

Heuristic analysis

for planning problems for an Air Cargo transport

Introduction

The project is a part of Udacity Artificial Intelligence Nanodegree Program and consists of deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. It includes skeletons for the classes and functions needed to solve those problems.

Problem definitions

All problems are classical PDDL problems and defined in the Air Cargo domain. They have the same action schema defined, but different initial states and goals.

- 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 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))
```

- 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))
```

- 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)
```

$\wedge \text{Cargo}(C1) \wedge \text{Cargo}(C2) \wedge \text{Cargo}(C3) \wedge \text{Cargo}(C4)$
 $\wedge \text{Plane}(P1) \wedge \text{Plane}(P2)$
 $\wedge \text{Airport}(JFK) \wedge \text{Airport}(SFO) \wedge \text{Airport}(ATL) \wedge \text{Airport}(ORD))$
 $\text{Goal}(\text{At}(C1, JFK) \wedge \text{At}(C3, JFK) \wedge \text{At}(C2, SFO) \wedge \text{At}(C4, SFO))$

Uniformed non-heuristic search

I experimented with the following searches:

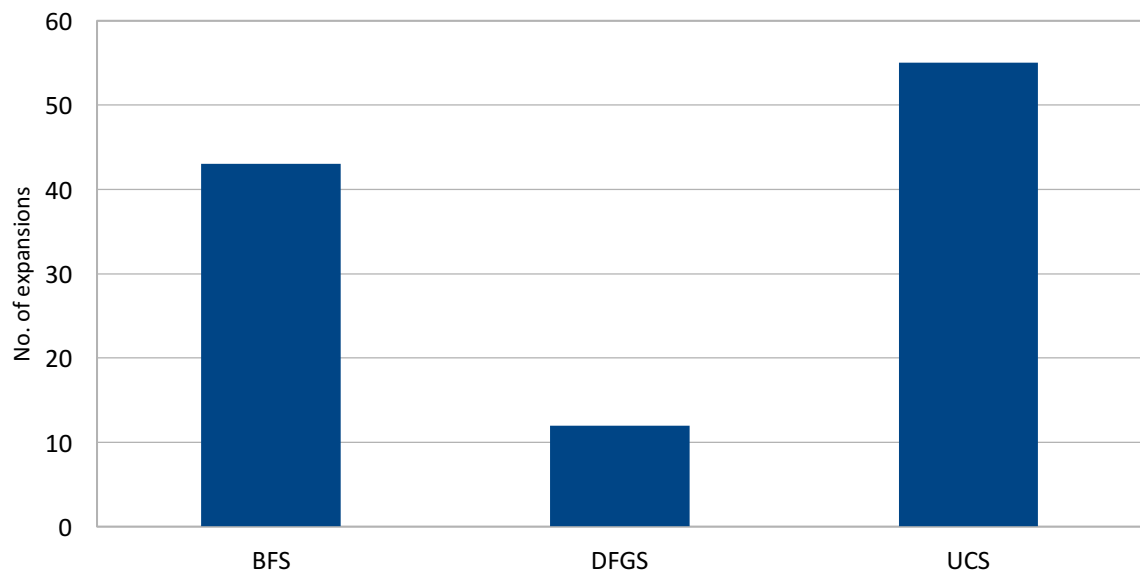
- **BFS** – Breadth First Search.
- **DFGS** – Depth First Graph Search.
- **UCS** – Uniform Cost Search.

