Insurance Referee Assignment Problem

Sahil Yogesh Hadke
Email: shadke1@asu.edu
ASU ID: 1229679960

Abstract

This project tackles the challenge of efficiently assigning referees to insurance claims. The system considers both internal (salaried) and external (paid per case) referees, incorporating their expertise, workload limits, and geographical preferences. It leverages Constraint Programming (CP) to ensure all essential rules (hard constraints) are met, such as not overloading a referee or assigning a high-value claim to an external reviewer. Additionally, the system strives for optimal solutions that consider preferences (soft constraints) like prioritizing internal referees, balancing workload and payments among external referees, and respecting referee preferences for case types and regions. This Clingo-based system was tested with successful results, demonstrating its ability to find optimal assignments that adhere to all constraints and preferences. The project also explores opportunities for future work, including incorporating more sophisticated fairness measures, handling uncertainty in data, and improving transparency in the decision-making process.

Problem Statement

In the insurance industry, the efficiency and accuracy of claims handling are paramount for customer satisfaction and operational effectiveness. The problem focuses on optimizing the allocation of referees—both internal and external—to various insurance claims based on a set of complex constraints. Referees are responsible for inspecting claims and determining their validity, and each has a specified maximum workload, geographical areas, and types of cases they are best suited to handle. The task is to assign these referees to insurance cases in such a way that no referee is overloaded, geographical and domain-specific preferences are honored, and financial expenditures are minimized.

The insurance company employs referees who specialize in different types of claims, such as passenger cars or trucks and operates across various geographical regions defined by postal codes. Each referee has a preference rating for both the type of claim and the region where they operate. Internal referees are salaried, while external referees are paid per case, with fees varying by the complexity and type of the case.

The challenge is to develop an algorithm that can handle the assignment of referees to insurance cases within a single day, adhering to hard constraints like workload limits and geographical and domain eligibility, and soft constraints aimed at reducing costs and balancing work distribution. The algorithm's effectiveness will be measured against a set scoring system, prioritizing cost efficiency, fairness, and adherence to preferences. This system should ensure that the insurance claims are processed efficiently, cost-effectively, and to the satisfaction of all parties involved.

Project Background

This project tackled a challenge faced by insurance companies: assigning referees efficiently to investigate customer claims. Referees visit locations and assess damages. The goal was to consider various factors and ensure an optimal assignment of referees to insurance cases.

Incorporating Preferences and Constraints: The system I built takes into account both hard constraints (strict rules) and soft constraints (preferences). Hard constraints ensure essential criteria are met. For example, a referee's daily workload cannot be exceeded, and referees can only handle cases in their designated regions and expertise (e.g., car vs. truck damage).

Soft constraints reflect preferences. The system prioritizes using internal referees, as they are salaried employees. Additionally, it strives for fairness among external referees, aiming to balance their workload and earnings across cases. Similarly, for both internal and external referees, the system prefers cases that align with their preferred regions and areas of expertise.

Utilizing Knowledge Representation and Reasoning (KRR): To achieve this intelligent assignment, I implemented concepts from Knowledge Representation and Reasoning (KRR). KRR equips computers with the ability to understand and reason about knowledge. One resource that proved valuable was the "Potassco Guide", which provides a comprehensive introduction to KRR concepts.

In the project, KRR helped me represent the knowledge about referees (workload limits, region/expertise preferences), cases (effort, damage amount, payment for external referees), and the relationships between them (e.g., a ref-

eree cannot handle a case outside their region). KRR then allowed me to reason about this knowledge and find optimal assignments that satisfy

Approach to Solving the Problem

My approach to the referee assignment problem leverages Constraint Programming (CP), specifically the Clingo system, to efficiently match referees with insurance claims. This approach excels at handling complex scenarios with various rules and preferences.

Modeling with Variables and Rules:

At the heart of my solution lie variables and rules. Variables represent the decisions that need to be made. For instance, the variable *assign(Cas_id, Reff_id)* indicates whether case "Cas_id" is assigned to referee "Reff_id."

Rules then capture the constraints and preferences governing these assignments. Hard constraints, which are essential and cannot be violated, are strictly enforced. Examples include a referee's daily workload limit and the expertise required for specific case types or regions. I present these as rules that would cause a conflict if broken (e.g., assigning a case to a referee who cannot handle it due to regional limitations).

On the other hand, soft constraints represent preferences that guide the assignment process towards a more desirable outcome. The system prioritizes assigning cases to internal referees, a cost-saving measure for the insurance company. Additionally, strive for fairness among external referees by aiming to balance their overall workload and the payments they receive from assigned cases. Finally, aim to respect referee preferences by prioritizing cases aligned with their preferred regions and expertise. These preferences are incorporated as optimization goals within the CP framework.

