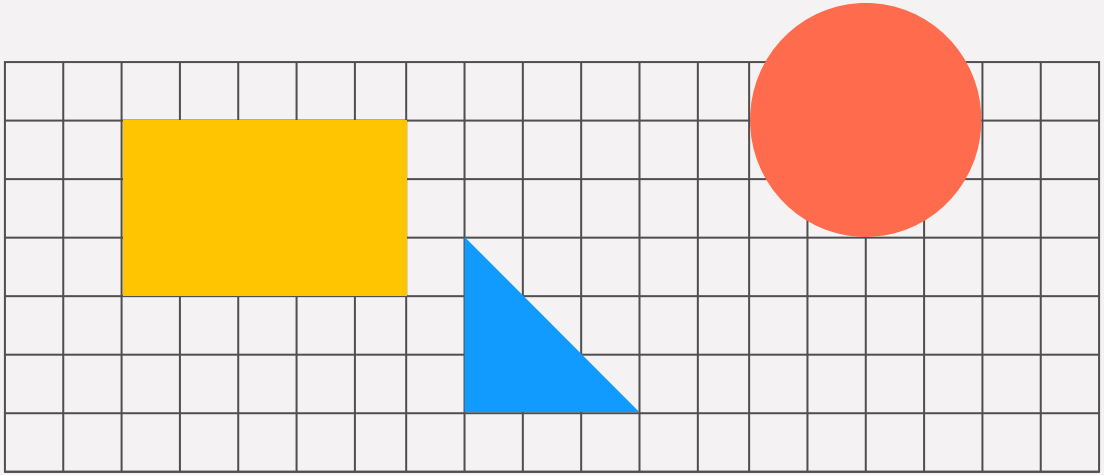
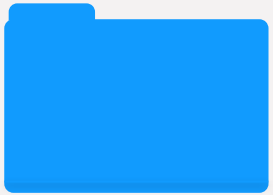
Wedding Travel Agency

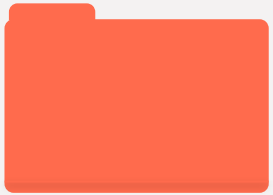
Planning & Reasoning project
Fabio Massimo Ercoli - 802397
March 2025



Domain



Planning



Heuristics

IndiGolog

Domain

- Travel agency - specialized in wedding trips.
- According to the budget and time constraints provided by the customers, we will propose a wedding trip to the future spouses.
- The goal is to visit attractions (places of interests).
- For each day they have a limited amount of hours that they can spend to travel and to visit attractions. After that they need to rest in a hotel.

