KINGS COLLEGE LONDON

4CCS1IAI INTRODUCTION TO ARTIFICIAL INTELLIGENCE

AI Planning Application Coursework
Deliverable 2: Planning Solutions

Wonjoon Seol (k1631098)
Munkhtulga Battogtokh (k1631010)
Britton Forsyth (k1630500)
Eugene Fong (k1630435)

Supervised by
Dr.Dan MAGAZZENI
Dr.Lela KOULOURI

April 21, 2017

1 Deliverable 2: Planning Solutions

1.1 Introduction

Based on the planning domain, this section elaborates on the results obtained by experimenting with the solutions, using the planner OPTIC. Within the experiments various ranges of problem cases and problem scales are considered.

1.2 Planner and Analysis

During this process there were numerous complications and limitations upon creating the domain. Initially the domain consisted of more complex real life actions for example - *join-table*, which was a durative-action used specifically in scenarios where the group of people waiting for a table is greater than the capacity of the table, therefore a staff member could combine available tables together to accommodate larger groups. This also includes actions like *go-to-cashier-to-process-order*, *pass-order-to-chef* and six other similar actions that used the mean travelling-time function for each table, entrance, cashier and kitchen for more realistic staff movements.

However, due to Optics being unable to handle more complex situations efficiently (throwing std:bad_alloc). These actions which created too many branches, were removed from the domain to enhance the efficiency and have ended with our current "core" domain. We were unable to make the problem file too large (having table number > 5 and more than 10 customer groups) under our current system due to the aforementioned OPTIC crash. Still, as far as the coursework is concerned, the current domain has been sufficient to come up with decently effective and meaningful staff management plans (for average scale restaurants as we intended to).

The following graph shows the relation of number of literals to the scheduling time, in seconds:

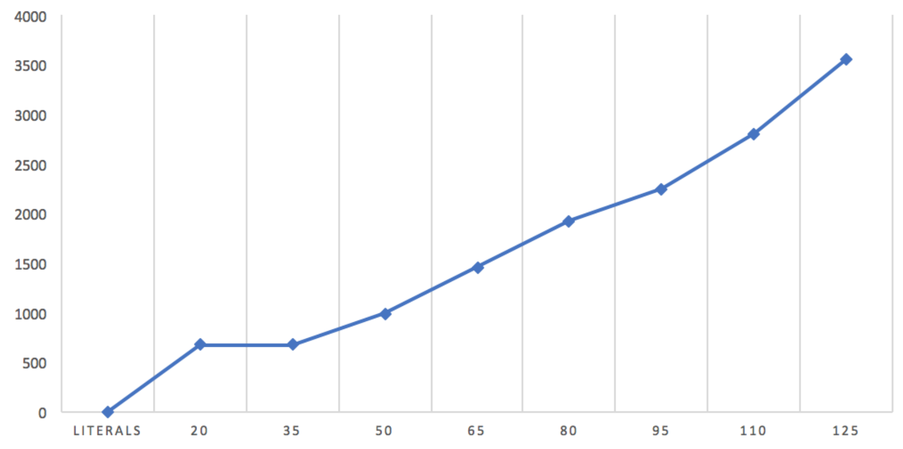


Figure 1: Literal against scheduling time in seconds graph

This models the artificial situation where the restaurant has 3 tables and 2 staff members (Problem 1). In this case, table numbers, table capacity and the number of available staff members were fixed to understand how the limited staff members would be able to handle having almost no customers to busier situation. Starting from 1 customer group, the group numbers were increased by one until the Optic was not able to handle the problem files.

The resulting graph shows a positive correlation - greater the literals are, more time is required by the planner to reach the goal states. The first two points follows our initial expectation, at literal 20 and 35 are the cases when there exists single and two customer groups. The numbers are less than and equal to the number of staff members and less than the available table numbers. Therefore we expected these two points to take the roughly the same time which turned out to be so. More interestingly, once there are greater customer numbers than the staff members the graph is strictly increasing function following approximately linear relationship. In fact the gradient is about 34 in our artificial case when the domain scales up in terms of the customer numbers.

Another test involves finding the optimal number of staff members for a given table numbers. It is obvious that the number of the table must be the upper bound for the staff members, but what about its optimal numbers. In order to find out we used the problem 2 files.

This time, we fixed the number of customer numbers but varied the number of staff members in the restaurants:

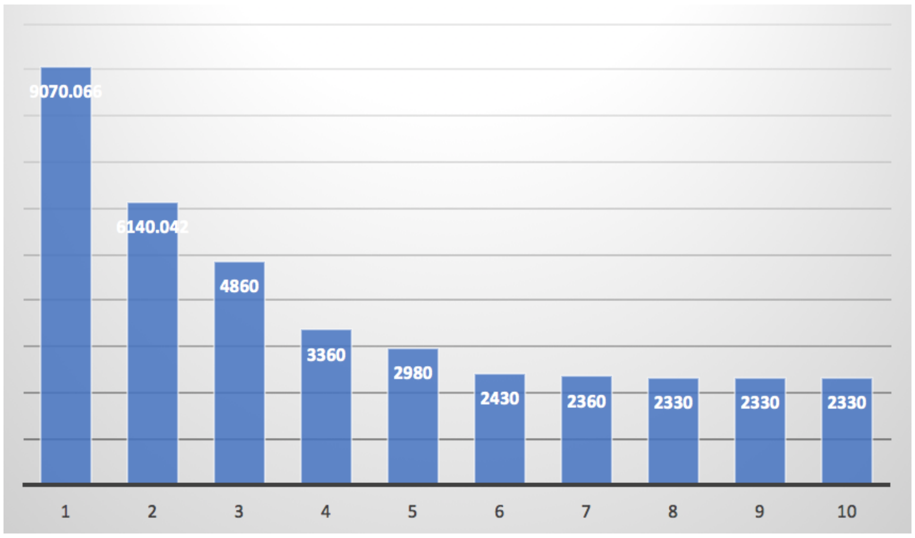


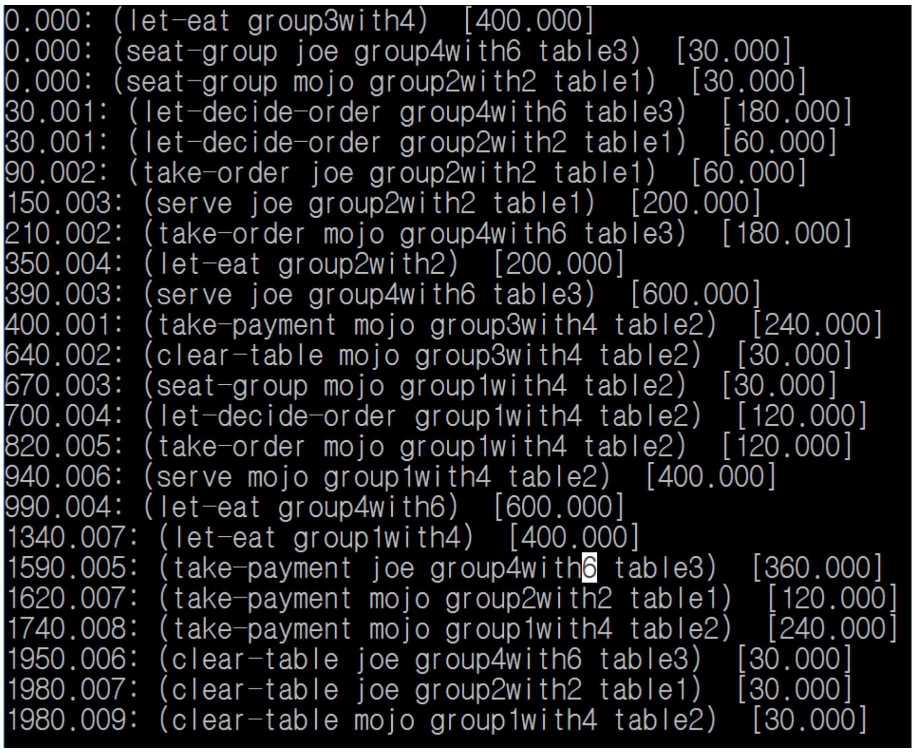
Figure 2: How does staff numbers affect scheduling time?

In this diagram, the restaurant has 10 tables and tests effective staff members range using 15 customer groups, the highest possible customer group number for Optics to handle. Unfortunately, as the problem size were large we couldn't get the minimum time possible using *(:metric minimize(total-time))* as Optics would always crash with this amount of problem size.

Instead, the first solution that appeared from the planner was used. Indeed, the solution was not the fastest possible one but it still demonstrated an useful trend. For each staff member added up to three, there are approximately 30% decrease in the overall processing time. As can be seen from the graph, the optimum number of waiters for a restaurant with 10 tables would be 4. Our test demonstrated that the more staff members would lead to point of diminishing returns.

Despite our rather basic domain, it was apparent that it is suffice to achieve useful information from the model. This problem file demonstrate the advantage of planning, that we are able to gain results that would cost a lot of time and money if tested using the conventional business trials instead.