Clingo's Reasoning Engine: Finding Optimal Solutions

Clingo's powerful reasoning engine plays a crucial role. It doesn't simply find any feasible assignment; it searches for the optimal solution that satisfies all hard constraints while maximizing the fulfillment of soft constraints and minimizing the cost. Clingo achieves this by calculating metrics that reflect the degree to which soft constraints are met. For example, it calculates the difference in workload or payment received by different referees. Then, it searches for an assignment that minimizes these differences, leading to a fair and efficient allocation of cases.

Flexibility and Adaptability:

A significant advantage of this approach is its flexibility. I can easily modify the rules within the Clingo framework to adapt the system to changing requirements or specific company policies. Additionally, incorporating new types of constraints or preferences would simply involve adding new rules within the existing framework.

Furthermore, Clingo's powerful solver ensures that I find optimal solutions that consider all factors simultaneously. This is particularly important in complex scenarios with numerous interacting constraints and preferences.

Hard Constraints

I began by comprehending and simplifying the hard constraints required for this problem to be translated into ASP.

- **Maximum Workload per Referee:** This constraint ensures no referee is overloaded. The code calculates the total workload (Eff) assigned to a referee by summing the efforts of all cases assigned to them (similar to what I discussed earlier). A rule checks if this sum exceeds the referee's maximum workload capacity retrieved from the referee predicate. If so, the assignment is invalid.

```
:- Wrkld_sum = #sum{Eff,Reff_id ,Cas_id :
                    assign(Cas_id,Reff_id),
                case(Cas_id, Case_type, Eff, Dmg, Pstl_Cde, Pymnt)},
                referee(Reff_id,Reff_type, Wrkld_max,
                    Wrkld_prv, Pymnt_Prev),
                Wrkld_sum > Wrkld_max.
```

- **Case-Referee Type Compatibility:**

  This constraint ensures a case is only assigned to a referee with expertise in the case type. Similar to the regional constraint, the code utilizes a prefType predicate that defines a referee's preference level for a particular case type. Another rule checks if the assigned case type matches a type where the referee has a preference of 0 (no expertise). If so, the assignment is invalid. This ensures accurate claim assessments by assigning cases to referees with the relevant expertise.

```
:- assign(Cas_id,Reff_id),
referee(Reff_id,Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
case(Cas_id,Case_type, Eff, Dmg, Pstl_Cde, Pymnt),
prefType(Reff_id,Case_type,0).
```

- **Case-Referee Region Compatibility:** This constraint ensures a case is only assigned to a referee with expertise in the case's region. The code utilizes the prefRegion predicate, which defines a referee's preference level (0 for none, higher values for stronger preference) for a particular postal code (region). Another rule checks if the assigned case's postal code falls within a region where the referee has a preference of 0 (no expertise). If so, the assignment is invalid. This ensures efficient claim processing by assigning cases to referees familiar with the region.

```
:- assign(Cas_id, Reff_id),
referee(Reff_id, Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
case(Cas_id, Case_type, Eff, Dmg, Pstl_Cde, Pymnt),
prefRegion(Reff_id,Pstl_Cde,0).
```

- **High-Damage Cases for Internal Referees Only:** This constraint ensures only internal referees (identified by referee type "i") handle cases exceeding a certain damage threshold. The code retrieves the maximum damage allowed for internal referees from an externalMaxDamage predicate. A rule checks if the assigned case's damage is greater than this threshold and the referee assigned is external (type "e"). If so, the assignment is invalid. This likely reflects a company policy reserving high-value cases for internal referees with potentially more experience or oversight.

```
:- assign(Cas_id,Reff_id),
referee(Reff_id,Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
case(Cas_id,Case_type, Eff, Dmg, Pstl_Cde, Pymnt),
Reff_type==e ,
externalMaxDamage(Max_D), Dmg > Max_D.
```

Weak Constraints

- **Prioritizing Internal Referees:** This constraint aims to minimize costs by favoring internal referees (identified by referee type "i") over external ones (type "e"). The code defines an intnum function that counts the number of assignments to internal referees. A maximization rule prioritizes solutions that maximize this count, effectively pushing Clingo to assign cases to internal referees whenever possible.

```
intnum(N) :- N=#count{ 1 , Reff_id :
assign(Cas_id,Reff_id),
referee(Reff_id,Reff_type, Wrkld_max,
Wrkld_prv, Pymnt_Prev),
Reff_type==i }.
#maximize{N: intnum(N)}.
```

- **Balancing Payment Among External Referees:** This constraint strives for fairness by ensuring external referees receive comparable total payments across assigned cases. The code calculates the total payment for each external referee by summing the payments of their assigned cases. It then finds the minimum and maximum total payments among all external referees, representing the current payment imbalance.