move

to a **PLACE**

visit

an **ATTRACTION**

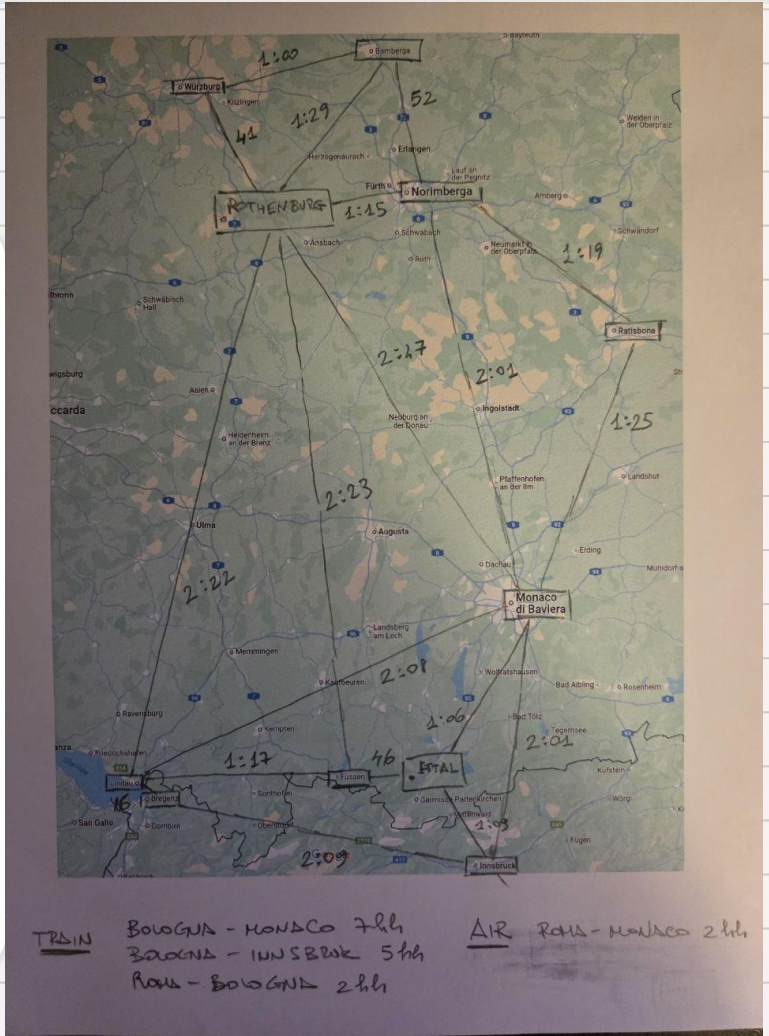
rest

at an **HOTEL**

Space model

- **Place**: denotes a municipality or a small geographical area.
- An **Hotel** or an **attraction** belongs to some place.

- The movement between items in the same place are modeled as **free** movements.
- To move from one place to another, an itinerary, called **travel** needs to exist.
- There are different kind of travels (car, air, train).



Cost & time model

The basic idea is that the customers can recharge themselves spending a night in a hotel. The day after they are ready to move and visit again!

- **Move** to a **place** using a **travel** has a **cost**. Some are more expensive, for instance air travels, other less expensive, for instance train travels.
- **Visit** an **attraction** has a **cost**, that can be also zero (for free attractions).
- **Rest** at an **hotel** has a **cost**.

- **Move** to a **place** using a **travel** takes some **time**. Some are faster, for instance air travels, other slower, for instance train travels.
- **Visit** an **attraction** takes some **time**.
- **Rest** at an **hotel**, spending a night, allows to reset the **time** to the maximum value: *day_activity_minutes*.

In general we want **minimize cost** and **time**, **maximizing** the **visited attractions**.

Tips

PDDL

```
(:types
        place attraction hotel travel - object
        train_travel air_travel car_travel - travel
)
(:predicates
        (at ?p - place)
        (connect ?t - travel ?from ?to - place)
        (hotel_at ?h - hotel ?p - place)
        (attraction_at ?a - attraction ?p - place)
        (visited ?a - attraction)
)
(:functions
        (time_travel ?t - travel)
        (time_attraction ?a - attraction)
        (cost_travel ?t - travel)
        (cost_attraction ?a - attraction)
        (cost_hotel ?h - hotel)
        (day_activity_minutes)
        (max_day_activity_minutes)
        (days)
        (attractions_visited)
        (total-cost)
)
```

Bolded: not-static, they may change from one state to another

types

:types

predicates

:predicates

functions

:functions

PDDL

```
(:action move
 :parameters (?t - travel ?from ?to - place)
 :precondition (and
  (at ?from)
  (or (connect ?t ?from ?to) (connect ?t ?to ?from))
  (<= (+ (day_activity_minutes) (time_travel ?t)) (max_day_activity_minutes)))
 :effect (and
  (increase (total-cost) (cost_travel ?t))
  (increase (day_activity_minutes) (time_travel ?t))
  (not (at ?from))
  (at ?to)))

(:action visit
 :parameters (?a - attraction ?p - place)
 :precondition (and
  (not (visited ?a))
  (at ?p)
  (attraction_at ?a ?p)
  (<= (+ (day_activity_minutes) (time_attraction ?a)) (max_day_activity_minutes)))
 :effect (and
  (increase (total-cost) (cost_attraction ?a))
  (increase (day_activity_minutes) (time_attraction ?a))
  (visited ?a)
  (increase (attractions_visited) 1)))
```

move

to a **PLACE**

visit

an **ATTRACTION**

rest

at an **HOTEL**

PDDL

```
(:action rest
 :parameters (?h - hotel ?p - place)
 :precondition (and
  (at ?p)
  (hotel_at ?h ?p))
 :effect (and
  (increase (total-cost) (cost_hotel ?h))
  (assign (day_activity_minutes) 0)
  (increase (days) 1)))
```

From here starts the problem file...
common to all problems

```
(:objects
  Bamberg Nurnberg Regensburg ... - place
  HotelEuropaBamberg ParkInnNurnberg IbisStylesRegensburg ... - hotel
  NeuschwansteinCastle EttalAbbey SchlossLinderhof - attraction
  RomeMunchen - air_travel
  RomeBologna BolognaInnsbruck BolognaMunchen - train_travel
  WurzburgBamberg WurzburgRothenburg BambergRothenburg ... - car_travel
)
(:init
 (hotel_at HotelEuropaBamberg Bamberg)
 (= (cost_hotel HotelEuropaBamberg) 89)
 (hotel_at ParkInnNurnberg Nurnberg)
 (= (cost_hotel ParkInnNurnberg) 80)
 ...
```

move

to a **PLACE**

visit

an **ATTRACTION**

rest

at an **HOTEL**

PDDL

```
(:init
  ...
  (attraction_at NeuschwansteinCastle Fussen)
  (= (time_attraction NeuschwansteinCastle) 240)
  (= (cost_attraction NeuschwansteinCastle) 21)
  (attraction_at EttalAbbey Ettal)
  (= (time_attraction EttalAbbey) 60)
  (= (cost_attraction EttalAbbey) 0)
  ...
  (connect RomeMunchen Rome Munchen)
  (= (time_travel RomeMunchen) 120)
  (= (cost_travel RomeMunchen) 200)
  (connect RomeBologna Rome Bologna)
  (= (time_travel RomeBologna) 120)
  (= (cost_travel RomeBologna) 67)
  ...
  (= (day_activity_minutes) 0)
  (= (max_day_activity_minutes) 600)
  (= (days) 0)
  (= (attractions_visited) 0)
)

(:metric minimize (- (total-cost) (* 100 (attractions_visited))))
```

hotel

has a **COST**

travel

has a **COST** and a **TIME**

attraction

has a **COST** and a **TIME**

Problem 1: Find the path between 2 attractions

```
(:init
  (at Bamberg)
  ...
)

(:goal (and
  (at Ettal)
  (visited EttalAbbey)
  (visited BambergOldTown)
))

(:metric minimize (- (total-cost) (*
100 (attractions_visited))))
```

We want to find a series of moves from Bamberg to Ettal. Also we want to visit one attraction at the start place and one attraction at the end place.
Minimizing the cost, with a little bonus for each more attraction visited.

PLANNERS (Expressive Numeric Heuristic Search Planner)

ENHSP optimal	ENHSP optimal	ENHSP SAT
blind	**HADD**	**HADD**

HEURISTICS

Problem 1: Find the path between 2 attractions

(visit BambergOldTown Bamberg)
(move BambergNurnberg Bamberg Nurnberg)
(visit ImperialCastleOfNuremberg Nurnberg)
(move NurnbergMunchen Nurnberg Munchen)
(move EttalMunchen Munchen Ettal)
(visit EttalAbbey Ettal)

optimal blind

(visit BambergOldTown Bamberg)
(move BambergNurnberg Bamberg Nurnberg)
(move NurnbergMunchen Nurnberg Munchen)
(move EttalMunchen Munchen Ettal)
(visit EttalAbbey Ettal)

optimal hadd

(move BambergNurnberg Bamberg Nurnberg)
(move NurnbergMunchen Nurnberg Munchen)
(move EttalMunchen Munchen Ettal)
(visit EttalAbbey Ettal)
(rest KlosterhotelEttal Ettal)
(move EttalMunchen Ettal Munchen)
(move NurnbergMunchen Munchen Nurnberg)
(move BambergNurnberg Nurnberg Bamberg)
(visit BambergOldTown Bamberg)
(move BambergNurnberg Bamberg Nurnberg)
(move NurnbergMunchen Nurnberg Munchen)
(move EttalMunchen Munchen Ettal)

sat hadd

Problem 1	optimal blind	optimal hadd	sat hadd
Plan-Length	6	5	12
Metric (Search)	24.0	24.0	215.0
Planning Time (msec)	101	25	54
Heuristic Time (msec)	0	12	39
Search Time (msec)	99	24	53
Expanded Nodes	528	8	138
States Evaluated	1973	42	733
Number of Dead-Ends detected	0	0	15
Number of Duplicates detected	637	1	129

Problem 2: Given a series of attractions, suggest an itinerary

```
(:init
  (at Bamberg)
  ...
)

(:goal (and
  (at Ettal)
  (visited EttalAbbey)
  (visited BambergOldTown)
  (visited WurzburgResidence)
  (visited AltesRathausRegensburg)
))

(:metric minimize (- (total-cost) (*
100 (attractions_visited))))
```

We want to visit the Ettal abbey, the old town of Bamberg, the Wurzburg residence palace and the Regensburg City Hall.
Starting from Bamberg, ending to Ettal.
Minimizing the cost, with a little bonus for each more attraction visited.

PLANNERS (Expressive Numeric Heuristic Search Planner)

ENHSP optimal	ENHSP optimal	ENHSP SAT
blind	**HADD**	**HADD**

HEURISTICS

Problem 2: Given a series of attractions, suggest an itinerary

(visit BambergOldTown Bamberg)
(move WurzburgBamberg Bamberg Wurzburg)
(visit WurzburgResidence Wurzburg)
(move WurzburgRothenburg Wurzburg Rothenburg)
(visit RothenburgObDerTauber Rothenburg)
(rest HotelRothenburgerHof Rothenburg)
(move NurnbergRothenburg Rothenburg Nurnberg)
(move NurnbergRegensburg Nurnberg Regensburg)
(visit AltesRathausRegensburg Regensburg)
(move RegensburgMunchen Regensburg Munchen)
(move EttalMunchen Munchen Ettal)
(visit EttalAbbey Ettal)

optimal blind

(visit BambergOldTown Bamberg)
(move WurzburgBamberg Bamberg Wurzburg)
(visit WurzburgResidence Wurzburg)
(move WurzburgRothenburg Wurzburg Rothenburg)
(rest HotelRothenburgerHof Rothenburg)
(visit RothenburgObDerTauber Rothenburg)
(move NurnbergRothenburg Rothenburg Nurnberg)
(move NurnbergRegensburg Nurnberg Regensburg)
(visit AltesRathausRegensburg Regensburg)
(move RegensburgMunchen Regensburg Munchen)
(move EttalMunchen Munchen Ettal)
(visit EttalAbbey Ettal)

optimal hadd

(move BambergNurnberg Bamberg Nurnberg)
(move NurnbergRegensburg Nurnberg Regensburg)
(visit AltesRathausRegensburg Regensburg)
(move RegensburgMunchen Regensburg Munchen)
(move EttalMunchen Munchen Ettal)
(visit EttalAbbey Ettal)
(move EttalMunchen Ettal Munchen)
(rest BoutiqueHotelMunchen Munchen)
(move RothenburgMunchen Munchen Rothenburg)
(move WurzburgRothenburg Rothenburg Wurzburg)
(visit WurzburgResidence Wurzburg)
(move WurzburgBamberg Wurzburg Bamberg)
(rest HotelEuropaBamberg Bamberg)
(visit BambergOldTown Bamberg)
(move BambergNurnberg Bamberg Nurnberg)
(move NurnbergMunchen Nurnberg Munchen)
(move EttalMunchen Munchen Ettal)

sat hadd

Problem 2	optimal blind	optimal hadd	sat hadd
Plan-Length	12	12	17
Metric (Search)	110.0	110.0	345.0
Planning Time (msec)	2373	473	39
Heuristic Time (msec)	1	345	29
Search Time (msec)	2372	471	37
Expanded Nodes	32876	18486	51
States Evaluated	67328	21699	350
Number of Dead-Ends detected	0	368	3
Number of Duplicates detected	45969	24966	30

Problem 3: Provide a complete travel proposal

```
(:init
  (at Rome)
  ...
)

(:goal (and
  (at Rome)
  (>= (attractions_visited) 7)
  (<= (days) 4)
))

(:metric minimize (- (total-cost) (* 100 (attractions_visited))))
```

Starting from Rome and ending to Rome, we want to propose a complete travel solution:

- Visiting at 7 attractions of Bayern
- Spending at most 4 nights at the hotel
- Minimizing the cost
- With a little bonus for each more attraction visited

PLANNERS (Expressive Numeric Heuristic Search Planner)

ENHSP optimal	ENHSP optimal	ENHSP SAT
blind	**HADD**	**HADD**

HEURISTICS

Problem 3: Provide a complete travel proposal

----------Time: 1424s ; Expanded Nodes: 17448122 (Avg-Speed 12252.0 n/s); Evaluated States: 24244450
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at com.carrotsearch.hppc.DoubleArrayList.clone(DoubleArrayList.java:435)
at com.hstairs.ppmajal.PDDLProblem.PDDLState.<init>(PDDLState.java:51)
at com.hstairs.ppmajal.PDDLProblem.PDDLState.clone(PDDLState.java:121)
at com.hstairs.ppmajal.PDDLProblem.PDDLState.clone(PDDLState.java:39)
at com.hstairs.ppmajal.PDDLProblem.PDDLProblem$stateIterator.hasNext(PDDLProblem.java:1519)
at com.hstairs.ppmajal.search.WAStar.search(WAStar.java:129)
at com.hstairs.ppmajal.PDDLProblem.PDDLPlanner.plan(PDDLPlanner.java:85)
at planners.ENHSP.search(ENHSP.java:545)
at planners.ENHSP.planning(ENHSP.java:209)
at main.main(main.java:31)

optimal blind

(move RomeBologna Rome Bologna)
(visit PiazzaMaggiore Bologna)
(move BolognaInnsbruck Bologna Innsbruck)
(rest YouthHostelInnsbruck Innsbruck)
(move EttalInnsbruck Innsbruck Ettal)
(visit EttalAbbey Ettal)
(visit SchlossLinderhof Ettal)
(move FussenEttal Ettal Fussen)
(visit HohesSchloss Fussen)
(move RothenburgFussen Fussen Rothenburg)
(rest HotelRothenburgerHof Rothenburg)
(visit RothenburgObDerTauber Rothenburg)
(move NurnbergRothenburg Rothenburg Nurnberg)
(visit ImperialCastleOfNuremberg Nurnberg)
(move NurnbergMunchen Nurnberg Munchen)
(visit MunichResidence Munchen)
(move RomeMunchen Munchen Rome)

optimal hadd

(move RomeBologna Rome Bologna)
(visit PiazzaMaggiore Bologna)
(rest HotelPalaceBologna Bologna)
(move BolognaInnsbruck Bologna Innsbruck)
(rest YouthHostelInnsbruck Innsbruck)
(move BregenzInnsbruck Innsbruck Bregenz)
(visit Pfander Bregenz)
(move LindauBregenz Bregenz Lindau)
(move LindauFussen Lindau Fussen)
(rest HotelSchlosskroneFussen Fussen)
(visit HohesSchloss Fussen)
(move FussenEttal Fussen Ettal)
(visit EttalAbbey Ettal)
(visit SchlossLinderhof Ettal)
(rest KlosterhotelEttal Ettal)
(move EttalMunchen Ettal Munchen)
(visit MunichResidence Munchen)
(visit SchlossNymphenburg Munchen)
(move RomeMunchen Munchen Rome)

