

CW1 Report

F29AI - Artificial Intelligence

Keqin ZHANG (H00460395) Yuwei ZHAO (H00460398)

OUC CW1 33 2025-11-10

Abstract

This report involves the solutions to the tasks outlined in Coursework 1 for the F29AI – Artificial Intelligence course. The main objective of the coursework is to search algorithms and automated planning using *PDDL*.

See Resources on [git@github.com:ZhangKeqin0307/coursework1.git](https://github.com:ZhangKeqin0307/coursework1.git).

1 Introduction

2 Procedure

2.1 Part 1 - Solving and Analyzing Sudoku with Search Algorithms

2.1.1 Part 1A

< Brief Description >

Assume that a *Sudoku* consists of a 9×9 grid. The objective is to fill the grid with digits in such a way that each row, each column, and each of the 9 principal 3×3 sub-squares contains all of the digits from 1 to 9. An approach to an intelligent sudoku solver is needed.

- Define Sudoku formally as a constraint satisfaction problem.
- What are the variables, domains, and constraints?
- Discuss the time complexity of brute-force search vs. backtracking in Sudoku.

< Solution >

A CSP(constraint satisfaction problem) should involve the following three components: Variables, Domains and Constraints. Therefore, we can define the Sudoku problem as follows:

$$\text{Sudoku} = \langle V, D, C \rangle$$

where

1. V : The set of 81 variables, $V = \{V_{i,j} \mid i, j \in \{1, 2, \dots, 9\}\}$.
2. D : The domain $D_{i,j}$ for each variable $V_{i,j}$ is defined as:
 - $D_{i,j} = \{k\}$, if $V_{i,j}$ is a given cell with value k .
 - $D_{i,j} = \{1, 2, \dots, 9\}$, if $V_{i,j}$ is an empty cell.
3. C : The set of 27 "All-Different" constraints:
 - C_{row} : For each row i , all variables $V_{i,1}, V_{i,2}, \dots, V_{i,9}$ must have different values.
 - C_{col} : For each column j , all variables $V_{1,j}, V_{2,j}, \dots, V_{9,j}$ must have different values.
 - C_{subgrid} : For each 3×3 subgrid, all 9 variables within that subgrid must have different values.

Time Complexity Analysis:

- **Brute-force Search Algorithm:**
 For each of the k spaces, there are 9 possible choices of numbers. This results in a total of $9 \times 9 \times \dots \times 9$ (k times) combinations. Therefore, the time complexity of the brute-force search algorithm is $O(9^k)$. When the worst-case scenario occurs, the algorithm needs to explore all possible combinations, leading to the $O(9^{81})$ time complexity.
- **Backtracking Search Algorithm:**
 It checks the validity of constraints (row, column, and 3×3 sub-grid) **immediately** after assigning a number to a cell. If a conflict is detected (i.e., the current partial solution is invalid), the algorithm recursively "backtracks" to the previous step to try a different number. This process effectively **prunes** large sub-trees of the search space that are known to be invalid.

 While the theoretical worst-case time complexity remains $O(9^k)$ (similar to brute-force), the average-case performance is drastically faster. This is because the *effective branching factor* b becomes significantly smaller than 9 ($b \ll 9$) as the constraints restrict the number of valid choices for each subsequent cell.

2.1.2 Part 1B

< Brief Description >

Build an intelligent *Sudoku* solver that accepts input puzzles from *.txt* or *.csv* files. The program displays the solved puzzle in a clean format while printing relevant performance metrics, and concludes with a comparison against the A^* search algorithm.

< Solution >

Language	Python
Solver	Backtracking with pruning
Input	9×9 CSV file (0 represents empty)
Output	GUI display and relevant metrics printed
Testing	Several <i>Sudoku</i> puzzle tests

Table 1: Preparation of Sudoku Solver Implementation

Backend Part:

1. Problem abstraction & CSP modelling

As stated in Part 2.1.1, model the puzzle as a CSP:

$$Sudoku = \langle V, D, C \rangle$$

where V is the 81 variables $V_{i,j}$; $D_{i,j}$ are domains and C are the row / column / 3×3 subgrid all-different constraints.

In code this is corresponding to:

- `sudoku_solver.board` : a 9×9 integer matrix (0 means empty).
- `is_valid(row,col,num)` : enforces the three constraint types for a tentative assignment.

