Department of Computer Science
Technical University of Cluj-Napoca

Introduction to Artificial Intelligence

Laboratory activity 2019-2020

Project title: Conformant planning
Tool: Conformant-FF

Name: Rusu Carla-Maria
Group: 30431
Email: carla.rusu9@gmail.com

Assoc. Prof. dr. eng. Anca Marginean
Anca.Marginean@cs.utcluj.ro

Contents

Chapter 1

Installing the tool (W_2)

The steps required to install the tool are:

1. access the homepage of the tool: https://fai.cs.uni-saarland.de/hoffmann/cff.html;
2. download the linux executable for conformant-ff and the corresponding benchmark problems;
3. unzip the archives and place them in the same folder;
4. open a terminal and run the executable with one of the benchmark problems, mentioning both the domain file and the problem description;

The general commands of running the executable is:
`./Conformant-FF -o [domain-path] -f [problem-pddl-path]`
An example of running a problem using the executable is given below:
`./Conformant-FF -o ./cff-tests/cube/domain -f ./cff-tests/cube/p3-1.pddl`
In case something goes wrong:

1. unzip the files in the `HOME` directory;
2. try renaming the executable;
3. be aware of case sensitive commands

The tool was run on Ubuntu 18.04.2 Virtual Machine (64-bit)

Chapter 2

Running and understanding examples (W_3)

Conformant-FF is a tool used for solving conformant planning problems which represent a branch of AI planning problems.

Classical planning Classical planning deals with determining all the small tasks that must be carried out in order to accomplish a goal.

It involves choosing a sequence of actions that will transform the state of the world at each step so that the goal will be satisfied.

The system contains atomic facts and actions that modify the system's state.

Planning can be found in everyday life events, such as the task of buying a gallon of milk. It sounds simple at first, yet sub-problems can be extracted:

1. Goal: buy a gallon of milk
2. Tasks involved: obtain keys, obtain wallet, start car, drive to store, find and obtain milk, purchase milk, etc.
3. Constraints: you must obtain your keys and wallet before driving to the store and you must obtain the milk before purchasing it

Conformant planning Conformant planning, as opposed to classical planning, is the task of generating plans given uncertainty about the initial state and action effects, and without any sensing capabilities during plan execution.

The plan should be successful regardless of which particular initial world we start from.

Conformant planning can be transformed into a search problem in belief space, the space whose elements are sets of possible worlds.

To be exact, can be viewed as the problem of finding a path in belief space.

In general, a belief state could require an exponentially large (in number of state variables) description.

Conformant Planning problem

```
[P,A,I,G]
P: action sequence that leads to a belief state s
A: action set
I: initial world state
G: goal world state
```

Conformant-FF was evaluated on a variety of domains, such as traditional domains: Bomb-in-the-toilet, Ring, Cube, Omelette, and Safe, as well as modified classical domains: Blocksworld, Logistics, and Grid.

The inputs are a file describing the domain and another describing the problem's task.

2.1 Example 1: Bomb-in-the-toilet

The simplest benchmark problem was chosen for the first example.
The domain code:

```
 (define (domain bomb)
(:predicates (bomb ?x)
             (toilet ?x)
             (armed ?x)
             (clogged ?x))

(:action dunk
    :parameters  (?bomb ?toilet)
    :precondition (and (bomb ?bomb) (toilet ?toilet)
                       (not (clogged ?toilet)))
    :effect (and (when (armed ?bomb) (not (armed ?bomb)))
                 (clogged ?toilet)))

(:action flush
    :parameters  (?toilet)
    :precondition (toilet ?toilet)
    :effect (when (clogged ?toilet) (not (clogged ?toilet))))

)
```

The domain and task are both written in LISP. The domain contains:

- the domain name;
- the predicates, i.e. the objects that make up the world;
- the actions, i.e. the events that modify the world state; actions can have preconditions and (conditional) effects;

The task code for p5-1.pddl:

```
(define (problem bomb-5-1)
(:domain bomb)
(:objects bomb1 bomb2 bomb3 bomb4 bomb5 toilet1)
(:init
(bomb bomb1)
(bomb bomb2)
(bomb bomb3)
(bomb bomb4)
(bomb bomb5)
(toilet toilet1)
(unknown (armed bomb1))
```

```
(unknown (armed bomb2))
(unknown (armed bomb3))
(unknown (armed bomb4))
(unknown (armed bomb5))
)
(:goal
(and
(not (armed bomb1))
(not (armed bomb2))
(not (armed bomb3))
(not (armed bomb4))
(not (armed bomb5))
)
)
)
```