sat hadd

Problem 3	optimal blind	optimal hadd	sat hadd
Plan-Length	FAILS	17	19
Metric (Search)	-	540.0	852.0
Planning Time (msec)	1424	141534	3459
Heuristic Time (msec)	-	110322	3091
Search Time (msec)	1424	141532	3458
Expanded Nodes	17448122	5590621	62377
States Evaluated	24244450	7406733	232061
Number of Dead-Ends detected	-	180195	33896
Number of Duplicates detected	-	10682477	54162

```
place(P) :- member(P, [bamberg, nurnberg, regensburg, ..., ettal]).
hotel(H) :- member(H, [hotelEuropaBamberg, ..., hotelPalaceBologna]).
attraction(A) :- member(A, [neuschwansteinCastle, ..., piazzaMaggiore]).

% travel(from, to, time, cost, type)

travel(rome, munchen, 120, 200, air).
travel(munchen, rome, 120, 200, air).
...

% hotel_info(hotel, place, cost)

hotel_info(hotelEuropaBamberg, bamberg, 89).
hotel_info(parkInnNurnberg, nurnberg, 80).
...

% attraction_info(attraction, place, time, cost)

attraction_info(neuschwansteinCastle, fussen, 240, 21).
attraction_info(ettalAbbey, ettal, 60, 0).
```

Using **indigolog_plain** interpreter, no exogenous events, no concurrency, ...

non fluents

Static entities

like relational DB

Enumerated explicitly

Closed world assumption

Facts not present false

```
% Relational Fluents

prim_fluent(at(P)) :- place(P).
prim_fluent(visited(A)) :- attraction(A).

% Functional Fluents

prim_fluent(day_activity_minutes).
prim_fluent(days).
prim_fluent(attractions_visited).
prim_fluent(total_cost).

% Init (S0)

initially(at(P), true) :- =(P, rome).
initially(at(P), false) :- place(P), \+ initially(at(P), true).
initially(day_activity_minutes, 0).
initially(days, 0).
initially(attractions_visited, 0).
initially(total_cost, 0).
```

fluents

Situation dependent

initial situation

Fluents at S0

Closed world assumption

Facts not present false

```
prim_action(move(P)) :- place(P).

poss(move(To), and(
        and(at(From), travel(From, To, Time, _, _)),
                =<(+(day_activity_minutes, Time), 600)
)).

causes_val(move(To), at(From), false, and(
        at(From),
        travel(From, To, _, _, _)
)).
causes_val(move(To), at(To), true, true).
causes_val(move(To), day_activity_minutes, N, and(
        and(at(From),
        travel(From, To, Time, _, _)),
        N is day_activity_minutes + Time
)).
causes_val(move(To), total_cost, N, and(
        and(at(From),
        travel(From, To, _, Cost, _)),
        N is total_cost + Cost
)).
```

move

to a **PLACE**

visit

an **ATTRACTION**

rest

at an **HOTEL**

```
prim_action(visit(A)) :- attraction(A).
prim_action(rest(H)) :- hotel(H).

poss(visit(A), and( and(neg(visited(A)),at(P)),
        and(attraction_info(A, P, Time, _),
        =<(+(day_activity_minutes, Time), 600))
)).
poss(rest(H), and(at(P), hotel_info(H, P, _))).

causes_val(visit(A), day_activity_minutes, N,
        and(attraction_info(A, _, Time, _), N is day_activity_minutes + Time)).
causes_val(visit(A), total_cost, t, N,
        and(attraction_info(A, _, _, Cost), N is total_cost + Cost)).
causes_val(visit(A), visited(A), true, true).
causes_val(visit(_), attractions_visited, N, N is attractions_visited + 1).

causes_val(rest(H), total_cost, N, and(
        hotel_info(H, _, Cost), N is total_cost + Cost)).
causes_val(rest(_), day_activity_minutes, 0, true).
causes_val(rest(_), days, N, N is days + 1).
```

move

to a **PLACE**

visit

an **ATTRACTION**

rest

at an **HOTEL**

```
proc(attraction_not_visited(P),
        some(A, and(attraction_info(A, P, _, _), neg(visited(A))))
).

proc(no_attraction(P),
        neg(some(A, attraction_info(A, P, _, _)))
).

proc(smart_move(P), [?(attraction_not_visited(P)), move(P)]).
proc(neutral_move(P), [?(no_attraction(P)), move(P)]).

proc(pi_visit, pi(a, visit(a))).
proc(pi_smart_move, pi(to, smart_move(to))).
proc(pi_neutral_move, pi(to, neutral_move(to))).
proc(pi_move, pi(to, move(p))).
proc(pi_rest, pi(h, rest(h))).

proc(pi_any, ndet(ndet(pi_visit, ndet(pi_smart_move,
pi_neutral_move)), ndet(pi_rest, pi_move))).
```

behavioural

From here **not declarative**

Non deterministically chose - giving a priority - backtracking:

1. Visit
2. Move to a place in which there is a non visited attraction
3. Move to a place in which no attractions are present
4. Move to a place in which all attractions are visited
5. Rest in a hotel

```
% single steps tests (for debug purpose!):
...
proc(control(test3), search([pi_any, pi_any, pi_any, pi_any])).
proc(control(test4), search(while(days < 2, pi_any))).

% Other procedures

proc(find_n_attractions(N, D), while(attractions_visited < N,
        [?(days =< D), pi_any]
)).

proc(visit_saving_costs(A1, A2, A3, C), while(
        or(or(neg(visited(A1)), neg(visited(A2))), neg(visited(A3))),
        [?(total_cost =< C), pi_any]
)).

proc(visit_saving_times(A1, A2, A3, D), while(
        or(or(neg(visited(A1)), neg(visited(A2))), neg(visited(A3))),
        [?(days =< D), pi_any]
)).
```

Visit *n* attractions

Within amount of days

Visit set of attractions

Within amount of time

Visit set of attractions

Within amount of days

```
proc(control(problem1), search(find_n_attractions(12, 4))).

Executing controller: *problem1*
move(munchen) visit(munichResidence)
visit(schlossNymphenburg) move(ettal) visit(ettalAbbey)
visit(schlossLinderhof) rest(klosterhotelEttal) move(fussen)
visit(neuschwansteinCastle) visit(hohesSchloss) move(lindau)
visit(lindauHafen) rest(hotelEngelLindau) move(rothenburg)
visit(rothenburgObDerTauber) move(wurzburg)
visit(wurzburgResidence) move(bamberg) move(nurnberg)
move(bamberg) rest(hotelEuropaBamberg)
visit(bambergOldTown) move(nurnberg)
visit(imperialCastleOfNuremberg) move(regensburg)
visit(altesRathausRegensburg)

26 actions.
true.
```

Problem 1

Given a hotel, find all the attractions I can reach there in a given amount of time, using
airplanes, trains or renting a car.

```
proc(control(problem2),
        search(visit_saving_costs(neuschwansteinCastle,
rothenburgObDerTauber, lindauHafen, 500))).

Executing controller: *problem2*
move(munchen) visit(munichResidence)
visit(schlossNymphenburg) move(ettal) visit(ettalAbbey)
visit(schlossLinderhof) rest(klosterhotelEttal) move(fussen)
visit(neuschwansteinCastle) visit(hohesSchloss) move(lindau)
move(bregenz) visit(pfander) rest(ibisBregenz) move(lindau)
visit(lindauHafen) move(rothenburg)
visit(rothenburgObDerTauber)

18 actions.
true.
```

Problem 2

Determine If we can visit 3 given attractions with a given amount of money.

```
proc(control(problem3),
        search(visit_saving_times(neuschwansteinCastle,
rothenburgObDerTauber, lindauHafen, 2))).

Executing controller: *problem3*
move(munchen) visit(munichResidence)
visit(schlossNymphenburg) move(ettal) visit(ettalAbbey)
visit(schlossLinderhof) rest(klosterhotelEttal) move(fussen)
visit(neuschwansteinCastle) visit(hohesSchloss) move(lindau)
visit(lindauHafen) rest(hotelEngelLindau) move(rothenburg)
visit(rothenburgObDerTauber)

15 actions.
true.
```

Problem 3

Determine If we can visit 3 given attractions with a given amount of time.

Thank you

Planning:
https://github.com/fax4ever/ai-play/tree/main/wedding-travel-agency

Reasoning:
https://github.com/fax4ever/indigolog/tree/main/examples/wedding_travel_agency

Email

fax.ercoli@gmail.com

GitHub

https://github.com/fax4ever

Linkedin

https://www.linkedin.com/in/fabioercoli/