Insurance Referee Assignment

Jeffrey Pan

Problem Statement

Insurance companies will send out referees to inspect insurance claims and to write reports. However, assigning referees to various cases is hard to optimize.

Each referee has preferred case types and preferred regions to work in. For example, it would not be viable to assign a referee to a location that is very far away; instead, referees should be working in a specific area. But what about locations that are a little far but not *too* far? A "soft constraint" could be implemented to assign the case to a referee if there are no better options.

There are also other factors to keep in mind. A referee is also limited to the amount of time he/she can work each day. In addition, when there are too many cases for internal referees to handle, external referees can be hired as a backup. All of these must be kept in mind while keeping the workload relatively even among referees.

Project Background

The complexity of assigning referees to cases is best suited for a machine to determine, instead of doing it manually each day. The goal of this project is to use Answer Set Programming where a list of referees and cases can be input, and an optimal assignment of cases is returned.

Answer Set Programming is a type of programming used to solve logic problems. Restrictions are defined, a scenario is input, and the program returns a list of possible solutions. A "hard restriction" could be something like: "Cases should not be assigned to referees that are not in charge of that region." This style of programming is very flexible because after the restrictions are defined, various input scenarios can be given to it. This differs from traditional programming where each rule has be individually programmed and a system of checking for viable scenarios must be coded. Traditional programming requires many more lines of code and are often less flexible in terms of inputs. In a "referee assignment" problem like this, the types of cases and available referees differ from a day-to-day basis.

Answer Set Programming can be further tuned to not only give all possible solutions, but to also optimize certain variables. For example, although many solutions may exist, I want to minimize the total payment to external referees.

Clingo is a specific type of Answer Set Programming system and is what was used to approach this referee assignment problem.

Approach

The first step to any ASP program is to create a general search space. This was simply defined by assigning each case to exactly one referee.

```
{assign(CID,RID) : referee (RID, RTYPE,
MAX_WORKLOAD, PREV_WORKLOAD,
PREV_PAYMENT)} = 1 :- case(CID, CTYPE, EF-
FORT, DAMAGE, POSTC, PAYMENT).
```

This statement can be roughly read as "For each case, assign exactly one referee to it."

Next, four hard constraints were created (Code can be found in the appendix):

1. Total workload of a referee could not exceed their "MAX_WORKLOAD"
2. Cases should not be assigned to referees that are not in charge of that region
3. Cases should not be assigned to referees that are not in charge of that type of case
4. Cases with damage exceeding the specific external max damage must be given to internal referees (as opposed to external referees)

Lastly, a mixture of five weak constraints must be optimized to reduce the total cost based on a given formula.

1. C_A = Minimize total payment to external referees
2. C_B = Balance overall payments to external referees
3. C_C = Balance overall workload to all referees
4. C_D = Assign cases to referees based on their case type preference
5. C_E = Assign cases to referees based on their postal code type preference

Results and Analysis

While it took lots of tweaking to create the code, thanks to Answer Set Programming, I simply had to define the restrictions I was looking for. The overall program only had 21 lines of code. The most difficult part was creating weak constraints.

I created formulas to define each of the five weak constraints, the most time consuming being calculating the total

divergence of the payment to external referees and workload given. To optimize the overall costs, I simply had to have clingo minimize the costs based on the equation given:
Overall cost: $16\cdot C_A + 7\cdot C_B + 9\cdot C_C + 34\cdot C_D + 34\cdot C_E$

After writing the code, all that was needed was to run it in conjunction with the examples or any scenario given. While the code runs inefficiently, it still provides an optimal scenario.

Conclusion

Answer Set Programming is a powerful approach to solving complex problems with minimal coding involved. With only 21 lines of code, the task of assigning hundreds of different cases to various referees can be easily done by a computer program.

The program I have written runs very inefficiently. Perhaps a future project would be to focus on optimizing the code. In addition, the code doesn't feel very clean, with a plethora of variable names that aren't descriptive or easily readable.

Opportunities for Future Work

In addition to optimizing the code as previously mentioned, I found the concept of Answer Set Programming to approach common problems fascinating. It follows much more closely with human logic and reasoning than many other programing languages. I hope to approach many more problems in the future with ASP.

Appendix 1.