The task contains:

- the problem name;
- the domain used;
- the objects defined;
- the initialisation of said objects and the related facts (known and unknown); the facts are evaluated as CNF (conjunctive normal forms);
- finally, the goal is given;

The output file:

```
 Cueing down from goal distance:    6 into depth [1][2]
                                    5            [1][2]
                                    4            [1][2]
                                    3            [1][2]
                                    2            [1]
                                    0

ff: found legal plan as follows

step    0: DUNK BOMB5 TOILET1
        1: FLUSH TOILET1
        2: DUNK BOMB4 TOILET1
        3: FLUSH TOILET1
        4: DUNK BOMB3 TOILET1
        5: FLUSH TOILET1
        6: DUNK BOMB2 TOILET1
        7: FLUSH TOILET1
        8: DUNK BOMB1 TOILET1


statistics:    0.00 seconds instantiating 6 easy, 0 hard action templates
               0.00 seconds reachability analysis, yielding 18 facts and 6
```

```
actions
0.00 seconds creating final representation with 12 relevant
facts (10 max U, 5 CNF max U)
0.00 seconds building connectivity graph
0.00 seconds (   0.00 pure) evaluating 19 states, to a max
depth of 2
0.00 seconds in DP for 0 RPG ini state implication checks
0.00 seconds in DP for 0 RPlan extract ini state implication
checks (0 lits removed)
0.00 seconds generating,    0.00 seconds encoding 29 state
transition base CNFs
0.00 seconds in DP solving 15 state transition CNFs
0.00 seconds checking for self-contradictions, including 0 DP
calls
0.00 seconds checking for stagnating states (0 hits),
including 0 DP calls
0.00 seconds altogether checking for dominated states making 0
comparisons (0 conformant, 0 hits),
     spending    0.00 seconds doing 0 DP calls
  15 total DP calls, 15 total UP calls,    0.00 sec membership
0.00 seconds for remaining searching duties
0.00 seconds total time (+    0.00 secs for CNF memory allocation)
```

The output contains:

- the graphs depths (how far each object is from the goal);
- the devised plan (if found), given step by step;
- the statistics used for benchmark purposes;

In this problem, a number of bombs and toilets are given without the knowledge of which bomb is armed. By throwing a bomb in a toilet, a possibly armed bomb gets disarmed. In order to throw another bomb in the same toilet, the toilet must be unclogged first by flushing it.
The outputted plan proposes to throw the bombs one by one in the toilet, flushing after each one. As such, all armed bombs will be disarmed.

Chapter 3

Understanding conceptual instrumentation (W_4)

Conformant planning poses two main issues:

How do we represent belief states efficiently?

- Small size desirable
- Ability to quickly detect goal satisfaction
- Ability to quickly detect which action is applicable

How can we generate good heuristic estimates? The use of heuristics will be detailed later.

Symbolic representations:

- Conjunctive Normal Form (Conjunction of Disjunctions)
  - In Boolean logic, a formula is in CNF if it is a conjunction (ANDs) of disjunctions (ORs)
  - (pvqvr) & (-pvwvd) & (-wvqvs)
- Binary Decision Diagrams
- Disjunctive Normal Form: (p&q&r) v (-p&w&d) v (-w&qs&)

In Conformant-FF the representation of world states is done using 2-CNF (a stronger formula than CNF) with a heuristic for FF (fast forward search). The implication graph of 2-CNF supports linear time reasoning.

For conformant planning, it suffices to know the propositions that are true in the intersection of the worlds contained in a belief state – only these propositions will be true no matter what initial world we start from.

STRIPS is a language used for formalizing actions. It contains facts (or state variables) that are either TRUE or FALSE (0,1) and actions.
Actions consist of:

- PRECONDITION

- The ADD=a1,a2,. . . ,aN list
- The DELETE=d1,d2,. . . ,dM list

Assume ADD and DELETE don't intersect. An action is possible if all the variables in the precondition have the value TRUE.
A goal is a set of state variables. A state is a goal if all the goals have the value TRUE in it.

STRIPS with conditional effects is used for the tool, more exactly:

- (A, I, G) -¿ action set, initial world state, goal world state
- World states w are sets of propositions
- Actions a are pairs (pre(a), E(a)); a set of preconditions and the effects
- A conditional effect E(a) is a triple (con(e), add(e), del(e)); condition, add, delete lists

Fast-forward planning systems rely on forward search in the state space, guided by a heuristic that estimates goal distances by ignoring delete lists.