The first two are obligatory according to the task description and the last one I selected, since it works at best for a non-heuristic search and I wanted to compare obligatory searches to it.

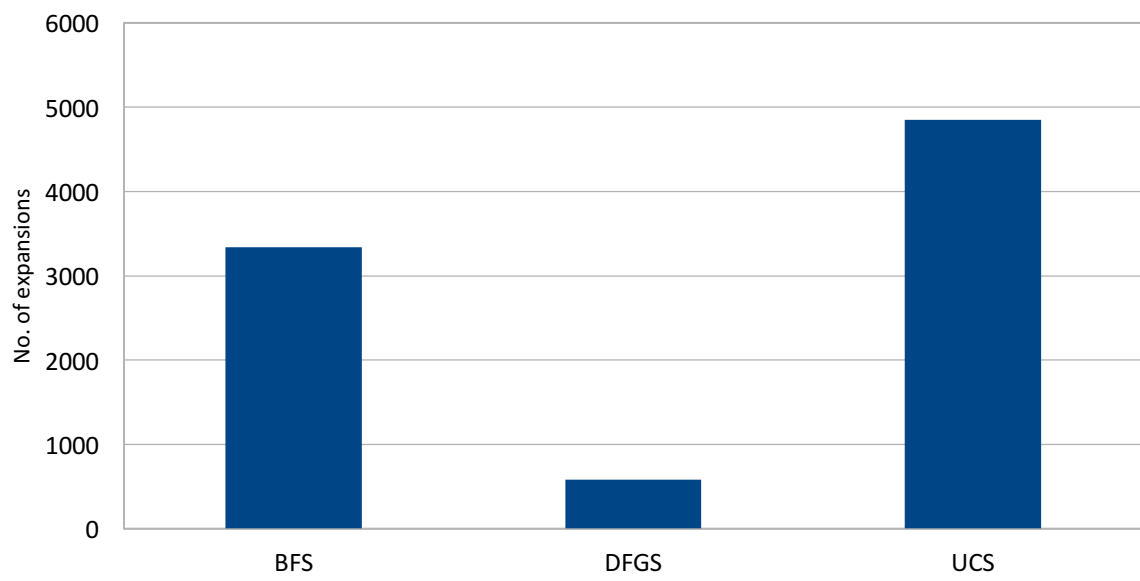
Problem	Search type	No. of expansions	No. of goal tests	Plan length	Time elapsed (sec.)	Is optimal
P1	BFS	43	56	6	0.028	Yes
P1	DFGS	12	13	12	0.008	No
P1	UCS	55	57	6	0.035	Yes
P2	BFS	3343	4609	9	11.769	Yes
P2	DFGS	582	583	575	2.704	No
P2	UCS	4852	4854	9	9.672	Yes
P3	BFS	14663	18098	12	87.707	Yes
P3	DFGS	627	628	596	2.763	No
P3	UCS	18235	18237	12	42.841	Yes

Memory consumption charts

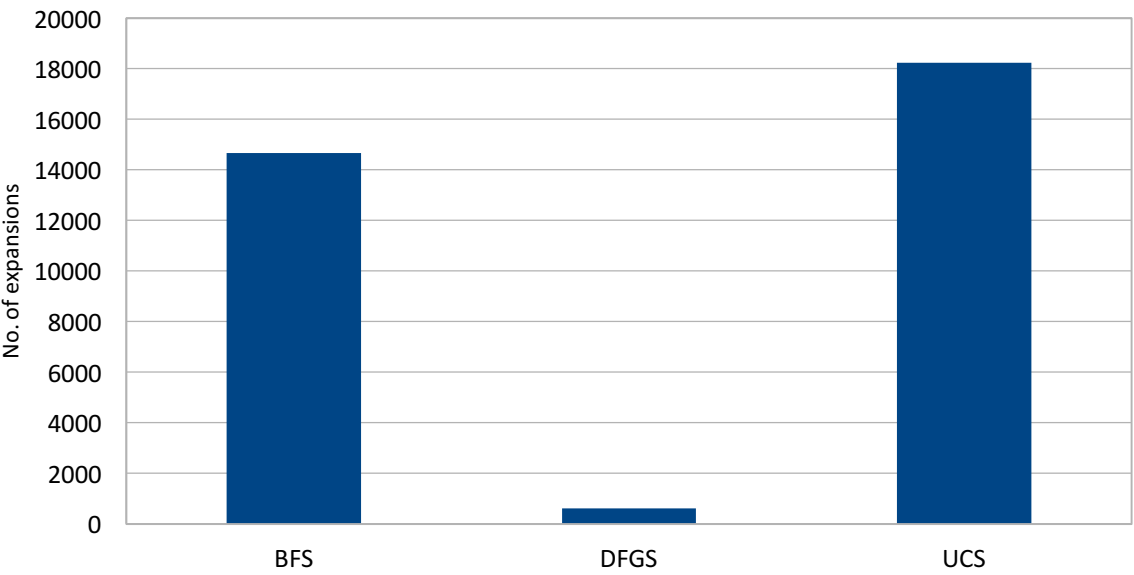
Problem 1



Problem 2

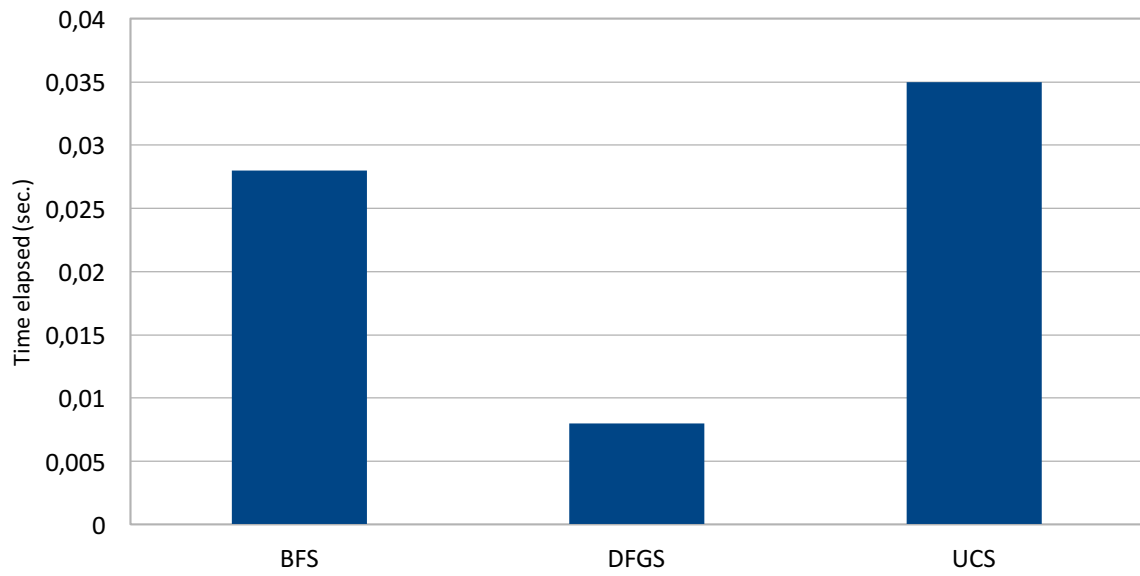


Problem 3

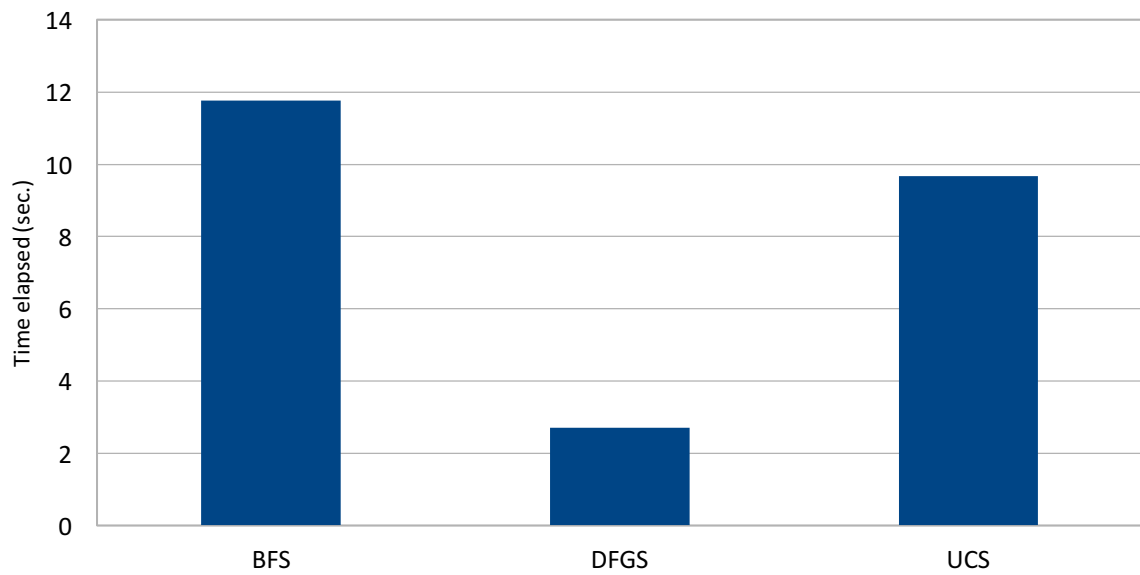


Execution time charts