%GENERATE a Search Space

```
%For each case, max one assignment to a ref
{assign(CID,RID) : referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT)} = 1 :- case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT).
```

%Hard Constraints

```
%TotalWorkload of a referee (sum of effort of all cases assigned to that ref) but not exceed MAX_WORKLOAD
:- TotalWorkload = #sum{EFFORT,CID : case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT), assign(CID,RID)}, referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), TotalWorkload > MAX_WORKLOAD.

%Do not assign cases to referees that are not in charge of that region (aka pref = 0)
:- assign(CID,RID), case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT), referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), prefRegion(RID,POSTC,PREF), PREF==0.

%Do not assign cases to referees that are not in charge of that type of case (aka pref = 0)
:- assign(CID,RID), case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT), referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), prefType(RID,CTYPE,PREF), PREF==0.

%Cases with DAMAGE exceeding externalMaxDamage(d) must be given to internal referees (aka cannot be given to external referees)
:- assign(CID,RID), case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT), referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), externalMaxDamage(MAXDAMAGE), DAMAGE > MAXDAMAGE, RTYPE==e.
```

%Optimization Criteria

```
%Optimize based on:
%req1. Minimize total payment to external referees
%req2. Balance overall payments to external referees. Minimize divergence from the average
%req3. Balance overall workload to all referees. Minimize divergence from the average
%req4. Prioritize cases with higher postal preferences. Minimize "cost", calculated by (3 - pref)
%req5. Prioritize cases with higher region preferences. Minimize "cost", calculated by (3 - pref)
%Minimize overall optimization formula: 16*req1 + 7*req2 + 9*req3 + 34*req4 + 34*req5

%req1. Calculate total payment to external referees
totalExternalPaymentNoPrev(TEPNP) :- TEPNP = #sum{PAYMENT : assign(CID,RID), case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT), referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), RTYPE==e}.

%req2. Balance overall payments to external referees. Minimize divergence from the average
externalPaymentNoPrev(RID, EPNP) :- EPNP = #sum{PAYMENT, CID : assign(CID,RID), case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT)}, referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), RTYPE==e.
externalPayment(RID, EP) :- EP = EPNP + PREV_PAYMENT, externalPaymentNoPrev(RID, EPNP), referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), RTYPE==e.
totalExternalPayment(TEP) :- TEP = #sum{EP, RID : externalPayment(RID, EP)}.
numOfExternalRefs(NOER) :- NOER = #count{RID : referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT), RTYPE==e}.
avgExternalPayment(AEP) :- AEP = TEP / NOER, totalExternalPayment(TEP), numOfExternalRefs(NOER).
paymentDivergence(PD) :- PD = #sum{|EP - AEP|, RID : avgExternalPayment(AEP), externalPayment(RID, EP)}.

%req3. Balance overall workload to any referee. Minimize divergence from the average
workloadNoPrev(RID, WNP) :- WNP = #sum{PAYMENT, CID : assign(CID,RID), case(CID,CTYPE,EFFORT,DAMAGE,POSTC,PAYMENT)}, referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT).
workload(RID, W) :- W = WNP + PREV_PAYMENT, workloadNoPrev(RID, WNP), referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT).
totalWorkload(TW) :- TW = #sum{W, RID : workload(RID, W)}.
numOfRefs(NOR) :- NOR = #count{RID : referee(RID,RTYPE,MAX_WORKLOAD,PREV_WORKLOAD,PREV_PAYMENT)}.
```

```
avgWorkload(AW) :- AW = TW / NOR, totalWork-
   load(TW), numOfRefs(NOR).
workloadDivergence(WD) :- WD = #sum{|W - AW|, RID :
   avgWorkload(AW), workload(RID, W)}.

%req4. Calculate case type preference cost
casePrefCost(CPC) :- CPC = #sum{3 - PREF, CID : as-
   sign(CID,RID),         case(CID,CTYPE,EFFORT,DAM-
   AGE,POSTC,PAYMENT),                           ref-
   eree(RID,RTYPE,MAX_WORKLOAD,PREV_WORK-
   LOAD,PREV_PAYMENT),
   prefType(RID,CTYPE,PREF)}.

%req5. Calculate postal code type preference cost
postPrefCost(PPC) :- PPC = #sum{3 - PREF, CID : as-
   sign(CID,RID),         case(CID,CTYPE,EFFORT,DAM-
   AGE,POSTC,PAYMENT),                           ref-
   eree(RID,RTYPE,MAX_WORKLOAD,PREV_WORK-
   LOAD,PREV_PAYMENT),                           pre-
   fRegion(RID,POSTC,PREF)}.

#minimize{16*TEPNP + 7*PD + 9*WD + 34*CPC +
   34*PPC, CID, RID : assign(CID,RID), totalExternalPay-
   mentNoPrev(TEPNP), paymentDivergence(PD), work-
   loadDivergence(WD), casePrefCost(CPC), postPref-
   Cost(PPC)}.

#show assign/2.
```