The planning systems relax the planning problem by ignoring parts of the domain specification, i.e. the delete lists of all actions.

The effects of relaxation are: all actions only add new atoms to the state, but don't remove any, thus states only grow; as such, the problem is solved as soon as all goal states have been added by some actions.

Say we have an action that moves a robot from some point A to another point B.

The precondition contains a fact stating that the robot needs to be at location A for the action to be applicable.

After applying the action, the add list produces a fact stating the robot stands at location B, and the delete list removes the fact stating it stands at A.

In the relaxation, the delete is ignored, so the precondition fact is not removed; after executing the relaxed action, the robot is located at A and B simultaneously.

The obvious benefit is that the heuristic helps identify the successors of a search node that seem to be (and usually are) most helpful in getting to the goal.

Enforced hill-climbing is chosen as the search algorithm in hope to reach the goal by evaluating as few states as possible.

Facing a search state S, FF evaluates all of its direct successors. If none of those has a heuristic value than S, it goes one step further, i.e., search then looks at the successor's successors. If none of those two-step successors looks better than S, FF goes on to the three-step successors,

and so on.

The process terminates when a state S0 with better evaluation than S is found.

The path to S0 is then added to the current plan, and search continues with S0 as the new starting state.

Chapter 4

Project description (W_5)

Using the tool Conformant-FF developed by Ronen I. Brafman and Jorg Hoffmann, I have developed a disaster recovery planner. The application is meant to provide a plan in case of emergency situations that are often encountered in real life situations, such as terrorist attacks, floods, hurricanes, fires, tornadoes, tsunamis, etc.

The domain contains a vast range of predicates and actions.
The state of roads, civilians and buildings constitute known or unknown factors (their position, state, general health). Ambulances, fire brigades, resource teams (food and water providers) and other such organs can be deployed, based on their availability.
The actions will be directed to certain entities or interacting entities and will often have preconditions. For example: saving civilians from the rubble is only possible when the road to the site is intact and when an ambulance is not in use somewhere else.

A number of problems will be provided, to test the capabilities of the project. Different scenarios will be created, in order to push the tool to its limits (for certain problem tasks, a legal plan might not be found).

Chapter 5

Implementation details (W_9)

The project consists of two types of files: the domain, which contains the possible predicates and actions that can take place, and the problem file, which represents a certain instance of such predicates for which the tool must find a plan (a set of actions that lead invariably to the goal state).

5.1 The domain

The predicates

The domain contains the following predicates which can take the corresponding states:

- vehicles:
  1. helicopter
     - *available* or *not available*
  2. ambulance
     - *available* or *not available*
  3. firefighters
     - *available* or *not available*
  4. police
     - *available* or *not available*
  5. defuse-team
     - *available* or *not available*
  6. intervention
     - *available* or *not available*
- streets
  - *destroyed* or *not destroyed*
- civilians
  - *need-medical* or *not need-medical*
  - *need-resources* or *not need-resources*
  - *on (a certain) street*

- bombs
  - *armed* or *not armed*
  - *at (a certain) building*
- buildings
  - *under-siege* or *not not under-siege*
  - *on-fire* or *not on-fire*
  - *evacuated* or *not evacuated*
  - *on (a certain) street*

The actions

The set of actions is modeled after the parameter-(optional) precondition-effect archetype and is described in the domain as:

1. EVACUATE-BUILDING
   The parameters are a street, a building, an intervention team and a bomb.

   The precondition states that all parameters should be of their corresponding type and the intervention team should be available.

   The effect states that: if the intervention team was available, it no longer is (until returned); the building can be evacuated if and only if the bomb is at the building, the building is on the street given as parameter and the bomb is no longer armed (if it previously was).

2. SEND-DEFUSE-TEAM
   The parameters are a street, a building, a defuse team and a bomb.

   The precondition states that all parameters should be of their corresponding type and the defuse-team should be available.

   The effect states that: if the defuse team was available, it no longer is (until returned); the bomb can be disarmed if and only if the bomb is at the building, the building is on the street given as parameter and the building is no longer under siege (if it previously was).

3. SEND-POLICE
   The parameters are a street, a building and a police car.

   The precondition states that all parameters should be of their corresponding type and the police car should be available.

   The effect states that: if the police car was available, it no longer is (until returned); the building can be not under siege if and only if the building is on the street given as parameter and the building is no longer on fire (if it previously was).