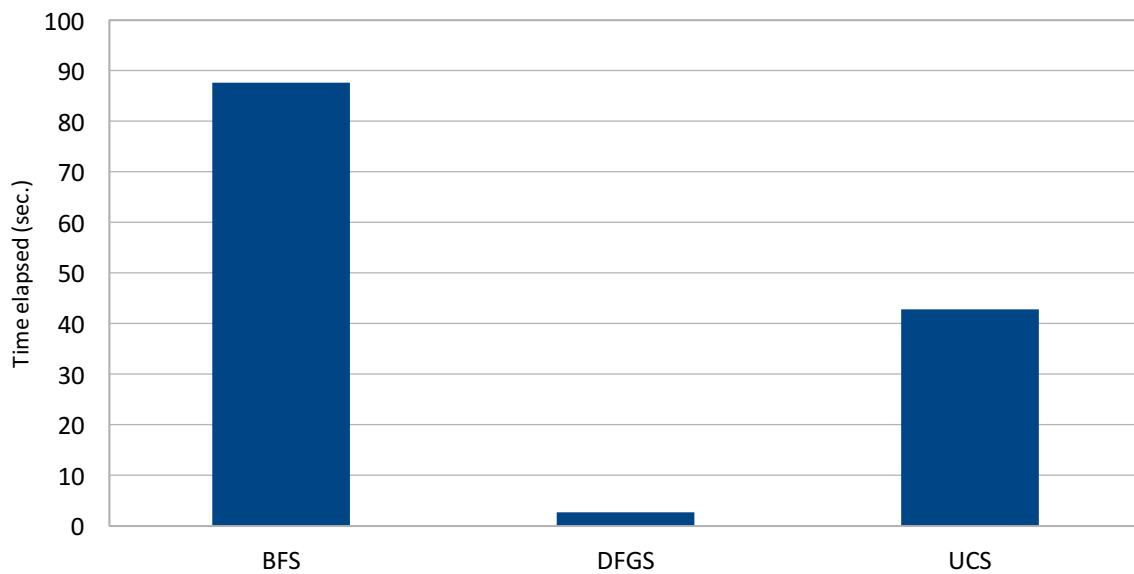
Problem 1



Problem 2



Problem 3



Analysis

Columns explanation

“**Time elapsed**” column indicates the speed of the search – the less is better. “**No. of expansions**” indicates the memory consumption – how often a search node was expanded, the less is better. “**Plan length**” indicates optimality – the optimal plan allows a goal achieving in minimal number of steps.

Optimal plan

Optimal plans for the problem are following:

- Problem 1. The plan has a length of 6:

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

- Problem 2. The plan has a length of 9:

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

- Problem 3. The plan has a length of 12:

```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

Searches analysis

All applied searches do not use any heuristic for a next search node selection.

Depth First Graph Search is a fastest search and uses much less memory comparing to other searches. While Breadth First Search and Uniformed Cost Search produced an optimal solution, DFGS never achieved it.

Breadth First Search and **Uniformed Cost Search** both produced optimal solutions for the problems. For all problems **BFS** required less memory but was slower than **UCS** on more complex problems.

Recommendation

I do not recommend using of **Depth First Graph Search** since it didn't produce an optimal result. To me it is a KO criteria for a search.

Generally I recommend using of **Uniformed Cost Search** because it produced optimal solution, was faster than **Breadth First Search** on more complex problems and consumed only **20-30%** more memory.

A* heuristic search

I experimented with 2 heuristics for A* search:

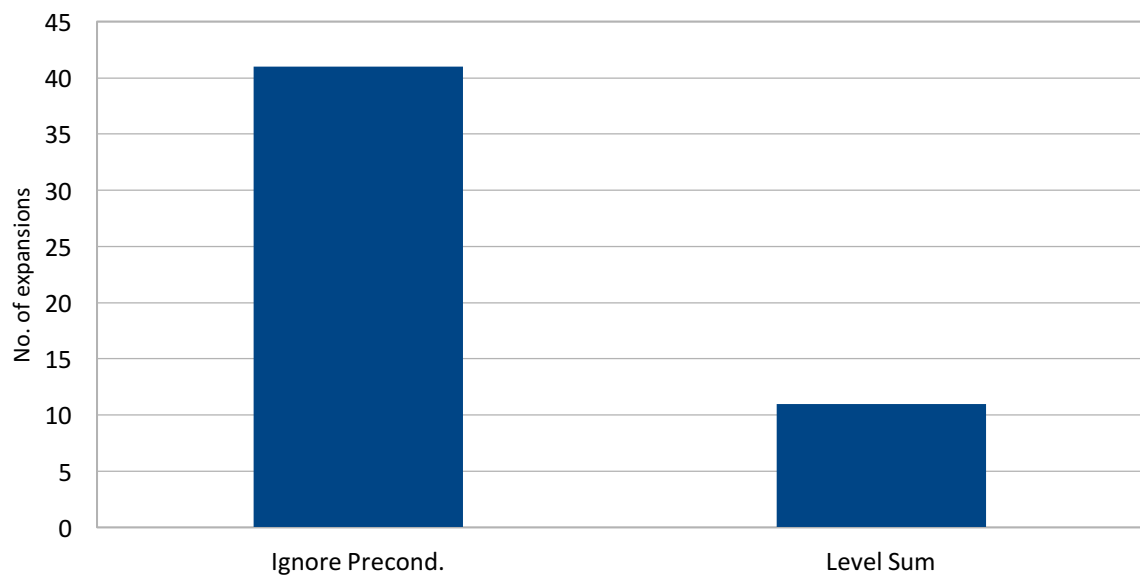
- **Ignore Preconditions**, where we relax a problem by ignoring preconditions of actions.
- **Level Sum**, where we follow the subgoal independence assumption, returning the sum of level costs of the goals.

Problem	Heuristic	No. of expansions	No. of goal tests	Plan length	Time elapsed (sec.)	Is optimal
P1	Ignore Precond.	41	43	6	0.028	Yes

