P1, SFO)
Load(C2, P2, JFK) ; 1st load to P2
Fly(P2, JFK, ORD) ; P2 carries C2
Load(C4, P2, ORD) ; 2nd load to P2
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO) ; P2 carries C2 and C4
```

```
11 Unload(C2, P2, SFO)
12 Unload(C4, P2, SFO)
```

2 Uninformed and informed planning

Non-heuristic algorithms: The non-heuristic strategies used were *Breadth-first* and *Breadth-first Tree* (tree search using a FIFO queue-based frontier), *Depth-first Graph* (graph search using a stack-based frontier), *Depth-limited* and *Uniform cost* (best-first graph using the path cost as the score) search.

Out of these search strategies, the *depth-first graph* search consistently is among the fastest execution times, but never resulted in optimal plans; on the contrary, plan lengths resulting from this algorithm were one to two orders of magnitudes off from the optimal plan length, often undoing actions performed in previous states in order to then repeat this cycle. However, the number of node expansions and goal tests was the smallest within the group of uninformed algorithms and close to the best algorithms in the informed group. *Breadth-first* and *uniform cost* searches achieved similar results in number of espansions, goal tests, node creations and execution time, yielding optimal plans both on all problems.[1] For more complex problems, uniform cost search turned out to be slightly more efficient in execution time despite the slightly bigger amount of node explorations performed. The *Breadth-first tree* and *depth limited* searches failed to converge to a solution within a six hour timeframe for the problems 2 (table 2) and 3 (table 3). *Depth-limited* search was the worst performer for problem 2 in terms of execution time and the second-worst in problem 1, directly followed by *Breadth-first tree* search (table 1) .

Heuristic algorithms: The heuristic strategies used were *Recursive best-first*, *Greedy best-first* (best-first search using the heuristic as the score) and *A** (best-first with the sum of path cost and heuristic as the score) searches with *Constant cost (H1)* (every action has a cost heuristic of 1), *Precondition ignoring* (every action is applicable) and *Planning Graph Level-sum* heuristics.

Out of the informed strategies, *Greedy best-first Graph H1* consistently achieved the fastest execution time, but failed to yield optimal plans for the two more complex problems. The *A** algorithm with *Planning Graph Level-sum* heuristic, on the other hand, consistently resulted in the lowest amount of explored states and resulted in optimal plans, but also required an execution time that was about tenfold of the others (the exception here is *Recursive best-first Graph* search with *H1* (non-)heuristic which did not finish within a six hour timeframe for problems 2 and 3). The second-best (informed) algorithm across all problems appear *(*A*) with *ignoring preconditions* heuristic, which resulted in a significantly higher number of explored nodes and goal tests but required only a tenth of the execution time of the *Planning Graph Level-sum* variant.

The *Planning Graph Level-sum* heuristic results in the lowest number of explorations (significantly less than the competitors) and should consequently be a very good candidate for the best algorithm. Since the difference is in the heuristic, the comparatively long execution time must come from the planning graph construction and level-sum evaluation. Optimizing the implementation might result in a much better rating; as of writing this document, however, no further optimization was performed.

[1] Since the path cost tends to increase with the depth of the search tree, both strategies essentially use a similar approach.

Performance between classes: If plan optimality is not required, *Greedy best-first Graph H1* performed best according to execution time with number of explored states within one order of magnitude of the best algorithm; the plan length consistently was less than twice the length of the optimal plan for each problem. When number of explorations and plan optimality is critical, *A** with *Planning Graph Level-sum* heuristic is the go-to choice, with execution times within one order of magnitude of the fastest (optimal) algorithm.[2] If plan optimality and execution time are favored, *A** with *Ignore preconditions* is the best candidate across the given problems, yielding optimal plans and one of the fastest execution times even on the more complex problems. The *Depth-first Graph* algorithm finally resulted in the fastest execution times with low explorations but resulted in the longest plans. It might be worthwile to derive a plan using this algorithm first and then fine-tune it to obtain an optimal result.

In comparison to the uninformed algorithms, all optimal informed strategies were faster except for the simplest case of problem 1 (also ignoring *Planning Graph Level-sum* heuristic as discussed and *Recursive best-first* search with the *(*H1) non-heuristic which failed to converge). This leads to the conclusion that adding any guides (in the form of heuristics) that help directing the search in a graph results in faster convergence to the solution, even in presence of suboptimal heuristics.