4. SEND-AMBULANCE
   The parameters are a street, a civilian and an ambulance.

The precondition states that all parameters should be of their corresponding type, the ambulance should be available and the street should not be destroyed.

The effect states that: if the ambulance was available, it no longer is (until returned); the civilian will not need medical assistance if and only if the civilian is on the street given as parameter.

5. SEND-INTERVENTION
   The parameters are a street, a civilian and an intervention team.

   The precondition states that all parameters should be of their corresponding type, the intervention team should be available and the street should not be destroyed.

   The effect states that: if the intervention team was available, it no longer is (until returned); the civilian will not need resources if and only if the civilian is on the street given as parameter.

6. SEND-HELICOPTER
   The parameters are a street, a civilian and a helicopter.

   The precondition states that all parameters should be of their corresponding type, the helicopter should be available and the street should be destroyed (so that no other vehicle could access it).

   The effect states that: if the helicopter was available, it no longer is (until returned); the civilian will not need medical assistance or resourced if and only if the civilian is on the street given as parameter.

7. SEND-FIREFIGHTERS
   The parameters are a street, a building and a firefighter team.

   The precondition states that all parameters should be of their corresponding type and the firefighters should be available.

   The effect states that: if the firefighter team was available, it no longer is (until returned); the building will no longer be on fire if and only if the building is on the street given as parameter.

8. RETURN-VEHICLE
   The parameter is a vehicle.

   The precondition states that the vehicle should be either an ambulance, a police car, a helicopter, a firefighter team, an intervention team or a defuse team and that is should not be available.

   The effect states that: if the vehicle was not available, it now is.

These actions and predicates will be used in the problem description files to describe an instance of the world.

5.2 The problem description file

The objects and their initialization version 1

- streets : s0, s1
  - s0 destroyed
  - s1 not destroyed
- helicopters : h0
  - h0 available
- ambulances : a0
  - a0 available
- civilians : c0, c1, c2
  - c0, c1 need-medical
  - c0 need-resources
  - c2 need-medical or resources (unknown)
- firefighters : f0
  - f0 available
- buildings : b0, b1, b2
  - b0 on-fire
  - b0, b1 under-siege
  - b0 not-evacuated
- bombs : bo0, bo1
  - bo0, bo1 armed
- police cars : p0
  - p0 available
- defuse teams : dt0
  - dt0 available
- intervention teams : i0
  - i0 available
- other states (of interacting objects):
  - c0 at s0 or s1 (unknown)
  - c1 at s1
  - c2 at s1
  - b0 on s0 or s1 (unknown)

  - b1 on s1
  - b2 on s1
  - bo0 at b2
  - **bo1 at b0**

The difference between version 1 and version 2 is bolded.

The objects and their initialization version 2

- streets : s0, s1
  - s0 destroyed
  - s1 not destroyed
- helicopters : h0
  - h0 available
- ambulances : a0
  - a0 available
- civilians : c0, c1, c2
  - c0, c1 need-medical
  - c0 need-resources
  - c2 need-medical or resources (unknown)
- firefighters : f0
  - f0 available
- buildings : b0, b1, b2
  - b0 on-fire
  - b0, b1 under-siege
  - b0 not-evacuated
- bombs : bo0, bo1
  - bo0, bo1 armed
- police cars : p0
  - p0 available
- defuse teams : dt0
  - dt0 available
- intervention teams : i0
  - i0 available
- other states (of interacting objects):

- c0 at s0 or s1 (unknown)
- c1 at s1
- c2 at s1
- b0 on s0 or s1 (unknown)
- b1 on s1
- b2 on s1
- bo0 at b2
- **bo1 at b0 or b1 (unknown)**

The goal state

- (not (need-medical c0))
- (not (need-medical c1))
- (not (need-resources c2))
- (not (need-medical c2))
- (not (need-resources c0))
- (not (not-evacuated b0))
- (not (on-fire b0))
- (not (on-fire b1))
- (not (under-siege b1))
- (not (under-siege b0))
- (not (armed bo0))
- (not (armed bo1))

Chapter 6

Results (W_{11})

Version 1 output: Enforced hill-climbing plan found

```
    Cueing down from goal distance:   14 into depth [1]
                                      13            [1]
                                      12            [1][2]
                                      11            [1]
                                      10            [1][2]
                                       9            [1][2]
                                       8            [1]
                                       7            [1][2]
                                       6            [1][2][3]
                                       5            [1][2][3]
                                       4            [1][2]
                                       3            [1]
                                       2            [1]
                                       0

ff: found legal plan as follows

step    0: SEND-HELICOPTER S0 C0 H0
        1: SEND-DEFUSE-TEAM S1 B2 DT0 BO0
        2: SEND-INTERVENTION S1 C0 I0
        3: RETURN-VEHICLE I0
        4: SEND-INTERVENTION S1 C2 I0
        5: SEND-AMBULANCE S1 C0 A0
        6: RETURN-VEHICLE A0
        7: SEND-AMBULANCE S1 C1 A0
        8: RETURN-VEHICLE A0
        9: SEND-AMBULANCE S1 C2 A0
       10: SEND-POLICE S1 B1 P0
       11: RETURN-VEHICLE P0
       12: SEND-FIREFIGHTERS S1 B0 F0
       13: RETURN-VEHICLE F0
       14: SEND-FIREFIGHTERS S0 B0 F0
       15: SEND-POLICE S1 B0 P0
       16: RETURN-VEHICLE P0
       17: SEND-POLICE S0 B0 P0
       18: SEND-DEFUSE-TEAM S1 B0 DT0 BO1
```

```
       19: EVACUATE-BUILDING S1 B0 I0 BO1
       20: SEND-DEFUSE-TEAM S0 B0 DT0 BO1
       21: EVACUATE-BUILDING S0 B0 I0 BO1


statistics:    0.00 seconds instantiating 45 easy, 6 hard action templates
               0.00 seconds reachability analysis, yielding 645 facts and 51 actions
               0.00 seconds creating final representation with 42 relevant facts (24 max
               0.00 seconds building connectivity graph
               0.00 seconds (   0.00 pure) evaluating 154 states, to a max depth of 3
               0.00 seconds in DP for 7700 RPG ini state implication checks
               0.00 seconds in DP for 0 RPlan extract ini state implication checks (0 li
               0.00 seconds generating,    0.00 seconds encoding 1101 state transition b
               0.00 seconds in DP solving 328 state transition CNFs
               0.00 seconds checking for self-contradictions, including 0 DP calls
               0.00 seconds checking for stagnating states (241 hits), including 1515 DP
               0.02 seconds altogether checking for dominated states making 566 comparis
                    spending    0.00 seconds doing 4062 DP calls
              13605 total DP calls, 30164 total UP calls,    0.00 sec membership
               0.00 seconds for remaining searching duties
               0.02 seconds total time (+    0.02 secs for CNF memory allocation)
```

Version 2 output: Enforced hill-climbing plan not found; Best-first search used

```
    Cueing down from goal distance:   14 into depth [1]
                                      13           [1]
                                      12           [1][2]
                                      11           [1]
                                      10           [1]
                                       9           [1][2]
                                       8           [1][2]
                                       7           [1]
                                       6           [1][2]
                                       5           [1][2][3]
                                       4           [1][2][3]
                                       3           [1]
                                       2           [1]
                                       1

Enforced Hill-climbing failed !
switching to Best-first Search now.

advancing to distance :   14
                          13
                          12
                          11
                          10
                           9
                           8
```

```
                        7
                        6
                        5
                        4
                        3
                        2
                        0

ff: found legal plan as follows

step    0: SEND-HELICOPTER S0 C0 H0
        1: SEND-DEFUSE-TEAM S1 B2 DT0 BO0
        2: SEND-POLICE S1 B1 P0
        3: RETURN-VEHICLE P0
        4: SEND-AMBULANCE S1 C0 A0
        5: RETURN-VEHICLE A0
        6: SEND-AMBULANCE S1 C1 A0
        7: RETURN-VEHICLE A0
        8: SEND-AMBULANCE S1 C2 A0
        9: SEND-INTERVENTION S1 C0 I0
       10: RETURN-VEHICLE I0
       11: SEND-INTERVENTION S1 C2 I0
       12: EVACUATE-BUILDING S1 B2 I0 BO0
       13: SEND-DEFUSE-TEAM S1 B1 DT0 BO1
       14: SEND-FIREFIGHTERS S0 B0 F0
       15: RETURN-VEHICLE F0
       16: SEND-FIREFIGHTERS S1 B0 F0
       17: RETURN-VEHICLE I0
       18: SEND-POLICE S0 B0 P0
       19: RETURN-VEHICLE P0
       20: SEND-POLICE S1 B0 P0
       21: SEND-DEFUSE-TEAM S0 B0 DT0 BO1
       22: SEND-DEFUSE-TEAM S1 B0 DT0 BO1


statistics:    0.00 seconds instantiating 45 easy, 6 hard action templates
               0.00 seconds reachability analysis, yielding 647 facts and 51 actions
               0.00 seconds creating final representation with 46 relevant facts (26 max
               0.00 seconds building connectivity graph
               0.10 seconds (   0.04 pure) evaluating 630 states, to a max depth of 3
               0.06 seconds in DP for 31735 RPG ini state implication checks
               0.00 seconds in DP for 0 RPlan extract ini state implication checks (0 li
               0.14 seconds generating,    0.06 seconds encoding 6572 state transition b
               0.00 seconds in DP solving 656 state transition CNFs
               0.00 seconds checking for self-contradictions, including 0 DP calls
               0.00 seconds checking for stagnating states (2994 hits), including 20405
               0.16 seconds altogether checking for dominated states making 2944 compari
                    spending    0.04 seconds doing 21480 DP calls
              74276 total DP calls, 172313 total UP calls,    0.02 sec membership
               0.02 seconds for remaining searching duties
```

```
0.48 seconds total time (+    0.00 secs for CNF memory allocation)
```

