Insurance Referee Assignment

Sahith Reddy Gaddam
Arizona State University
Sgaddam7@asu.edu

Problem Statement

An insurance company needs to check if customer claims in insurance cases are justified. To this end, it sends referees to different locations to inspect damages (e.g. damaged cars) and write reports. The insurance company employs its own referees (internal referees) but can also authorize external referees to handle a case. The overall task is to assign referees to insurance cases according to various hard and weak constraints. Referees have a maximum workload per day, which is given by the maximum number of working minutes per day. Referees are in charge of certain geographical regions, which are identified by their postal code, where the assignment of referees to regions is gradual to support preferences (e.g. referees must not, can but prefer not to, or prefer to take a case in a certain region, etc); the degree of preference is represented by an integer. Moreover, referees can be specialized in certain domains (e.g. passenger cars, trucks, etc). As for the regions, the assignment of referees to types of cases supports degrees of preferences. While internal referees receive a fixed salary per month, external referees are paid per case, where the amount of payment depends on the case. Insurance cases are defined by the expected effort to handle it (number of working minutes), the amount of damage (in Euros), and the payment an external referee receives if assigned to this case. Note that these attributes do not necessarily correlate, i.e., there can be cases with smaller expected effort and/or amount of damage but higher payment for the referee. Our scenario focuses on an assignment of referees to cases on a single working day. An instance consists of the set of available referees and the set of cases to handle. The task is to assign exactly one referee to each case (but a referee can be assigned to multiple cases) according to the following constraints.

Project Background

To find assignments which are optimal in popular coding languages which support a bit of imperative programming is usually complicated. Imperative languages usually require a constraint implementation and enumerating the search space implementation in some sort of effective way. Instead of that, only encoding the constraints would be preferrable. For e.g., Prolog, which is a declarative programming language, reduces the size of the code significantly for search space enumeration and instead allows a person to concentrate on the constraint encoding. I mainly focused on Clingo over Prolog because of the following reasons:

- Clingo surely terminates at some point, but Prolog will not.
- Programmer must require having knowledge of complicated operators such as the branch cut operator for him to be fluent in search space enumeration.
- Such properties are achieved by clingo through Answer Set Programming, a form of declarative programming and utilizes stable model semantics

Approach in Solving the Problem

To overcome constraint problems, ASP only requires a little planning compared to others. There are 2 major procedures in ASP:

Generation of possible stable models.
For all possible assignments to be generated, all referees are needed to be assigned to one case. To accomplish this, The following syntax is used:

Testing the stable models

Models are eliminated and scored in the testing part. There are 2 kinds of constraints which are: hard constraints and weak constraints.

Every model which is generated is tested against every constraint. A model isn't considered a viable model if the constraint is a hard and the model doesn't satisfy it. If it is a weak constraint, it can be violated and fine is assigned with each violation. At the end, a model which satisfies every hard constraint and as well minimizes the fine of the weak constraints violation is the one that is chosen. The company has given the 9 constraints to be considered as mentioned below. Adding to that, 10 mock scenarios were given to check the accuracy of the implementation of those 9 constraints.

Constraints

Maximum Working Time

Maximum Working minutes of a referee should not be exceeded by the defined workload, where the workload is the addition of the efforts of all cases assigned to this referee. Each referee has a maximum workload associated with their id.

:- referee(rid, type, max_workload, prev_workload, prev_payment),
#sum { effort, cid
: case(cid, type, effort, damage, postc, payment),
assign(cid, rid) } > max_workload.

Region Hard Constraint

Referees are assigned to certain regions and they can only work in those regions. This can be encoded as not having any preference for the location. The preference score is encoded as an integer from zero to three. Here zero means the referee cannot work in that area and three means the location for that referee is of highest preference. Moreover, if none of the preference is specified for the location, then there is no rule. This should be considered as the referee has a preference of 0 for the region. 2 rules are required to enforce this,

First rule:
:- assign(cid, rid),
case(cid, type, effort, damage, postc, payment),
not prefRegion(rid, postc, pref).