Lastly, our final test involves problem file 3. This problem demonstrates real-life example of the domain we devised. In this example, the solver program stored inside the restaurant manager's computer is started in the mid day of the restaurant rather than when the restaurant is opened. One of the table - *table 2*, is already occupied by a group of customers already.
The planner is able to successfully analyse remaining available table and the current state of the occupied customer and finds best solution available:



```
0.000: (let-eat group3with4)  [400.000]
0.000: (seat-group joe group4with6 table3)  [30.000]
0.000: (seat-group mojo group2with2 table1)  [30.000]
30.001: (let-decide-order group4with6 table3)  [180.000]
30.001: (let-decide-order group2with2 table1)  [60.000]
90.002: (take-order joe group2with2 table1)  [60.000]
150.003: (serve joe group2with2 table1)  [200.000]
210.002: (take-order mojo group4with6 table3)  [180.000]
350.004: (let-eat group2with2)  [200.000]
390.003: (serve joe group4with6 table3)  [600.000]
400.001: (take-payment mojo group3with4 table2)  [240.000]
640.002: (clear-table mojo group3with4 table2)  [30.000]
670.003: (seat-group mojo group1with4 table2)  [30.000]
700.004: (let-decide-order group1with4 table2)  [120.000]
820.005: (take-order mojo group1with4 table2)  [120.000]
940.006: (serve mojo group1with4 table2)  [400.000]
990.004: (let-eat group4with6)  [600.000]
1340.007: (let-eat group1with4)  [400.000]
1590.005: (take-payment joe group4with6 table3)  [360.000]
1620.007: (take-payment mojo group2with2 table1)  [120.000]
1740.008: (take-payment mojo group1with4 table2)  [240.000]
1950.006: (clear-table joe group4with6 table3)  [30.000]
1980.007: (clear-table joe group2with2 table1)  [30.000]
1980.009: (clear-table mojo group1with4 table2)  [30.000]
```

Figure 3: Solution

The solution appearing from the planner is sound and optimal. The provided optimal strategy for the staff members are the followings:

1. Joe - Seat group 4 to table 3, take order from group 2, serve food to table 3, take payment from group 4, clear table 3 and clear table 1
2. Mojo - Seat group 2 to table 1, take order from group 4, take payment from existing customers (table 2), clear table 2, Seat group 1 to table 2, take order from group 1, serve food to table 2, take payment from group 2 and 1, clear table 2

Tailored restaurant specific planner can be obtained if duration values is altered to restaurant's individual values. Further, Joe currently does 6 actions whereas Mojo does 9 actions. This discrepancy is due to our assumption in the model, negligible travel time for travelling between tables and to the kitchen and entrance. As explained earlier, this is due to our domain model being minimalistic. This can be solved using our original planner prior to pruning it down - add action for travelling between tables, entrance and kitchen using assigning mean travelling time function between them.

The remaining question would be how reliable this model can be in the real life application? In truth, apart from the possible improvements addressed above, there are many shortcomings which this planner doesn't take into account. The main issue is that the planner doesn't take human unreliability into account. For example, the staff member is assumed to be working non-stop without any breaks, hardworking, and would memorise the strategy issued from the planner and follow them in the exact order. Therefore this would be unreasonable strategy in the real life.

Nonetheless, the planner demonstrated its effectiveness by being able to simulate real-life scenario and test multiple hypotheses without conventional expensive business tests. Furthermore, the solution provided here may become more feasible in the future with advances in the technology sector. For example, staff members can be issued with portable computer which calculates similar solutions in real-time in synchronisation with other staff members and display possible actions for them to follow. Ultimately, there will be a point where all staff members are replaced by robots to carry out our current solution reliably.

2 Appendix

2.1 Problem file 1

Problem file 1: 8 Groups, 3 Tables, 2 Staff members

```
(define (problem seat-customers)
    (:domain restaurant-managing)
    (:objects group1with2 group2with2 group3with3 group4with4 group5with2
     group6with3 group7with4 group8with3 - group table1 table2 table3 - table Danielle
     Johnny - staffmember)
    (:init (= (people-count group1with2) 2)
        (= (people-count group2with2) 2)
        (= (people-count group3with3) 3)
        (= (people-count group4with4) 4)
        (= (people-count group5with2) 2)
        (= (people-count group6with3) 3)
        (= (people-count group7with4) 4)
        (= (people-count group8with3) 3)
        (= (table-capacity table1) 4)
        (= (table-capacity table2) 4)
        (= (table-capacity table3) 4)
        (= (table-id table1) 1)
        (= (table-id table2) 2)
        (= (table-id table3) 3)
        (waiting-table group1with2)
        (waiting-table group2with2)
        (waiting-table group3with3)
        (waiting-table group4with4)
        (waiting-table group5with2)
        (waiting-table group6with3)
        (waiting-table group7with4)
        (waiting-table group8with3)
        (table-available table1)
        (table-available table2)
        (table-available table3)
        (member-available Danielle)
        (member-available Johnny)
        )
        (:goal (and (group-complete group1with2)
            (group-complete group2with2)
            (group-complete group3with3)
            (group-complete group4with4)
            (group-complete group5with2)
            (group-complete group6with3)
            (group-complete group7with4)
            (group-complete group8with3)
            )
        )
        (:metric minimize (total-time))
    )
```

2.2 Problem file 2

Problem file 2: 15 Groups, 10 Tables, 10 Staff members

```
(define (problem seat-customers)
    (:domain restaurant-managing)
    (:objects group1with2 group2with2 group3with3 group4with4 group5with2 group6with3
      group7with4 group8with3 group9with4 group10with1 group11with2 group12with3
      group13with3 group14with2 group15with1 - group table1 table2 table3 table4 table5
      table6 table7 table8 table9 table10 - table s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 - staffmember)
    (:init (= (people-count group1with2) 2)
        (= (people-count group2with2) 2)
        (= (people-count group3with3) 3)
        (= (people-count group4with4) 4)
        (= (people-count group5with2) 2)
        (= (people-count group6with3) 3)
        (= (people-count group7with4) 4)
        (= (people-count group8with3) 3)
        (= (people-count group9with4) 4)
        (= (people-count group10with1) 1)
        (= (people-count group11with2) 2)
        (= (people-count group12with3) 3)
        (= (people-count group13with3) 3)
        (= (people-count group14with2) 2)
        (= (people-count group15with1) 1)
        (= (table-capacity table1) 4)
        (= (table-capacity table2) 4)
        (= (table-capacity table3) 4)
        (= (table-capacity table4) 4)
        (= (table-capacity table5) 4)
        (= (table-capacity table6) 4)
        (= (table-capacity table7) 4)
        (= (table-capacity table8) 4)
        (= (table-capacity table9) 4)
        (= (table-capacity table10) 4)
        (= (table-id table1) 1)
        (= (table-id table2) 2)
        (= (table-id table3) 3)
        (= (table-id table4) 4)
        (= (table-id table5) 5)
        (= (table-id table6) 6)
        (= (table-id table7) 7)
        (= (table-id table8) 8)
        (= (table-id table9) 9)
        (= (table-id table10) 10)
        (waiting-table group1with2)
        (waiting-table group2with2)
        (waiting-table group3with3)
        (waiting-table group4with4)
        (waiting-table group5with2)
```

Problem file 2: 15 Groups, 10 Tables, 10 Staff members (continued)

```
        (waiting-table group6with3)
        (waiting-table group7with4)
        (waiting-table group8with3)
        (waiting-table group9with4)
        (waiting-table group10with1)
        (waiting-table group11with2)
        (waiting-table group12with3)
        (waiting-table group13with3)
        (waiting-table group14with2)
        (waiting-table group15with1)

        (table-available table1)
        (table-available table2)
        (table-available table3)
        (table-available table4)
        (table-available table5)
        (table-available table6)
        (table-available table7)
        (table-available table8)
        (table-available table9)
        (table-available table10)

        (member-available s1)
        (member-available s2)
        (member-available s3)
        (member-available s4)
        (member-available s5)
        (member-available s6)
        (member-available s7)
        (member-available s8)
        (member-available s9)
        (member-available s10)
    )

    (:goal (and (group-complete group1with2)
        (group-complete group2with2)
        (group-complete group3with3)
        (group-complete group4with4)
        (group-complete group5with2)
        (group-complete group6with3)
        (group-complete group7with4)
        (group-complete group8with3)
        (group-complete group9with4)
        (group-complete group10with1)
        (group-complete group11with2)
        (group-complete group12with3)
        (group-complete group13with3)
        (group-complete group14with2)
        (group-complete group15with1)
        )
    )
    (:metric minimize (total-time))
)
```

2.3 Problem file 3

Problem file 3: 4 Groups, 3 tables, 2 Staff members, One table already occupied

```
(define (problem seat-customers)
    (:domain restaurant-managing)
    (:objects group1with4 group2with2 group3with4 group4with6 - group table1 table2
     table3 - table Joe Mojo - staffmember)
    (:init (= (people-count group1with4) 4)
        (= (people-count group2with2) 2)
        (= (people-count group3with4) 4)
        (= (people-count group4with6) 6)
        (= (table-capacity table1) 2)
        (= (table-capacity table2) 4)
        (= (table-capacity table3) 6)
        (= (table-id table1) 1)
        (= (table-id table2) 2)
        (= (table-id table3) 3)
        ; group 3 is already seated on table 2, and has just been served
        (seated group3with4 table2)
        (served group3with4)
        (waiting-table group1with4)
        (waiting-table group2with2)
        (waiting-table group4with6)
        (table-available table1)
        (table-available table3)
        (member-available Joe)
        (member-available Mojo)
        )
        (:goal (and (group-complete group1with4)
            (group-complete group2with2)
            (group-complete group3with4)
            (group-complete group4with6)
            )
        )
        (:metric minimize (total-time))
    )
```