2. Input format preset

Implement robust input routines that accept *.csv* formatted 9×9 grids. It needs to read rows with comma separation, convert each token to int, accept 0 or blank as empty.

In our code base this is the method `load_from_csv(filepath)`. Validate that the parsed grid has 9 rows \times 9 columns, otherwise throw / report an error.

3. Core solver build

...

Frontend Part:

1. ...

2.2 Part 2 - Automated Planning

2.2.1 Part 2A: Modelling the Domain

2.2.2 Part 2B: Modelling the Problems

2.2.3 Part 2C: Extension

3 Reflection and Analysis

4 Conclusion

5 Source Code

- Part1 - *sudoku_solver.py*

```
1 import csv
2 import time
3
4 class SudokuSolver:
5     def __init__(self):
6         self.board = []
7         # Relevant metrics
8         self.steps = 0          # Total number of steps
9         self.recursive_calls = 0 # Recursive calls
10        self.backtracks = 0      # Backtracks
11        self.start_time = 0
12        self.execution_time = 0
13
14    def load_from_csv(self, filepath):
15        self.board = []
16        try:
17            with open(filepath, 'r', encoding='UTF-8') as f:
18                reader = csv.reader(f)
19                for row in reader:
20                    # Converts the string to an integer, handling possible
21                    ↪ whitespace
22                    index = [int(num.strip()) for num in row if
23                    ↪ num.strip().isdigit()]
24                    if len(index) == 9:
25                        self.board.append(index)
26
27                if len(self.board) != 9:
28                    raise ValueError("Invalid row count in CSV")
29
30                print(f"[Succeed] File loaded: {filepath}")
31                return True
32        except Exception as e:
33            print(f"[Error] File loading failed: {e}")
34            return False
35
36    def is_valid(self, row, col, num):
37        # Row check
38        for x in range(9):
39            if self.board[row][x] == num:
40                return False
41
42        # Column check
43        for x in range(9):
44            if self.board[x][col] == num:
45                return False
46
47        # 3x3 box check
48        start_row = row - row % 3
49        start_col = col - col % 3
50        for i in range(3):
51            for j in range(3):
52                if self.board[i + start_row][j + start_col] == num:
```

```

53         return True
54
55     def solve_algorithm(self):
56         self.recursive_calls += 1
57
58         for i in range(9):
59             for j in range(9):
60                 if self.board[i][j] == 0:
61                     for num in range(1, 10):
62                         self.steps += 1
63                         # If invalid, prune it
64                         if self.is_valid(i, j, num):
65                             self.board[i][j] = num
66
67                             if self.solve_algorithm():
68                                 return True
69
70                             # Backtracking
71                             self.board[i][j] = 0
72                             self.backtracks += 1
73
74         return False
75     return True
76
77     def run_solver(self):
78         self.steps = 0
79         self.recursive_calls = 0
80         self.backtracks = 0
81
82         self.start_time = time.perf_counter()
83
84         success = self.solve_algorithm()
85
86         end_time = time.perf_counter()
87         self.execution_time = (end_time - self.start_time) * 1000
88
89         return success
90
91     def print_metrics(self):
92         print("\n")
93         print(f"Execution Time: {self.execution_time:.4f} ms")
94         print(f"Total Steps (Attempts): {self.steps}")
95         print(f"Recursive Calls: {self.recursive_calls}")
96         print(f"Backtracks: {self.backtracks}")
97         print("\n")

```

• Part1 - *sudoku_gui.py*

```

1  import os
2  import sys
3
4  from PyQt6.QtWidgets import (
5      QApplication, QMainWindow, QWidget,      QGridLayout,
6      QVBoxLayout, QHBoxLayout, QPushButton, QLineEdit,
7      QFileDialog, QLabel,      QFrame,      QMessageBox
8  )
9  from PyQt6.QtCore import Qt
10 from PyQt6.QtGui import QFont, QIntValidator
11