This ensures that if a referee doesn't have a preference it will set the score to 0.

Second Rule:
:- assign(cid, rid),
case(cid, type, effort, damage, postc, payment),
prefRegion(rid, postc, pref).

The above rule ensures that it cannot assign the referee to a location which has a preference zero score.

Work Preference Hard Constraint

A case cannot be assigned to a referee who is not in charge of the particular type of the case. This can be encoded like the referee has no preference for that type of job. It breaks down to as the Region Hard Constraint and the two rules are almost identical to the previous two rules with minor changes. We need to replace prefRegion(rid, postc, pref) with prefType(rid, caset, pref).

Damage Constraint

If the amount of damage exceed the given threshold then that case can only be assigned to internal referees. In this type of cases the company can specify the max amount of damage an outsider referee can handle. An external employee can be prevented from working on a high damage by using the following rule:

:- externalMaxDamage(ExtMaxDamage),
referee(rid, type, max_workload, prev_workload, prev_payment).
case(cid, type, effort, damage, postc, payment)
Damage > ExtMaxDamage, assign(cid, rid).

Weak Constraint

Prefer Internal Referees

The company prefers internal referees over external referees to minimize the cost. We can assign appropriate weights to each weak constraints as follows:

$17C_A + 6C_B + 10C_C + 30C_D + 30C_E$

Here C_i refers to a specific weak constraint and the coefficient in front of each weak constraint depicts how important the weak constraint is.

All Referees getting balanced workload

This constraint cannot be satisfied all the time. Utilizing the statistics as an equivalent for this problems helps us.

Balanced Payment for External Referees

The code for balancing the payment for external referee is similar to the previous constraint. We need to handle two instances where the domain is strictly restricted to the external referee.

Type Preference Optimization

Each employee is assigned a priority value from 0 to 3 for the type of work they prefer. Where 0 means that the referee doesn't want to take that job or least interested in taking that job and 3 indicates that the referee is most interested in taking that job. We need to program a penalty for assigning a least priority work to the referee and the code needs to be in a way that the ASP chooses the model which minimizes the penalty in the mentioned way.

Main Results and Analysis

The company provided a few mock scenarios to test the validity of the program. As discussed in the approach, our implementation of the code results in the expected output.

Results:

```
Starting test:
Command: clingo scenario_eight.lp scenario_one.lp --quiet=1
===========================

Test Output:
clingo version 5.3.0
Reading from scenario_eight.lp ...
Solving...
Answer: 1
assign(14,27) assign(15,27)
Optimization: 227988
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 227988
Calls        : 1
Time         : 0.060s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.01s)
CPU Time     : 0.063s


==== TEST RESULTS ====
TEST PASSED!
EXPECTED: assign(14,27) assign(15,27)
ACTUAL: assign(14,27) assign(15,27)
```

Conclusion and Self Assessment

The weights assigned to the penalties in hard constraints and weak constraints are randomly chosen by the company. After better research the model can be more efficient by applying reliable weights of penalties. Regarding the self assessment, I was fumbling a bit at the beginning of the project as time passed by I followed every lecture and explored some online resources to gain some exposure on Clingo and made my way through the project.

Opportunities for Future Work

As the advancements in Machine Learning models, it can be a good practice by incorporating the classification algorithms in the constraints as well which can well predict the weights of the penalties. This can be a better practice instead of randomly choosing the weights and tuning them manually. Scenario six took relatively more time to run and hence having the clingo run on distributed memory could make it utilize memory available on different clusters and can run faster even in more complex cases.

References

[1] *On the Application of Answer Set Programming to the Conference Paper Assignment Problem* **by** *Giovanni Amendola, Carmine Dodaro*

[2] Answer Set Programming: A Primer By Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner