Assignment 3- Constraint Satisfaction Problem

Alam, Praveen ; Suresh, Muthukumar

March 22, 2015

Contents

1 Introduction

In this assignment, we look at the Constraint Satisfaction Problem for the task of TA assignment for courses. It is a hard problem but conforms well to the task of CSP.
`There are few hard constraints that need to be taken care of:`

1. Timing constraints are hard constraints, they have to be taken care of without fail. So, if there is a recitation clashing with a class that the TA has to attend then that TA can never be assigned to the course. Same applies for class timings that the TA needs to Attend
2. A TA cannot be assigned to more than a total of 1 TA assignment.
3. A TA cannot be assigned to classes where the class timings clash.

`There are also a few soft constraints:`

1. A TA has certain skills. We make the assumption that the TA should have 30% of the skills matching. This is a reasonable assumption to make since we want atleast some match in skills.

`Other design specifications:`

1. Our solution chooses the best possible assignment that satisfy both the hard and soft constraints.
2. Best score is classified as follows:
   (a) Our main criteria is that we want maximum number of courses to have a TA.
   (b) Secondly, if two courses assign the same number of courses, then we check if the maximum number of TAs are assigned. These two constraints ensure that maximum possible TA assignments are made conforming to the hard and soft constraints.

2 Design Specification

- We have implemented it in python 3.4
- It takes as a file the input file in the format specified in the question. Important details: each of the terms should be separated by ,¡space¿ . Also, each of the tables are separated by a new line. The file name is hardcoded in the main(can be changed).
- As input it takes which mode we want to run the program. plain backtracking, enter 1. For backtracking with forward chaining, enter 2, For backtracking with Constraint propogation, enter 3. For everything, press 4.

`We describe our implementation:`

- We use dictionaries as our primary datastructure. That gives us good access time. One for TAs and One for Courses
- We follow a basic backtracking strategy, assigning TAs to courses. starting with 1 and then 0.5 and seeing the best possible score as described in the previous section. At the same time checking for constraints depending on the mode of implementation. We keep updating a global best score.
- Other details are pretty evident from the implementation.

3 analysis

NOTE: In python, the dictionary ordering is not consistent, as specified in the python documentation. So the cost may not be same each time as the nodes may be expanded in different order. We need to account for this change between different runs.

3.1 perfect assignment-all assigned

TESTSET: testdata1.txt Output:

```
(BT - Back tracking, FC - Forward Checking CS - Constraint Satisfaction
 : 1 - BT
2 - BT + FC
3 - BT + CS
4 - BT + FC + CS.
 Enter your option: 1, 2, 3, 4.
 >>4
Assignment Complete
Course Requirements:

CSE114 0.5
CSE110 2
CSE307 1.5
CSE101 2
CSE306 0.5
COURSE Assignment:

CSE114--->  --  TA6~Capacity:0.5  --  NOT ASSIGNED : 0.0
CSE110--->  --  TA6~Capacity:0.5  --  TA8~Capacity:1.0  --  TA7~Capacity:0.5  --  NOT ASSIGNED : 0.0
CSE307--->  --  TA7~Capacity:0.5  --  TA2~Capacity:1.0  --  NOT ASSIGNED : 0.0
CSE101--->  --  TA1~Capacity:1.0  --  TA5~Capacity:1.0  --  NOT ASSIGNED : 0.0
CSE306--->  --  NOT ASSIGNED : 0.5
TA Assignment:
TA4--->
TA10--->
TA6--->  --  CSE114~Capacity:0.5  --  CSE110~Capacity:0.5
TA8--->  --  CSE110~Capacity:1.0
TA2--->  --  CSE307~Capacity:1.0
TA7--->  --  CSE110~Capacity:0.5  --  CSE307~Capacity:0.5
TA1--->  --  CSE101~Capacity:1.0
TA9--->
TA3--->
TA5--->  --  CSE101~Capacity:1.0

NODES EXPANDED:
 45
```

Analysis:
As we can see, we have enough TA that meet all the constraints, so the algorithm works perfectly. We can really see our cost function working here. BEcause there are multiple assignments that assign all courses but the cost function also makes sure that the minimum number of courses have TA not assigned.
With plain backtracking, NODES EXPANDED: 94
With backtracking + FC, Nodes Expanded: 44
With backtracking + FC+CS, nodes expanded 45.
Because python does not keep consistent the ordering of keys. We cannot be assured that the assignments made are same in all runs. So nodes expanded may vary. It is not an accurate heuristic to judge the methods. We would like to generally note the performance of the algorithm.

3.2 imperfect assignment - Not enough TAs

Test data2 Output:

```
(BT - Back tracking, FC - Forward Checking CS - Constraint Satisfaction
 : 1 - BT
2 - BT + FC
3 - BT + CS
```

```
4 - BT + FC + CS.
 Enter your option: 1, 2, 3, 4.
 >>4
Assignment Complete
Course Requirements:

CSE101 2
CSE307 1.5
CSE306 0.5
CSE110 2
CSE114 0.5
COURSE Assignment:

CSE101--->  --  TA1~Capacity:1.0  --  TA2~Capacity:1.0  --  NOT ASSIGNED : 0.0
CSE307--->  --  NOT ASSIGNED : 1.5
CSE306--->  --  NOT ASSIGNED : 0.5
CSE110--->  --  NOT ASSIGNED : 2
CSE114--->  --  NOT ASSIGNED : 0.5
TA Assignment:
TA1--->  --  CSE101~Capacity:1.0
TA2--->  --  CSE101~Capacity:1.0

NODES EXPANDED:
 7
```

ANALYSYS: As we can see that when not enough TAs are present, it tries to assign the best possible for each course and leaves it at that.

4 conclusion

NOTE: Due to confusion on Piazza, the problem was over-complicated.
CSP is a constraint satisfaction problem. If constraints are not met, then the CSP is supposed to fail and that is what our program does. Example: If there is a course which cannot be assigned any TA, then the CSP fails that is the right thing to do. If not, it becomes a pure DFS because none of the forward propagation and Constraint propagation make sense. What we mean is:- if we allow certain courses not to be mapped even when there are no TAs present, and we continue trying to match other cases, there is no bounding case for the constraints and no backtracking, that is clearly not the essence of CSP. CSP is a yes-no problem, not a find-best problem and not treating it as such is not the right thing to do.
We have simple rules at when an assignment is successful:

1. When we see that there are no more TAs but there are courses. That is completely fine.
2. When all courses are mapped.

The assignment should fail when:

1. A course cannot be assigned to any TA. That is clearly a bounding condition. If we allow this, then it stops being a CSP problem. Again, we have a value of the number of skills that a TA shouuld possess, so varying this value can give us an assignment with 0 skills matching but failure of all constraints leads to a failure and by the idea of CSP, this is the right thing to do because success or failure of a branch is decided by its satisfaction of constraints.