```

```

12 from utils import load_qss, base_path
13 from sudoku_solver import SudokuSolver
14
15 class SudokuCell(QLineEdit):
16     def __init__(self):
17         super().__init__()
18         self.setFixedSize(50, 50)
19         self.setAlignment(Qt.AlignmentFlag.AlignCenter)
20         self.setFont(QFont("Arial", 20))
21         self.setMaxLength(1)
22         self.setValidator(QIntValidator(1, 9))
23
24         default_style_path = os.path.join(base_path(), "css_styles",
25             ↪ "default_style.qss")
26         solved_style_path = os.path.join(base_path(), "css_styles",
27             ↪ "solved_style.qss")
28
29         self.default_style = load_qss(default_style_path)
30         self.solved_style = load_qss(solved_style_path)
31
32         self.setStyleSheet(self.default_style)
33
34     def set_solved_style(self):
35         self.setStyleSheet(self.solved_style)
36
37     def reset_style(self):
38         self.clear()
39         self.setStyleSheet(self.default_style)
40
41 class SudokuMainWindow(QMainWindow):
42     def __init__(self):
43         super().__init__()
44         self.setWindowTitle(None)
45         self.setFixedSize(600, 750)
46
47         self.setStyleSheet("background-color: #333333;")
48
49         self.solver = SudokuSolver()
50         self.cells = [[None for _ in range(9)] for _ in range(9)]
51
52         central_widget = QWidget()
53         self.setCentralWidget(central_widget)
54         self.main_layout = QVBoxLayout(central_widget) # Veritcal Layout
55         self.main_layout.setContentsMargins(20, 20, 20, 20)
56         self.main_layout.setSpacing(15)
57
58         title = QLabel("Sudoku")
59         title.setFont(QFont("Arial", 25, QFont.Weight.Bold))
60         title.setAlignment(Qt.AlignmentFlag.AlignCenter)
61         title.setStyleSheet("color: #DDDDDD")
62         self.main_layout.addWidget(title)
63
64         self.grid_panel()
65
66         self.info_panel = QLabel("Sudoku Game")
67         self.info_panel.setObjectName("info_panel")
68         self.info_panel.setFont(QFont("Arial", 16))

```

```

69     panel_style_path = os.path.join(base_path(), "css_styles",
    ↪     "panel_style.qss")
70
71     self.info_panel.setStyleSheet(load_qss(panel_style_path))
72     self.info_panel.setAlignment(Qt.AlignmentFlag.AlignCenter)
73     self.main_layout.addWidget(self.info_panel)
74
75     self.button_panel()
76
77     self.center_window()
78
79     def grid_panel(self):
80         grid_container = QWidget()
81
82         outer_layout = QGridLayout(grid_container)
83         outer_layout.setSpacing(10)
84
85         for br in range(3):
86             for bc in range(3):
87                 block_frame = QFrame()
88                 block_frame.setStyleSheet("background-color: #DDDDDD;
    ↪                 border-radius: 5px;")
89                 inner_layout = QGridLayout(block_frame)
90                 inner_layout.setSpacing(2)
91                 inner_layout.setContentsMargins(2, 2, 2, 2)
92
93                 for i in range(3):
94                     for j in range(3):
95                         global_r = br * 3 + i
96                         global_c = bc * 3 + j
97
98                         cell = SudokuCell()
99                         self.cells[global_r][global_c] = cell
100                         inner_layout.addWidget(cell, i, j)
101
102                 outer_layout.addWidget(block_frame, br, bc)
103
104         self.main_layout.addWidget(
105             grid_container,
106             alignment = Qt.AlignmentFlag.AlignCenter
107         )
108
109     def button_panel(self):
110         btn_layout = QHBoxLayout()
111
112         button_style_path = os.path.join(base_path(), "css_styles",
    ↪         "button_style.qss")
113
114         btn_load = QPushButton("Load CSV")
115         btn_load.setStyleSheet(load_qss(button_style_path))
116         btn_load.clicked.connect(self.load_csv)
117
118         btn_solve = QPushButton("Solve Now")
119         btn_solve.setStyleSheet(load_qss(button_style_path))
120         btn_solve.clicked.connect(self.solve_puzzle)
121
122         btn_clear = QPushButton("Clear")
123         btn_clear.setStyleSheet(load_qss(button_style_path))
124         btn_clear.clicked.connect(self.clear_board)

