

CSE537 Fall 2019 Artificial Intelligence Homework3

Tianao Wang 112819772

November 5, 2019

1. Strategy

In this project, we need to find all the possible solutions. Then compare them with each other to select the minimum cost solution.

Firstly, I use `findall` function in prolog to get all the combination of all packages. During the process of finding each combination, I will check whether it can satisfy the total need and max accepted offer. Only these solutions which can satisfy both constraints can be added to the possible solutions list in prolog.

After finding all the possible solutions, I just need to calculate the cost of each solution and select the minimum cost solution.

2. Implementation

- (a) Use `findall(packages(O, P), offer(O, P, A, U), TotalPackagesWithD)` to find all the packages in the problem and use `compress()` to remove the duplicate.
- (b) Use `findall(X, comb(TotalPackages, X), Listlist)` to get all the combination of packages into one list called Listlist.
- (c) At the same time, not all the combination can be added into Listlist. I add two constraints along with `comb(TotalPackages, X)` in (b).
The first constraint is that the solution can meet max accepted offer, I need to add the number of packages of each operators can check whether is satisfy the max accepted offer. The second constraint is that the solution can meet the total need. I add up the need which each solution can satisfy and compare them to the need in the problem. Only these solutions which can satisfy the need can be added.
- (d) Finally, I calculate the cost of each legal solution. And get the minimum index of solution, then print the least cost and the solution.

3. How the run the program

There are two steps to run the program. There are two .pl files which are data.pl and trip.pl. In the file called data.pl, it stores the facts like this.

```
data.pl x trip.pl — C:\...\03 x
1 % packages of operator 1
2 offer(1,1,1,1).
3 offer(1,1,2,2).
4 price(1,1,8).
5 % packages of operator 2
6 offer(2,1,2,1).
7 offer(2,1,3,1).
8 price(2,1,9).
9
10 offer(2,2,3,2).
11 offer(2,2,4,1).
12 price(2,2,10).
13 % packages of operator 3
14 offer(3,1,3,1).
15 offer(3,1,4,1).
16 price(3,1,6).
17 % needs
18 need(2,1).
19 need(3,3).
20 need(4,1).
21 maxacceptedoffer(1).|
```

If we want to change the facts for different tests. We just need to change data.pl file.

The other file is trip.pl which store the main program of this project. We can run the program in the following order.

|? — [data].

|? — [trip].

|? — *totalcost*(X).

4. Test

Test two examples on the website.

Example1

```
XSB Version 3.8.0 (Three-Buck Chuck) of October 28, 2017
[x64-pc-windows; mode: optimal; engine: slg-wam; scheduling: local]
[Build date: 2017-10-31]

| ?- [data].
[Compiling .\data]
[data compiled, cpu time used: 0.0620 seconds]
[data loaded]

yes
| ?- [trip].
[Compiling .\trip]
++Warning[XSB]: [Compiler] .\trip : Singleton variable A in a clause of totalcost/1
++Warning[XSB]: [Compiler] .\trip : Singleton variable U in a clause of totalcost/1
++Warning[XSB]: [Compiler] .\trip : Singleton variable A1 in a clause of judgeEach/2
++Warning[XSB]: [Compiler] .\trip : Singleton variable U1 in a clause of judgeEach/2
++Warning[XSB]: [Compiler] .\trip : Singleton variable T in a clause of judgeEach/2
++Warning[XSB]: [Compiler] .\trip : Singleton variable A2 in a clause of judgeEach/2
++Warning[XSB]: [Compiler] .\trip : Singleton variable U2 in a clause of judgeEach/2
++Warning[XSB]: [Compiler] .\trip : Singleton variable SortedChosenNeed in a clause of judgeNeed/2
++Warning[XSB]: [Compiler] .\trip : Singleton variable P in a clause of extract/2
++Warning[XSB]: [Compiler] .\trip : Singleton variable Pivot in a clause of partition/4
% Specialising partially instantiated calls to comb/2
% Specialising partially instantiated calls to judgeEach/2
% Specialising partially instantiated calls to length/2
% Specialising partially instantiated calls to decode/2
% Specialising partially instantiated calls to compress/2
[trip compiled, cpu time used: 0.0160 seconds]
[trip loaded]

yes
| ?- totalcost(X).
Accepted offers: package 1 of tour operator 1, package 2 of tour operator 2, package 1 of tour operator 3,
X = 24;

no
| ?- _
```

Example2

```
| ?- [data].
[Compiling .\data]
[data compiled, cpu time used: 0.0000 seconds]
[data loaded]

yes
| ?- [trip].
[trip loaded]

yes
| ?- totalcost(X).
Accepted offers: package 2 of tour operator 1, package 2 of tour operator 2, package 3 of tour operator 2, package 1 of tour operator 3,
X = 25;

no
| ?- _
```