Insurance Referee Assignment

Kaiting Sin
Arizona State University
ksin3@asu.edu

Problem Statement

The problem of referee assignment optimization for a single working day aims to determine an optimal assignment of referees to cases based on their availability, workload, payment, and preferences. The input consists of a set of available referees, a set of cases to be handled, a threshold for external referees, and preferences of regions and types of cases. Referees are defined by their unique identifier, type (internal or external), maximum workload per day, previous workload, and previous payment. Cases are defined by their unique identifier, type, expected effort, damage, postal code of the location, and payment. The maximum damage of cases assigned to external referees is given by a single fact. The primary goal of this problem is to assign a single referee to each case while minimizing the total workload of the referees, maximizing the total payments received by external referees, and satisfying the preferences of regions and types of cases for each referee. This requires a thorough understanding of the preferences and constraints involved in the problem, as well as an effective algorithm for solving the problem. It is crucial to ensure that all hard constraints are satisfied, including the maximum workload for each referee and not allowing referees to handle certain cases. Failure to meet these constraints can result in an invalid solution.

Project Background

The Insurance Referee Assignment Problem involves assigning referees to cases for an insurance company while considering various constraints such as maximum workload, geographical region, case type, and hiring costs for external referees. This problem can be efficiently modeled as an Answer Set Programming (ASP) program using Clingo, which takes as input a set of referees, cases, and assignment constraints.

Answer Set Programming (ASP) is a declarative programming paradigm exceptionally suited for solving complex combinatorial optimization problems. It has found widespread use in domains such as planning, scheduling, and configuration. In ASP, a problem is represented by a logic program consisting of rules and constraints, with the objective of finding a set of assignments (called "answer sets") that satisfy these rules and constraints. ASP enables the specification of constraints and preferences, allowing for the generation of optimal or near-optimal solutions.

Basic Knowledge of ASP

ASP consists of a set of rules and constraints that describe a problem. Each rule is composed of a head and a body, connected by the symbol ‘:-‘. The head represents a derived fact, while the body contains a conjunction of literals (i.e., atoms or their negations). The rule is read as an implication, meaning that if the body of the rule is true, the head must also be true. The goal is to find a set of literals (an "answer set") that satisfy all the rules and constraints.

Basic Knowledge of Clingo

Clingo is a cutting-edge ASP solver adept at efficiently finding solutions for large-scale optimization problems. Clingo works by taking an ASP program as input and then attempting to find an answer set that satisfies all the program's rules and constraints.

The input to Clingo consists of several types of rules, including facts, rules, and constraints. Facts are basic pieces of information that hold true in the problem, while rules are used to describe relationships between atoms. Constraints are rules that must be satisfied by any valid solution. Optimization statements can also be used to express preferences and guide the solver towards optimal solutions.

By employing Clingo to solve the Insurance Referee Assignment Problem, insurance companies can optimize their

assignment process and deliver better service to their customers.

Approach in Solving the Program

Utilize a Clingo logic program to define constraints and preferences for the insurance referee assignment problem. The article by R. Shane demonstrates the application of KRR in the insurance industry, creating a knowledge-based system to assist with disputes. This system acts as a fair and unbiased referee, handling complex terminology with a knowledge base of rules, facts, and precedents. Leverage this knowledge to solve the assignment problem in following steps.

Model the Problem and Define Decision Variables

To solve the Insurance Referee Assignment Problem with Clingo, a logic-based representation of the problem needs to be created. This involves identifying the decision variables, defining the input format, and specifying the constraints and objectives. In this problem, the decision variables are the assignments of referees to cases, which can be represented as the predicate assign(cid, rid) where cid is the id of the case and rid is the id of the referee. In the generation step of the Clingo program, all possible assignments are enumerated using the following rule:

```
{assign(cid, rid): referee(rid, _, _, _, _)} = 1 :- case(cid, _, _, _, _, _).
```

This rule ensures that each case is assigned to exactly one referee, and it considers all available referees in the enumeration process.

Specify the Constraints

We need to specify the hard and weak constraints of the problem. The hard constraints are constraints that must be satisfied, and the weak constraints are objectives that we would like to optimize subject to the hard constraints.

Hard Constraints

The hard constraints for the insurance referee assignment problem encompass various factors to ensure efficient and accurate case distribution. Firstly, workload constraints are implemented to guarantee that each referee's assigned workload remains within their maximum working capacity, preventing overburdening. Secondly, regional preference constraints are established to ensure that referees are only assigned cases within the regions they are responsible for, fostering expertise and familiarity. Finally, type preference constraints are in place to mandate that cases are matched with referees who possess the requisite specialization in the corresponding field, promoting competence and effective resolution. Additionally, cases involving damage amounts surpassing a specified threshold are exclusively assigned to internal referees, ensuring the necessary expertise and scrutiny for high-stakes cases.