By following the resulted plan, the goal state is achieved. The tool takes actions with regard to the unknown initial states (ex.: which street a civilian is on) and makes certain that despite the initial state, the final state is the same.

For example, the tool chooses to send the ambulance down both streets 0 and 1 to reach civilian c0. Thus, by the end of these two actions, the civilian is certainly no longer in need of medical assistance.

The same approach is taken in all unknown initial states.

The tool is also aware of the order of actions. Certain actions cannot take place before others. In this way, a chain of events is necessary (ex.: put out the fire, take out the terrorists, defuse the bomb, evacuate the building).

Chapter 7

Related work and documentation (W_{12})

7.1 Related approaches

A conference paper on Conformant-FF published at ICAPS'04 and in Artificial Intelligence served as the main documentation for the theoretical component of the project. The way the tool works and the logic behind the choice of representation of the world were thoroughly explained, as well as the algorithms implemented in the source code.

The examples provided with the tool were studied in order to make sense of the correct way to write the domain and the problem description which, together, serve as input.

The sites ready.gov, redcross.org and fema.gov consitute subject references. They helped in modelling the domain of the project, by rounding up possible issues and solutions that may be encountered during a disaster.

7.2 Advantages and limitations of your solution

The main advantage of the tool is action planning in lieu of exact initial states. The tool can find a workaround despite the lack of information and lead to the desired state.

Another advantage is the efficiency of the found plan. By using enforced hill-climbing and best-first search, the best solution is discovered. No extra steps other than the ones that are strictly needed are taken.

However, limitations are not non-existent. The tool lacks the capability to enforce different levels of urgency (ex.: if a civilian is in more urgent need of medical assistance than another, the ambulance should reach him first). The tool is also blind during the execution phase; it is not aware of dynamic changes, but only the initial states. For example, the system cannot be made to learn of any changes, such as the state of a street being changed or a new bomb being planted.

7.3 Possible extensions of the current work

Whilst the implemented project provides quite a number of predicates, it is not complete and can be extended. More actions can be devised and modelled after real world situations.

Appendix A

Original code

This section should contain only code developed by you, without any line re-used from other sources. This section helps me to correctly evaluate your amount of work and results obtained. Including in this section any line of code taken from someone else leads to failure of IS class this year. Failing or forgetting to add your code in this appendix leads to grade 1. Don't remove the above lines.

DOMAIN

```
(define (domain disaster)
(:requirements :strips)
(:predicates (STREET ?street)
(HELICOPTER ?helicopter)
(AMBULANCE ?ambulance)
(CIVILIAN ?civilian)
(FIREFIGHTERS ?firefighters)
(BUILDING ?building)
(BOMB ?bomb)
(POLICE ?police)
(DEFUSE-TEAM ?defuse-team)
(INTERVENTION ?intervention)
(under-siege ?building)
(not-evacuated ?building)
(armed ?bomb)
(at ?obj ?location)
(need-medical ?civilian)
(need-resources ?civilian)
(available ?vehicle)
(destroyed ?street)
(on-fire ?building)
(on ?building ?street)
)

(:action EVACUATE-BUILDING
  :parameters (?street ?building ?intervention ?bomb)
  :precondition (and (INTERVENTION ?intervention) (STREET ?street) (BUILDING ?building)
(BOMB ?bomb)
)
  :effect (and (not (available ?intervention))
```

```
(when (and (at ?bomb ?building) (on ?building ?street) (not(armed ?bomb)))
(not (not-evacuated ?building)) )
)
)

(:action SEND-DEFUSE-TEAM
  :parameters (?street ?building ?defuse-team ?bomb)
  :precondition (and (DEFUSE-TEAM ?defuse-team) (STREET ?street) (BUILDING ?building)
(BOMB ?bomb)
)
  :effect (and (not (available ?defuse-team))
(when (and (at ?bomb ?building) (on ?building ?street) (not(under-siege ?building)))
(not (armed ?bomb)) )
)
)

(:action SEND-POLICE
  :parameters (?street ?building ?police)
  :precondition (and (POLICE ?police) (STREET ?street) (BUILDING ?building)
  (available ?police)
)
  :effect (and (not (available ?police))
(when (and (on ?building ?street) (not (on-fire ?building)))
(not(under-siege ?building)) )
)
)

(:action SEND-AMBULANCE
  :parameters (?street ?civilian ?ambulance)
  :precondition (and (CIVILIAN ?civilian) (STREET ?street) (AMBULANCE ?ambulance)
(available ?ambulance) (not (destroyed ?street))
)
  :effect (and (not (available ?ambulance))
(when (at ?civilian ?street) (not (need-medical ?civilian)) )
)
)

(:action SEND-INTERVENTION
  :parameters (?street ?civilian ?intervention)
  :precondition (and (CIVILIAN ?civilian) (STREET ?street) (INTERVENTION ?intervention)
(available ?intervention) (not (destroyed ?street))
)
  :effect (and (not (available ?intervention))
(when (at ?civilian ?street) (not (need-resources ?civilian)) )
)
)

(:action SEND-HELICOPTER
  :parameters (?street ?civilian ?helicopter)
  :precondition (and (CIVILIAN ?civilian) (STREET ?street) (HELICOPTER ?helicopter)
```

```
 (destroyed ?street) (available ?helicopter)
)
  :effect (and (not (available ?helicopter))
(when (at ?civilian ?street)
(and (not (need-medical ?civilian)) (not (need-resources ?civilian))) )
)
)

(:action RETURN-VEHICLE
  :parameters (?vehicle)
  :precondition (and (or (AMBULANCE ?vehicle) (POLICE ?vehicle) (HELICOPTER ?vehicle)
(FIREFIGHTERS ?vehicle) (INTERVENTION ?vehicle) (DEFUSE-TEAM ?vehicle))
(not (available ?vehicle))
)
  :effect (when (not (available ?vehicle)) (available ?vehicle))
)

(:action SEND-FIREFIGHTERS
:parameters (?street ?building ?firefighters)
:precondition (and (FIREFIGHTERS ?firefighters) (BUILDING ?building) (STREET ?street)
(available ?firefighters)
)
:effect (and (not (available ?firefighters))
(when (on ?building ?street) (not (on-fire ?building)))
)
)

)
```

ENFORCED HILL-CLIMBING PROBLEM DESCRIPTION

```
(define (problem disaster-a1-c2-h1-s2)
(:domain disaster)
(:objects
s0
s1
h0
a0
c0
c1
c2
f0
b0
b1
b2
bo0
bo1
p0
dt0
i0
)
```

```
(:init
(STREET s0)
(STREET s1)
(HELICOPTER h0)
(AMBULANCE a0)
(CIVILIAN c0)
(CIVILIAN c1)
(CIVILIAN c2)
(FIREFIGHTERS f0)
(BUILDING b0)
(BUILDING b1)
(BUILDING b2)
(BOMB bo0)
(BOMB bo1)
(POLICE p0)
(DEFUSE-TEAM dt0)
(INTERVENTION i0)
(at c1 s1)
(at c2 s1)
(need-medical c0)
(need-medical c1)
(need-resources c0)
(destroyed s0)
(available a0)
(on-fire b0)
(under-siege b0)
(under-siege b1)
(on b1 s1)
(on b2 s1)
(available p0)
(available f0)
(available h0)
(armed bo0)
(armed bo1)
(at bo0 b2)
(at bo1 b0)
(available i0)
(available dt0)
(not-evacuated b0)

(unknown (at c0 s0))
(unknown (at c0 s1))
(or
(not (at c0 s0))
(not (at c0 s1))
)
(oneof
(at c0 s0)
(at c0 s1)
```

```
)
(unknown (on b0 s0))
(unknown (on b0 s1))
(or
(not (on b0 s0))
(not (on b0 s1))
)
(oneof
(on b0 s0)
(on b0 s1)
)

(unknown (need-resources c2))
(unknown (need-medical c2))
(or
(not (need-resources c2))
(not (need-medical c2))
)
(oneof
(need-resources c2)
(need-medical c2)
)
)

(:goal
(and (not (need-medical c0))
(not (need-medical c1))
(not (need-resources c2))
(not (need-medical c2))
(not (need-resources c0))
(not (not-evacuated b0))
(not (on-fire b0))
(not (on-fire b1))
(not (under-siege b1))
(not (under-siege b0))
(not (armed bo0))
(not (armed bo1))
)
)

)
```