```

```

125
126     btn_layout.addWidget(btn_load)
127     btn_layout.addWidget(btn_solve)
128     btn_layout.addWidget(btn_clear)
129
130     self.main_layout.addLayout(btn_layout)
131
132     def load_csv(self):
133         file_path, _ = QFileDialog.getOpenFileName(self, "Open CSV",
134             ↪ base_path(), "CSV Files (*.csv)")
135         if file_path:
136             success = self.solver.load_from_csv(file_path)
137             if success:
138                 self.update_ui_from_solver()
139                 self.info_panel.setText(f"File Loaded:
140                 ↪ {file_path.split('/')[-1]}")
141             else:
142                 QMessageBox.critical(self, "Error", "File loaded failed")
143
144     def update_ui_from_solver(self):
145         for r in range(9):
146             for c in range(9):
147                 val = self.solver.board[r][c]
148                 cell = self.cells[r][c]
149                 cell.reset_style()
150                 if val != 0:
151                     cell.setText(str(val))
152                 else:
153                     cell.setText("")
154
155     def sync_gui_to_backend(self):
156         current_board = []
157         for r in range(9):
158             row_data = []
159             for c in range(9):
160                 text = self.cells[r][c].text()
161                 if text.isdigit():
162                     row_data.append(int(text))
163                 else:
164                     row_data.append(0)
165             current_board.append(row_data)
166
167         self.solver.board = current_board
168
169     def solve_puzzle(self):
170         try:
171             self.sync_gui_to_backend()
172
173             if not self.self_check_isvalid():
174                 raise ValueError("[WARNING] Contradiction appears")
175
176             if not self.solver.run_solver():
177                 raise ValueError("[ERROR] The question is fucking difficult")
178
179             for r in range(9):
180                 for c in range(9):
181                     val = self.solver.board[r][c]
182                     cell = self.cells[r][c]
183                     cell.setText(str(val))

```

```

182         cell.set_solved_style()
183
184         metrics = (
185             f" Time: {self.solver.execution_time:.2f} ms | "
186             f" Steps: {self.solver.steps} | "
187             f" Backtracks: {self.solver.backtracks}"
188         )
189         self.info_panel.setText(metrics)
190         self.solver.print_metrics()
191
192     except Exception as e:
193         print(e)
194         self.info_panel.setText("Try again!")
195         QMessageBox.critical(self, "Error", str(e))
196
197     def self_check_isvalid(self):
198         board = self.solver.board
199
200         for r in range(9):
201             for c in range(9):
202                 num = board[r][c]
203                 if num != 0:
204
205                     board[r][c] = 0
206                     if not self.solver.is_valid(r, c, num):
207                         board[r][c] = num
208                         return False
209                     board[r][c] = num
210         return True
211
212     def center_window(self):
213         screen = QApplication.primaryScreen()
214         screen_geometry = screen.availableGeometry()
215         window_geometry = self.frameGeometry()
216         window_geometry.moveCenter(screen_geometry.center())
217         self.move(window_geometry.topLeft())
218
219     def clear_board(self):
220         self.solver.board = [[0]*9 for _ in range(9)]
221         for r in range(9):
222             for c in range(9):
223                 self.cells[r][c].clear()
224                 self.cells[r][c].reset_style()
225         self.info_panel.setText("Clear!")
226
227 if __name__ == "__main__":
228     app = QApplication(sys.argv)
229
230     app.setStyle("Fusion")
231
232     window = SudokuMainWindow()
233     window.show()
234     sys.exit(app.exec())

```

• Part2A&B - domain1.pddl

```

1 (define (domain lunar)
2   (:requirements :strips :typing)
3

```

```

4      ; -----
5      ; Types
6      ; -----
7
8      (:types
9          lander
10         rover
11         location
12         sample
13         data
14         image scan - data
15     )
16
17     ; -----
18     ; Predicates
19     ; -----
20
21     (:predicates
22         ; Positions
23         (lander-at ?l - lander ?loc - location)
24         (at ?r - rover ?loc - location)
25
26         ; Connectivity between surface locations
27         (path ?from - location ?to - location)
28
29         ; The association between rovers and their respective landers
30         (assigned ?r - rover ?l - lander)
31
32         ; Landing location
33         (unplaced ?l - lander)
34
35         ; Deployment status
36         (deployed ?r - rover)
37
38         ; If lander had not carried sample
39         (lander-free ?l - lander)
40
41         ; Data tasks
42         ; Data objectives
43         (image-target ?img - image ?loc - location)
44         (scan-target ?sc - scan ?loc - location)
45         ; If data has been captured
46         (taken ?d - data)
47         ; If rover hold data
48         (empty-memory ?r - rover)
49         (holding-data ?r - rover ?d - data)
50         ; If data has been transmitted to lander
51         (transmitted ?d - data ?l - lander)
52
53         ; Sample tasks
54         ; Sample objectives
55         (sample-at ?s - sample ?loc - location)
56         ; If rover hold sample
57         (holding-sample ?r - rover ?s - sample)
58         ; Collection of sample
59         (sample-picked-up ?s - sample)
60         (sample-stored ?s - sample)
61         (stored ?s - sample ?l - lander)
62     )