This is an example of a hard constraint that limits the maximum number of working minutes for a referee. The actual workload of a referee is defined as the sum of the efforts of all cases assigned to that referee.

```
:- referee(rid,_,maxWork,_,_), S=#sum{effort: case(cid,_,effort,_,_,_),referee(rid,_,_,_,_), assign(cid,rid)}, S>maxWork.
```

The rule reads as follows: "If there exists a referee with a maximum working time of 'maxWork' minutes, and the sum of the efforts of all cases assigned to this referee is greater than 'maxWork', then the answer set is invalid."

Weak Constraints

The weak constraints consist of minimizing external costs, which entails assigning cases to internal referees to minimize the expenses associated with external referees. Moreover, there is the objective of fair payment, which ensures that payment is equitably distributed among external referees. Lastly, there is the objective of fair workload, which aims to balance the workload of all referees, both internal and external. This is achieved by penalizing the divergence from the average overall workload over all referees. Additionally, referees are assigned to cases based on their preference for case type and region, which aims to maximize their satisfaction and expertise in handling cases.

The provided code implements a weak constraint that ensures fair workload distribution.

```
w_rid(rid, w) :- referee(rid, _, _, prev_workload, _), w = prev_workload + new_w,
new_w = #sum{effort : case(cid, _, effort, _, _), assign(cid, rid)}.
avg_w(avg) :- avg = sum / refereeCount,
sum = #sum{w : w(rid, w)},
refereeCount = #count{rid : referee(rid, _, _, _, _)}.
:~ avg_w(avg), w(rid, w). [9*|avg-w|, rid]
```

The weak constraint aims to balance the overall workload of referees by penalizing the deviation of each referee's workload from the average workload. The overall workload is the sum of the referee's previous workload and the workload of newly assigned cases.

The predicate ‘w_rid’ calculates the overall workload ‘w’ of each referee ‘rid’ by summing their previous workload ‘prev_workload’ and the workload of newly assigned cases ‘new_w’. The ‘avg_w’ predicate computes the average workload ‘avg’ over all referees.

The weak constraint is then formulated as ':~ avg_w(avg), w_rid(rid,w). [9*|avg-w|,rid]', which penalizes the divergence from the average workload with costs '9*|avg-w|'. The 'rid' annotation ensures that the penalty is assigned to each referee individually.

Test the Model

The input for the Insurance Referee Assignment Problem includes a set of available referees, a set of cases to be handled, a threshold for external referees, and preferences of regions and types of cases. Referees are defined as facts including their unique identifier, type (internal or external), maximum working minutes per day, previous workload, and previous payment. Cases are defined as facts including their unique identifier, domain, expected working minutes, amount of damage, location postal code, and payment for external referees. The threshold for external referees is given as a single fact indicating the maximum amount of damage they can handle. Preferences of regions and case types are defined as facts including the referee identifier, region or type, and the level of preference ranging from 0 (not allowed) to 3 (highest preference). If no preference is specified for a referee and a region or case type, it is assumed to be 0 by default.
The optimization goal is to minimize the overall cost (C) calculated as:

$$C = 16C_A + 7C_B + 9C_C + 34C_D + 34C_E$$

where:

C_A = Total payments to external referees, aiming to minimize their cost.
C_B = Balance external referee payments, penalizing deviations from the average payment.
C_C = Balance referee workloads, penalizing deviations from the average workload.
C_D = Encourage case type preferences, incurring costs for non-preferred case types.
C_E = Encourage regional preferences, incurring costs for non-preferred regions, while respecting hard constraints for pref = 0.

By minimizing this cost function, the assignment process can be optimized while considering various factors such as payments, workloads, and preferences.

Main Results and Analysis

Instance 1, instance 3, instance 6, instance 8, and instance 10 were chosen as test cases for the Insurance Referee Assignment Problem to evaluate the effectiveness of the developed Clingo program.

```
(myenv) sin@CatherinedeMacBook-Air CSE579 % clingo ins1.txt referee.txt
clingo version 5.6.2
Reading from ins1.txt ...
Solving...
Answer: 1
assign(4,5)
Optimization: 2257
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 2257
Calls        : 1
Time         : 0.009s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.007s
```

Instance 1 test passed.

```
(myenv) sin@CatherinedeMacBook-Air CSE579 % clingo ins3.txt referee.txt
clingo version 5.6.2
Reading from ins3.txt ...
Solving...
Answer: 1
assign(6,11)
Optimization: 24740
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 24740
Calls        : 1
Time         : 0.010s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.008s
```