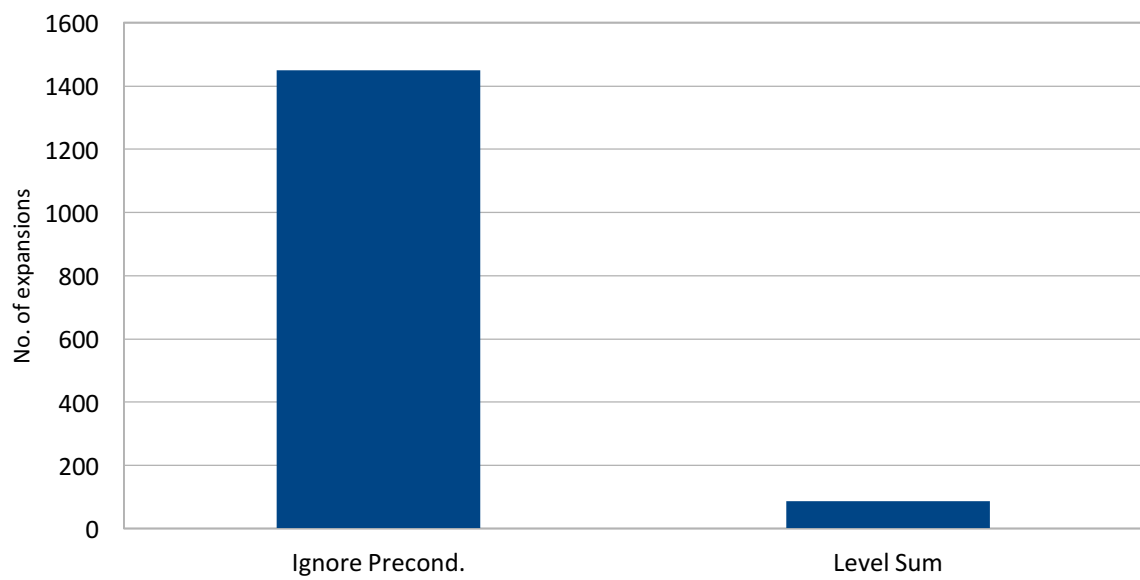
Problem	Heuristic	No. of expansions	No. of goal tests	Plan length	Time elapsed (sec.)	Is optimal
P1	Level Sum	11	13	6	0.675	Yes
P2	Ignore Precond.	1450	1452	9	3.01	Yes
P2	Level Sum	86	88	9	61.425	Yes
P3	Ignore Precond.	5040	5042	12	12.375	Yes
P3	Level Sum	318	320	12	303.83	Yes