[2]As discussed, an optimized implementation of both the planning graph and the heuristic might improve the rating in terms of execution time.

Table 1: Metrics for problem 1

Algorithm	**Expansions**	**Goal tests**	**New nodes**	**Length**	**Duration (s)**
Breadth-first	43	56	180	6	$1.86 \cdot 10^{-2}$
Breadth-first Tree	1,458	1,459	5,960	6	0.56
Depth-first Graph	21	22	84	20	$8.9 \cdot 10^{-3}$
Depth limited	101	271	414	50	$5.64 \cdot 10^{-2}$
Uniform cost	55	57	224	6	$2.21 \cdot 10^{-2}$
Recursive best-first H1	4,229	4,230	17,023	6	1.62
Greedy best-first H1	7	9	28	6	$3.5 \cdot 10^{-3}$
A* H1	55	57	224	6	$2.25 \cdot 10^{-2}$
A* Ignore Preconditions	41	43	170	6	$2.96 \cdot 10^{-2}$
A* PG Levelsum	11	13	50	6	0.6

The optimal plan length for this problem is 6 actions.

Figure 1: Metrics for problem 1

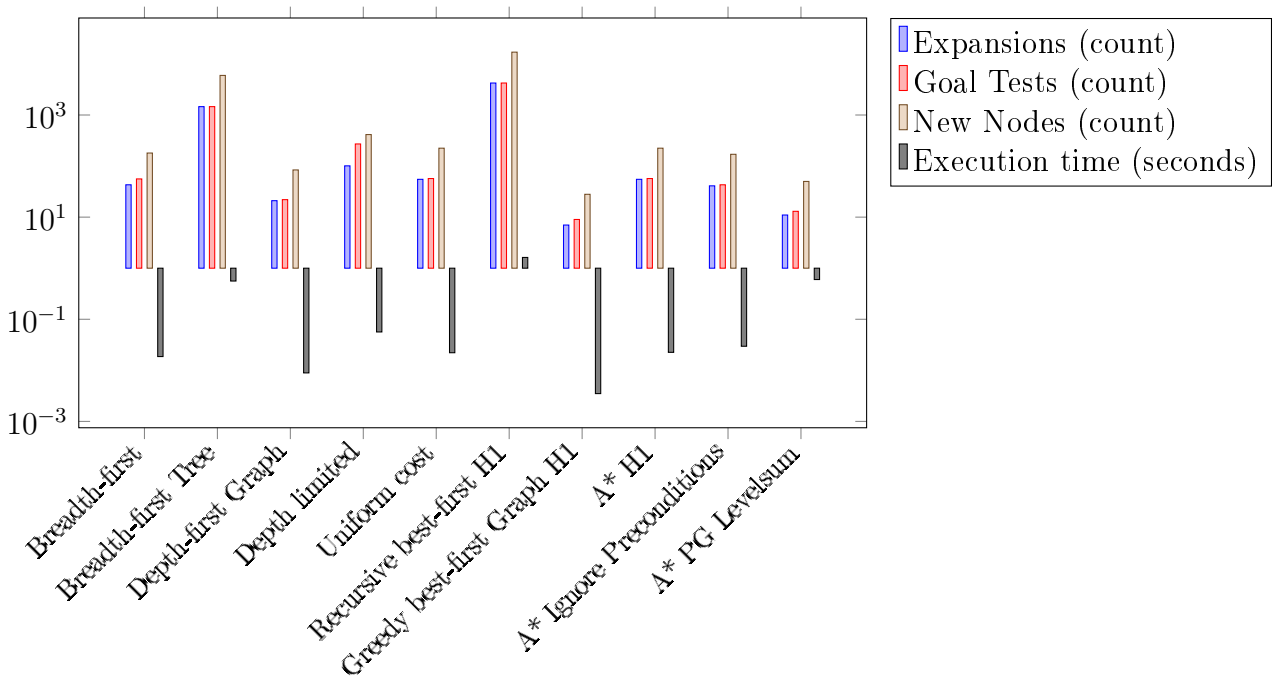

Table 2: Metrics for problem 2

Algorithm	**Expansions**	**Goal tests**	**New nodes**	**Length**	**Duration (s)**
Breadth-first	3,343	4,609	30,509	9	9.68
Breadth-first Tree					
Depth-first Graph	624	625	5,602	619	2.7
Depth limited	$2.23 \cdot 10^5$	$2.05 \cdot 10^6$	$2.05 \cdot 10^6$	50	1,006.6
Uniform cost	4,852	4,854	44,030	9	9.66
Recursive best-first H1					
Greedy best-first Graph H1	990	992	8,910	15	1.82
A* H1	4,852	4,854	44,030	9	9.47
A* Ignore Preconditions	1,450	1,452	13,303	9	4.78
A* PG Levelsum	86	88	841	9	56.25

The optimal plan length for this problem is 9 actions.Two of the ten heuristics could not be measured, because the execution time of the planner exceeded six hours.