```

```

63
64 ; -----
65 ; Actions
66 ; -----
67
68 ; Landing location
69 (:action choose-landing
70   :parameters (?l - lander ?wp - location)
71   :precondition
72     (and (unplaced ?l))
73   :effect
74     (and
75       (not (unplaced ?l))
76       (lander-at ?l ?wp)
77     )
78 )
79
80 ; Deploy rover
81 (:action deploy
82   :parameters (?r - rover ?l - lander ?loc - location)
83   :precondition (and
84     (assigned ?r ?l)
85     (lander-at ?l ?loc)
86     (not (deployed ?r))
87   )
88   :effect (and
89     (deployed ?r)
90     (at ?r ?loc)
91   )
92 )
93
94 ;Retrieve rover
95 (:action retrieve
96   :parameters (?r - rover ?l - lander ?loc - location)
97   :precondition (and
98     (assigned ?r ?l)
99     (deployed ?r)
100    (at ?r ?loc)
101    (lander-at ?l ?loc)
102  )
103   :effect (and
104     (not (deployed ?r))
105     (not (at ?r ?loc))
106  )
107 )
108
109 ; Movement of rover
110 (:action move
111   :parameters (?r - rover ?from - location ?to - location)
112   :precondition (and
113     (deployed ?r)
114     (at ?r ?from)
115     (path ?from ?to)
116   )
117   :effect (and
118     (at ?r ?to)
119     (not (at ?r ?from))
120   )
121 )

```

```

122
123 ; Take Image
124 (:action take-image
125   :parameters (?r - rover ?img - image ?loc - location)
126   :precondition (and
127     (deployed ?r)
128     (at ?r ?loc)
129     (image-target ?img ?loc)
130     (empty-memory ?r)
131     (not (taken ?img))
132   )
133   :effect (and
134     (holding-data ?r ?img)
135     (taken ?img)
136     (not (empty-memory ?r))
137   )
138 )
139
140 ; Perform Scan
141 (:action scan
142   :parameters (?r - rover ?sc - scan ?loc - location)
143   :precondition (and
144     (deployed ?r)
145     (at ?r ?loc)
146     (scan-target ?sc ?loc)
147     (empty-memory ?r)
148     (not (taken ?sc))
149   )
150   :effect (and
151     (holding-data ?r ?sc)
152     (taken ?sc)
153     (not (empty-memory ?r))
154   )
155 )
156
157 ; Transmit Data to Lander
158 (:action transmit
159   :parameters (?r - rover ?d - data ?l - lander)
160   :precondition (and
161     (assigned ?r ?l)
162     (deployed ?r)
163     (holding-data ?r ?d)
164   )
165   :effect (and
166     (transmitted ?d ?l)
167     (empty-memory ?r)
168     (not (holding-data ?r ?d))
169   )
170 )
171
172 ; Pick up Sample
173 (:action pick-up-sample
174   :parameters (?r - rover ?s - sample ?loc - location)
175   :precondition (and
176     (deployed ?r)
177     (at ?r ?loc)
178     (sample-at ?s ?loc)
179     (not (sample-picked-up ?s))
180 )

```

```

181         :effect (and
182             (holding-sample ?r ?s)
183             (sample-picked-up ?s)
184             (not (sample-at ?s ?loc))
185         )
186     )
187
188     ; Store Sample in Lander
189     (:action store-sample
190         :parameters (?r - rover ?s - sample ?l - lander)
191         :precondition (and
192             (not (deployed ?r))
193             (holding-sample ?r ?s)
194             (assigned ?r ?l)
195             (lander-free ?l)
196         )
197         :effect (and
198             (stored ?s ?l)
199             (sample-stored ?s)
200             (not (holding-sample ?r ?s))
201             (not (lander-free ?l))
202         )
203     )
204 )