Memory consumption charts

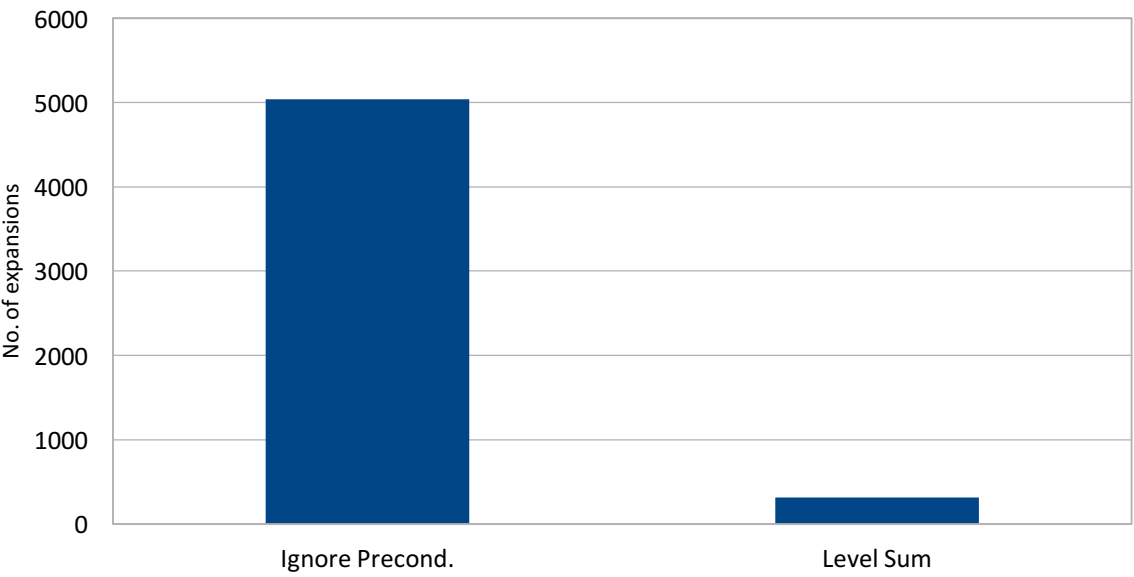
Problem 1



Problem 2

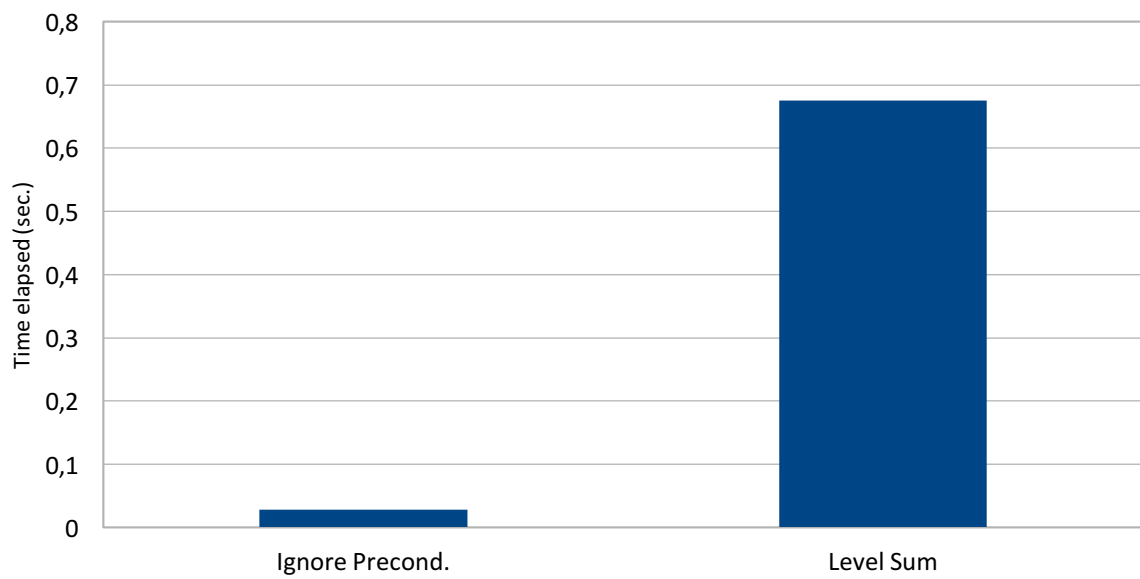


Problem 3

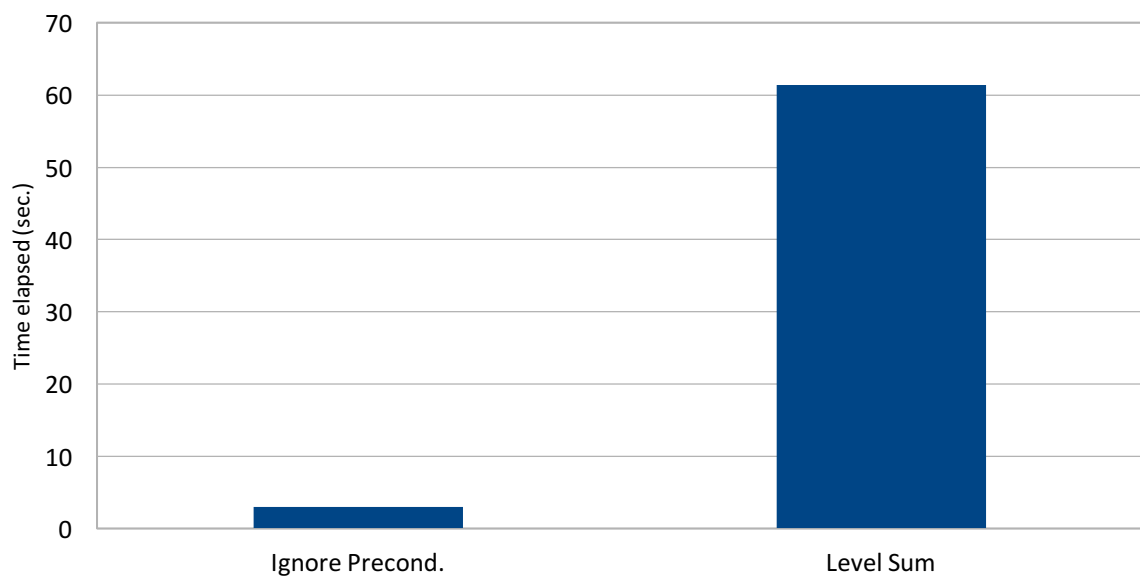


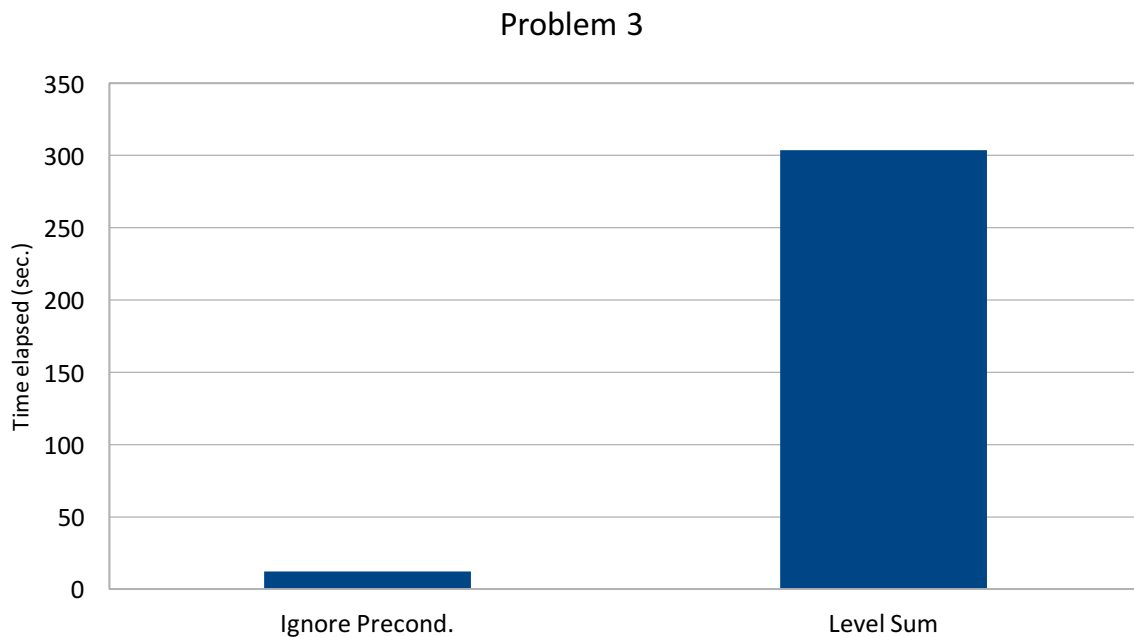
Execution time charts

Problem 1



Problem 2





Analysis

Here we have the same metrics as for non-heuristic searches, please see above the column explanation and optimal plans for the problems.

Comparing heuristics

A* search produced optimal solution with both heuristics, significant differences were memory consumption and speed. With “**Ignore Preconditions**” heuristic the A* search was **order 10 faster** than with “**Level Sum**”. From the other side, “**Level Sum**” allowed **order of 10 memory saving**.

Recommendation for heuristic

Depending on the problem and requirements I recommend:

- Using of “**Ignore Precondition**” heuristic if the speed matters and A* search do not result “Out of Memory” exception.
- Using of “**Level Sum**” in a case of complex problems where a great memory consumption is expected and a problem can be decomposed well.

Conclusion

Comparing uniformed non-heuristic searches with A* search that uses heuristics I recommend using the latter, because of memory consumption and running time. My favorite is **A* search with “Ignore Precondition” heuristic** for the following reasons:

- It found optimal solutions.
- It was fastest among considered searches that produced optimal results.
- It had approx. factor 3 of memory saving comparing to non-heuristic searches, though its memory consumption was not the best.