BEST-FIRST SEARCH PROBLEM DESCRIPTION

```
(define (problem disaster-a1-c2-h1-s2)
(:domain disaster)
(:objects
s0
s1
h0
a0
```

```
c0
c1
c2
f0
b0
b1
b2
bo0
bo1
p0
dt0
i0
)

(:init
(STREET s0)
(STREET s1)
(HELICOPTER h0)
(AMBULANCE a0)
(CIVILIAN c0)
(CIVILIAN c1)
(CIVILIAN c2)
(FIREFIGHTERS f0)
(BUILDING b0)
(BUILDING b1)
(BUILDING b2)
(BOMB bo0)
(BOMB bo1)
(POLICE p0)
(DEFUSE-TEAM dt0)
(INTERVENTION i0)
(at c1 s1)
(at c2 s1)
(need-medical c0)
(need-medical c1)
(need-resources c0)
(destroyed s0)
(available a0)
(on-fire b0)
(under-siege b0)
(under-siege b1)
(on b1 s1)
(on b2 s1)
(available p0)
(available f0)
(available h0)
(armed bo0)
(armed bo1)
(at bo0 b2)
(available i0)
```

```
(available dt0)
(not-evacuated b2)

(unknown (at c0 s0))
(unknown (at c0 s1))
(or
(not (at c0 s0))
(not (at c0 s1))
)
(oneof
(at c0 s0)
(at c0 s1)
)
(unknown (on b0 s0))
(unknown (on b0 s1))
(or
(not (on b0 s0))
(not (on b0 s1))
)
(oneof
(on b0 s0)
(on b0 s1)
)

(unknown (need-resources c2))
(unknown (need-medical c2))
(or
(not (need-resources c2))
(not (need-medical c2))
)
(oneof
(need-resources c2)
(need-medical c2)
)

(unknown (at bo1 b0))
(unknown (at bo1 b1))
(or
(not (at bo1 b0))
(not (at bo1 b1))
)
(oneof
(at bo1 b0)
(at bo1 b1)
)

)

(:goal
(and (not (need-medical c0))
```

```
(not (need-medical c1))
(not (need-resources c2))
(not (need-medical c2))
(not (need-resources c0))
(not (not-evacuated b2))
(not (on-fire b0))
(not (on-fire b1))
(not (under-siege b1))
(not (under-siege b0))
(not (armed bo0))
(not (armed bo1))
)
)

)
```

Appendix B

Quick technical guide for running the project

Requirements
The project was tested on a Ubuntu virtual machine.

Download the linux executable from `https://fai.cs.uni-saarland.de/hoffmann/cff.html`. Optionally, the benchmark problems can also be downloaded.

Step by step technical manual
Create the a folder named `cff-tests` or open the folder if the benchmark problems were downloaded.

Create the domain file and paste the code from Appendix A DOMAIN. Create the problem description files with the names enforced-hill-climbing and best-first-search (.pddl) and paste the code from Appendix A ENFORCED HILL-CLIMBING PROBLEM DESCRIPTION and BEST-FIRST SEARCH PROBLEM DESCRIPTION.

Open a terminal in the folder where the executable was downloaded. Write the following in the terminal:
./Conformant-FF -o ./cff-tests/disaster/domain -f ./cff-tests/disaster/enforced-hill-climbing.pddl

, respectively

./Conformant-FF -o ./cff-tests/disaster/domain -f ./cff-tests/disaster/best-first-search.pddl

Bibliography

https://fai.cs.uni-saarland.de/hoffmann/cff.html

https://www.ready.gov/

https://www.redcross.org/get-help/how-to-prepare-for-emergencies/make-a-plan.html

icaps04, Ronen Brafman and Jörg Hoffmann, Conformant Planning via Heuristic Forward Search: A New Approach, 2004

https://www.fema.gov/news-release/2018/03/29/how-make-disaster-plan