Instance 3 test passed.

```
(myenv) sin@CatherinedeMacBook-Air CSE579 % clingo ins6.txt referee.txt
clingo version 5.6.2
Reading from ins6.txt ...
Solving...
Answer: 1
assign(8,21) assign(9,21) assign(10,21)
Optimization: 90622
Answer: 2
assign(10,20) assign(8,19) assign(9,19)
Optimization: 89058
Answer: 3
assign(8,21) assign(9,19) assign(10,19)
Optimization: 88889
Answer: 4
assign(10,21) assign(8,19) assign(9,19)
Optimization: 88529
OPTIMUM FOUND

Models       : 4
  Optimum    : yes
Optimization : 88529
Calls        : 1
Time         : 23.719s (Solving: 5.40s 1st Model: 0.05s Unsat: 1.91s)
CPU Time     : 23.705s
```

Instance 6 test passed.

```
(myenv) sin@CatherinedeMacBook-Air CSE579 % clingo ins8.txt referee.txt
clingo version 5.6.2
Reading from ins8.txt ...
Solving...
Answer: 1
assign(14,28) assign(15,28)
Optimization: 230890
Answer: 2
assign(15,27) assign(14,27)
Optimization: 227988
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 227988
Calls        : 1
Time         : 0.035s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.033s
```

Instance 8 test passed.

```
(myenv) sin@CatherinedeMacBook-Air CSE579 % clingo ins10.txt referee.txt
clingo version 5.6.2
Reading from ins10.txt ...
Solving...
Answer: 1
assign(1,1) assign(2,1) assign(3,1)
Optimization: 24794
Answer: 2
assign(1,1) assign(2,1) assign(3,3)
Optimization: 22913
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 22913
Calls        : 1
Time         : 0.969s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.966s
```

Instance 10 test passed.

The results showed that the Clingo program was able to generate optimal assignments for each test instance. The program considered all available referees and cases, ensuring that each case was assigned to exactly one referee. The optimization value represents the minimum total workload for the referees, a key objective of the problem. The results demonstrated that the Clingo program found optimal solutions for each test instance with varying levels of complexity. The optimization values and solving times for each test instance are as follows:

Instance	Optimization value	Solving time
1	2257	0.009s
3	24740	0.010s
6	88529	23.719s
8	227988	0.035s
10	22913	0.969s

The optimization values ranged from 2257 to 230890, and the program was able to find the optimal solution within a reasonable amount of time, ranging from 0.009s to 23.719s. The test on instance 6 had a longer processing time compared to the other test instances, taking 23.719 seconds to find the optimal solution. This increased processing time can be attributed to the higher complexity of this particular test instance. As more cases and referees are involved, the number of possible assignments increases, leading to a larger search space for the program to explore. In the ins6.txt test instance, there were four different models generated before finding the optimal solution with an optimization value of 88529. The program had to explore more combinations and evaluate more models to find the best solution.

Overall, the results demonstrate the effectiveness of using ASP programming for solving complex optimization problems like the Insurance Referee Assignment Problem.

Conclusion

Using an ASP program for the Insurance Referee Assignment Problem has several advantages. It is adaptable to different problem sets, allowing for easy incorporation of additional constraints or modifications without a complete overhaul. It can also handle online, live assignment systems where availability can change dynamically with appropriate modifications.

Optimizing the model requires assigning reliable weights to the penalties in hard and weak constraints. Random assignment of weights may not produce the most efficient or optimal solution. Thus, thorough research and analysis are necessary to improve the model and achieve better results.

In conclusion, this project has demonstrated the capabilities of Clingo in solving optimization problems such as the Insurance Referee Assignment Problem. Although there were initial difficulties, leveraging resources such as lectures and online materials ultimately led to the successful completion of the project.

Opportunities for Future Work

There are potential avenues for future work to improve the use of Machine Learning models. For instance, it may be beneficial to integrate classification algorithms into the constraints to estimate the penalties' weights more accurately, rather than relying on arbitrary values that require manual adjustments. Additionally, to expedite the processing time of complex scenarios, running Clingo on distributed memory could be explored as a strategy to leverage available memory from different clusters. This approach may help optimize the runtime of the algorithm, especially in more challenging cases like scenario six.

Reference

ASP Challenge 2019. Retrieved from https://sites.google.com/view/aspcomp2019/home?authuser=0

Shane, R. (2019, October 2). Practical Applications of KRR. Retrieved from https://www.rshane.com/2019/10/02/practical-applications-of-krr/

Gebser, M., Ostrowski, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2010). A User's Guide to Gringo, Clasp, Clingo, and Iclingo (Version 3.x). [Technical report]. Retrieved from http://potassco.sourceforge.net/guide/