```

• Part2A&B - mission1.pddl

```

1  (define (problem lunar-mission-1)
2      (:domain lunar)
3
4      (:objects
5          lander1 - lander
6          rover1 - rover
7          ; Map
8          wp1 wp2 wp3 wp4 wp5 wp6 - location
9          ; Data
10         image5 - image
11         scan3 - scan
12         ; Sample
13         sample1 - sample
14     )
15
16     (:init
17         ; The association between rover1 and lander1
18         (assigned rover1 lander1)
19
20         ; Lander1 can land at any location
21         (unplaced lander1)
22
23         ; Rover1 has not been deployed
24         (not (deployed rover1))
25
26         ; Lander1 has not stored sample
27         (lander-free lander1)
28
29         ; Rover1 memory is empty
30         (empty-memory rover1)
31
32         ; Connectivity between surface locations(Figure 2)

```

```

33     (path wp1 wp4)
34     (path wp4 wp3)
35     (path wp3 wp5)
36     (path wp5 wp1)
37     (path wp1 wp2)
38     (path wp2 wp3)
39
40     ; task objective
41     (image-target image5 wp5)
42     (scan-target scan3 wp3)
43     (sample-at sample1 wp1)
44 )
45
46 (:goal
47   (and
48     ; Indicates that image5 has been captured
49     (taken image5)
50
51     ; Indicates that scan3 has been captured
52     (taken scan3)
53
54     ; Indicates that sample1 has been collected and brought back
55     (sample-stored sample1)
56   )
57 )
58 )

```

• Part2A&B - mission2.pddl

```

1  (define (problem lunar-mission-2)
2    (:domain lunar)
3
4    (:objects
5      lander1 lander2 - lander
6      rover1 rover2 - rover
7      ; Map
8      wp1 wp2 wp3 wp4 wp5 wp6 - location
9      ; Data
10     image2 image3 - image
11     scan4 scan6 - scan
12     ; Sample
13     sample1 sample5 - sample
14   )
15
16   (:init
17     ; The association between rovers and their respective landers
18     (assigned rover1 lander1)
19     (assigned rover2 lander2)
20
21     ; Rover initial deployment states
22     ; Rover1 starts deployed at wp2 (with lander2 at wp2)
23     (not (unplaced lander1))
24     (lander-at lander1 wp2)
25     (at rover1 wp2)
26     (deployed rover1)
27
28     ; Initial status
29     (lander-free lander1)
30     (empty-memory rover1)

```

```

31
32      ; Rover2 starts undeployed (lander2 unplaced ( to be chosen))
33      (unplaced lander2)
34      (not (deployed rover2))
35
36      ; Initial status of 1
37      (lander-free lander2)
38      (empty-memory rover2)
39
40      ; Connectivity between surface locations(Figure 2)
41      (path wp1 wp2)
42      (path wp2 wp1)
43      (path wp2 wp3)
44      (path wp3 wp5)
45      (path wp5 wp3)
46      (path wp5 wp6)
47      (path wp6 wp4)
48      (path wp2 wp4)
49      (path wp4 wp2)
50
51      ; task objective
52      ; Image
53      (image-target image2 wp2)
54      (image-target image3 wp3)
55      ; Scan
56      (scan-target scan4 wp4)
57      (scan-target scan6 wp6)
58      ; Sample
59      (sample-at sample1 wp1)
60      (sample-at sample5 wp5)
61  )
62
63  (:goal
64    (and
65      ; Indicates that image2 and image3 have been captured
66      (taken image2)
67      (taken image3)
68
69      ; Indicates that scan4 and scan6 have been captured
70      (taken scan4)
71      (taken scan6)
72
73      ; Indicates that sample1 and sample5 have been collected(picked up
74      ↪ and stored)
75      (sample-stored sample1)
76      (sample-stored sample5)
77    )
78  )

```

• Part2C - domain2.pddl

```

1  (define (domain lunar)
2    (:requirements :strips :typing)
3
4    ; -----
5    ; Types
6    ; -----
7

```

```

8      (:types
9          lander
10         rover
11         location
12         sample
13         data
14         image scan - data
15         ; New:
16         astronaut
17         area
18     )
19
20     ; -----
21     ; Predicates
22     ; -----
23
24     (:predicates
25         ; positions
26         (lander-at ?l - lander ?loc - location)
27         (at ?r - rover ?loc - location)
28
29         ; Connectivity between surface locations
30         (path ?from - location ?to - location)
31
32         ; The association between rovers and their respective landers
33         (assigned ?r - rover ?l - lander)
34
35         ; Landing location
36         (unplaced ?l - lander)
37
38         ; deployment status
39         (deployed ?r - rover)
40
41         ; If lander had not carried sample
42         (lander-free ?l - lander)
43
44         ; Data tasks
45         ; Data objectives
46         (image-target ?img - image ?loc - location)
47         (scan-target ?sc - scan ?loc - location)
48         ; If data has been captured
49         (taken ?d - data)
50         ; If rover hold data
51         (empty-memory ?r - rover)
52         (holding-data ?r - rover ?d - data)
53         ; If data has been transmitted to lander
54         (transmitted ?d - data ?l - lander)
55
56         ; Sample tasks
57         ; Sample objectives
58         (sample-at ?s - sample ?loc - location)
59         ; If rover hold sample
60         (holding-sample ?r - rover ?s - sample)
61         ; Collection of sample
62         (sample-picked-up ?s - sample)
63         (sample-stored ?s - sample)
64         (stored ?s - sample ?l - lander)
65
66         ; New:

```

```

67         ; The astronaut is located in which area of the lander
68         (crew-in ?a - astronaut ?l - lander ?ar - area)
69     )
70
71     ; -----
72     ; Actions
73     ; -----
74
75     ; New action:
76     ; Movement of astronaut in lander
77     (:action move-crew
78         :parameters (?a - astronaut ?l - lander ?from - area ?to - area)
79         :precondition (crew-in ?a ?l ?from)
80         :effect (and
81             (crew-in ?a ?l ?to)
82             (not (crew-in ?a ?l ?from))
83         )
84     )
85
86     ; Landing location
87     (:action choose-landing
88         :parameters (?l - lander ?wp - location)
89         :precondition
90             (and (unplaced ?l))
91         :effect
92             (and
93                 (not (unplaced ?l))
94                 (lander-at ?l ?wp)
95             )
96     )
97
98     ; Deploy rover
99     (:action deploy
100         :parameters (?r - rover ?l - lander ?loc - location ?a - astronaut)
101         :precondition (and
102             (assigned ?r ?l)
103             (lander-at ?l ?loc)
104             (not (deployed ?r))
105             (crew-in ?a ?l docking-bay)
106         )
107         :effect (and
108             (deployed ?r)
109             (at ?r ?loc)
110         )
111     )
112
113     ;Retrieve rover
114     (:action retrieve
115         :parameters (?r - rover ?l - lander ?loc - location ?a - astronaut)
116         :precondition (and
117             (assigned ?r ?l)
118             (deployed ?r)
119             (at ?r ?loc)
120             (lander-at ?l ?loc)
121             (crew-in ?a ?l docking-bay)
122         )
123         :effect (and
124             (not (deployed ?r))
125             (not (at ?r ?loc))

```

```

126     )
127 )
128
129 ; Movement of rover
130 (:action move
131     :parameters (?r - rover ?from - location ?to - location)
132     :precondition (and
133         (deployed ?r)
134         (at ?r ?from)
135         (path ?from ?to)
136     )
137     :effect (and
138         (at ?r ?to)
139         (not (at ?r ?from))
140     )
141 )
142
143 ; Take Image
144 (:action take-image
145     :parameters (?r - rover ?img - image ?loc - location)
146     :precondition (and
147         (deployed ?r)
148         (at ?r ?loc)
149         (image-target ?img ?loc)
150         (empty-memory ?r)
151         (not (taken ?img))
152     )
153     :effect (and
154         (holding-data ?r ?img)
155         (taken ?img)
156         (not (empty-memory ?r))
157     )
158 )
159
160 ; Perform Scan
161 (:action scan
162     :parameters (?r - rover ?sc - scan ?loc - location)
163     :precondition (and
164         (deployed ?r)
165         (at ?r ?loc)
166         (scan-target ?sc ?loc)
167         (empty-memory ?r)
168         (not (taken ?sc))
169     )
170     :effect (and
171         (holding-data ?r ?sc)
172         (taken ?sc)
173         (not (empty-memory ?r))
174     )
175 )
176
177 ; Transmit Data to Lander
178 (:action transmit
179     :parameters (?r - rover ?d - data ?l - lander ?a - astronaut)
180     :precondition (and
181         (assigned ?r ?l)
182         (deployed ?r)
183         (holding-data ?r ?d)
184         (crew-in ?a ?l control-room)

```

```

185     )
186     :effect (and
187         (transmitted ?d ?l)
188         (empty-memory ?r)
189         (not (holding-data ?r ?d))
190     )
191 )
192
193 ; Pick up Sample
194 (:action pick-up-sample
195     :parameters (?r - rover ?s - sample ?loc - location)
196     :precondition (and
197         (deployed ?r)
198         (at ?r ?loc)
199         (sample-at ?s ?loc)
200         (not (sample-picked-up ?s))
201     )
202     :effect (and
203         (holding-sample ?r ?s)
204         (sample-picked-up ?s)
205         (not (sample-at ?s ?loc))
206     )
207 )
208
209 ; Store Sample in Lander
210 (:action store-sample
211     :parameters (?r - rover ?s - sample ?l - lander ?a - astronaut)
212     :precondition (and
213         (not (deployed ?r))
214         (holding-sample ?r ?s)
215         (assigned ?r ?l)
216         (lander-free ?l)
217         (crew-in ?a ?l docking-bay)
218     )
219     :effect (and
220         (stored ?s ?l)
221         (sample-stored ?s)
222         (not (holding-sample ?r ?s))
223         (not (lander-free ?l))
224     )
225 )
226 )

```

• Part2C - *mission3.pddl*

```

1 (define (problem lunar-mission-2)
2   (:domain lunar)
3
4   (:objects
5     lander1 lander2 - lander
6     rover1 rover2 - rover
7     ; Map
8     wp1 wp2 wp3 wp4 wp5 wp6 - location
9     ; Data
10    image2 image3 - image
11    scan4 scan6 - scan
12    ; Sample
13    sample1 sample5 - sample
14    ; New:

```

```

15      ; Area
16      control-room docking-bay - area
17      ; People
18      alice bob - astronaut
19
20  )
21
22  (:init
23      ; New:
24      ; Crew position
25      (crew-in alice lander1 docking-bay)
26      (crew-in bob lander2 control-room)
27
28      ; The association between rovers and their respective landers
29      (assigned rover1 lander1)
30      (assigned rover2 lander2)
31
32      ; Rover initial deployment states
33      ; Rover1 starts deployed at wp2 (with lander2)
34      (not (unplaced lander1))
35      (lander-at lander1 wp2)
36      (at rover1 wp2)
37      (deployed rover1)
38
39      ; Initial status
40      (lander-free lander1)
41      (empty-memory rover1)
42
43      ; Rover2 starts undeployed (lander2 unplaced ( to be chosen))
44      (unplaced lander2)
45      (not (deployed rover2))
46
47      ; Initial status of 2
48      (lander-free lander2)
49      (empty-memory rover2)
50
51      ; Connectivity between surface locations(Figure 3)
52      (path wp1 wp2)
53      (path wp2 wp1)
54      (path wp2 wp3)
55      (path wp3 wp5)
56      (path wp5 wp3)
57      (path wp5 wp6)
58      (path wp6 wp4)
59      (path wp2 wp4)
60      (path wp4 wp2)
61
62      ; task objective
63      ; Image
64      (image-target image2 wp2)
65      (image-target image3 wp3)
66      ; Scan
67      (scan-target scan4 wp4)
68      (scan-target scan6 wp6)
69      ; Sample
70      (sample-at sample1 wp1)
71      (sample-at sample5 wp5)
72  )
73

```

```
74     (:goal
75       (and
76         ; Indicates that image2 and image3 have been captured
77         (taken image2)
78         (taken image3)
79
80         ; Indicates that scan4 and scan6 have been captured
81         (taken scan4)
82         (taken scan6)
83
84         ; Indicates that sample1 and sample5 have been collected(picked up
85         ↪ and stored)
86         (sample-stored sample1)
87         (sample-stored sample5)
88       )
89     )
```

6 Reference

@ CSP

<https://en.wikipedia.org/wiki/Constraint_satisfaction_problem>

<<https://www.geeksforgeeks.org/artificial-intelligence/constraint-satisfaction-problem/>>

@ Brute-force

<https://en.wikipedia.org/wiki/Brute-force_search>

<<https://www.geeksforgeeks.org/dsa/brute-force-approach-and-its-pros-and-cons/>>

@ Backtracking

<<https://en.wikipedia.org/wiki/Backtracking>>

<<https://www.geeksforgeeks.org/dsa/backtracking-algorithms/>>

@ pyqt

1. <<https://www.riverbankcomputing.com/static/Docs/PyQt6/>>

2. <<https://www.runoob.com/python3/python-pyqt.html>>

3. <<https://blog.csdn.net/u012117917/article/details/41604711>>

4. <<https://zhuanlan.zhihu.com/p/390192953>>

@