Figure 2: Metrics for problem 2

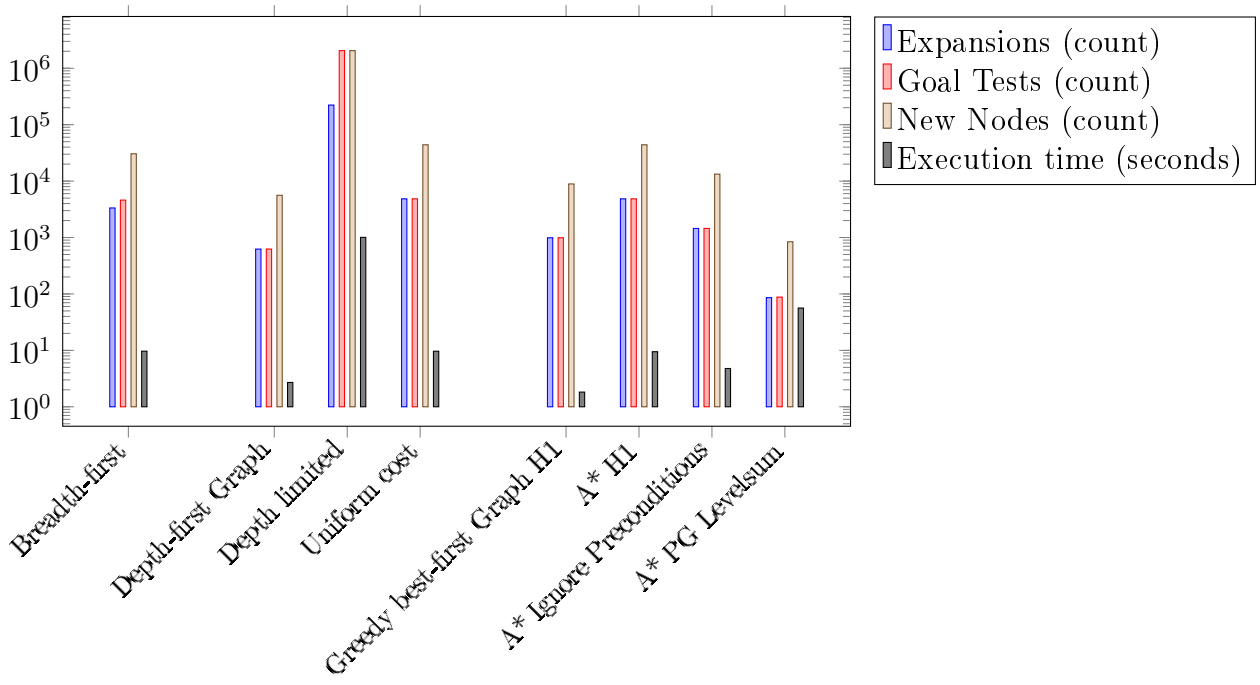

Table 3: Metrics for problem 3

Algorithm	**Expansions**	**Goal tests**	**New nodes**	**Length**	**Duration (s)**
Breadth-first	14,663	18,098	$1.3 \cdot 10^5$	12	71.75
Breadth-first Tree					
Depth-first Graph	408	409	3,364	392	1.35
Depth limited					
Uniform cost	18,235	18,237	$1.6 \cdot 10^5$	12	42.25
Recursive best-first H1					
Greedy best-first Graph H1	5,614	5,616	49,429	22	12.75
A* H1	18,235	18,237	$1.6 \cdot 10^5$	12	42.21
A* Ignore Preconditions	5,040	5,042	44,944	12	20.54
A* PG Levelsum	315	317	2,902	12	282.04

The optimal plan length for this problem is 12 actions. Three of the ten heuristics could not be measured, because the execution time of the planner exceeded six hours.

Figure 3: Metrics for problem 3