  A minimization rule prioritizes assignments that reduce this gap. In simpler terms, whenever Clingo considers assigning a case to a referee, it prioritizes options that will bring the total payments of external referees closer together.

```
payment(Reff_id , Pay) :- Pay=
#sum{Pymnt, Reff_id, Cas_id:assign(Cas_id, Reff_id) ,
case(Cas_id, Case_type, Eff, Dmg, Pstl_Cde, Pymnt)},
referee(Reff_id, Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
Reff_type == e.

total_payment(Reff_id ,Pay_Money) :- Pay_Money=Pymnt_Prev+Pay,
referee(Reff_id, Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
payment(Reff_id , Pay), Reff_type == e.
min_pay(Pymnt_Mny1) :- total_payment(Reff_id1,Pymnt_Mny1),
0==#count{1,Reff_id2 : Pymnt_Mny1>Pymnt_Mny2,
total_payment(Reff_id2,Pymnt_Mny2) },
total_payment(Reff_id1,Pymnt_Mny1).

max_pay(Pymnt_Mny1) :- total_payment(Reff_id1,Pymnt_Mny1),
0==#count{1,Reff_id2 : Pymnt_Mny1<Pymnt_Mny2,
total_payment(Reff_id2,Pymnt_Mny2) },
total_payment(Reff_id1,Pymnt_Mny1).

pay_diff(D) :- D = (MAX-MIN), max_pay(MAX), min_pay(MIN).

#minimize{D,Cas_id,Reff_id:pay_diff(D), assign(Cas_id,Reff_id),
referee(Reff_id,Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
case(Cas_id,Case_type, Eff, Dmg, Pstl_Cde, Pymnt) }.
```

- **Balancing Workload Among All Referees (Internal and External):** Similar to the previous constraint, this one promotes fairness by aiming for a balanced distribution of workload across all referees. The code calculates the total workload for each referee by summing the efforts of their assigned cases. It then finds the minimum and maximum total workloads, representing the current workload imbalance.

  A minimization rule, similar to the one for payment, prioritizes assignments that reduce this gap. This ensures that no single referee is overloaded while others remain idle.

```
work(Reff_id,Case_Work) :-  Case_Work =
#sum{Eff , Reff_id , Cas_id :assign(Cas_id, Reff_id) ,
case(Cas_id, Case_type, Eff, Dmg, Pstl_Cde, Pymnt)},
referee(Reff_id, Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev).

total_workload(Reff_id, TWork) :- TWork = Wrkld_prv + Case_Work,
referee(Reff_id, Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
work(Reff_id,Case_Work).

min_work(T_wrk1) :- total_workload(Reff_id1, T_wrk1),
0==#count{1, Reff_id2 : T_wrk1>T_wrk2,total_workload(Reff_id2, T_wrk2) },
total_workload(Reff_id1, T_wrk1).

max_work(T_wrk1) :- total_workload(Reff_id1, T_wrk1),
0==#count{1, Reff_id2 : T_wrk1<T_wrk2,
total_workload(Reff_id2, T_wrk2) },total_workload(Reff_id1, T_wrk1).
work_diff(D) :- D = (MAX-MIN), max_work(MAX), min_work(MIN).

#minimize{D,Cas_id ,Reff_id :
work_diff(D), assign(Cas_id,Reff_id),
referee(Reff_id,Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
case(Cas_id,Case_type, Eff, Dmg, Pstl_Cde, Pymnt) }.
```

- **Respecting Referee Preferences for Case Types and Regions:**

  This constraint incorporates referee preferences for case types and regions. The code utilizes prefType and prefRegion predicates that define a referee's preference level (0 for none, higher values for stronger preference) for a particular case type and postal code (region), respectively. The objective function within the maximize directive prioritizes assignments that align with these preferences. In simpler terms, Clingo favors assigning cases to referees who would prefer to handle them based on their expertise and desired locations.

```
#maximize{Case_Pre+Region_Pre,Reff_id:
assign(Cas_id,Reff_id),
referee(Reff_id,Reff_type, Wrkld_max, Wrkld_prv, Pymnt_Prev),
case(Cas_id,Case_type, Eff, Dmg, Pstl_Cde, Pymnt) ,
prefRegion(Reff_id,Pstl_Cde,Region_Pre),
prefType(Reff_id,Case_type,Case_Pre)}.
```

Main results and analysis

Upon rigorous testing of my program with 10 instances representing diverse scenarios of insurance cases and referee preferences, I have achieved highly promising results that underscore the efficacy and robustness of the approach in solving the referee assignment problem. Through meticulous modeling of constraints and preferences using Answer Set Programming (ASP) and strategic optimization techniques, the program consistently delivered optimal solutions across all tested instances.

A key aspect of the success lies in the careful formulation of hard and weak constraints within the ASP code. By precisely defining constraints related to maximum referee workload, regional preferences, specialization requirements, and damage threshold limitations, I ensured that the generated solutions adhered strictly to the specified criteria. This meticulous attention to detail allowed me to enforce critical limitations while maintaining flexibility and scalability in the solution space.

Furthermore, the incorporation of weak constraints played a pivotal role in optimizing various criteria such as minimizing costs associated with external referees, balancing overall payments and workloads, and maximizing preferences for case types and regions. Through strategic optimization techniques embedded within the ASP code, I was able to achieve fair and efficient resource allocation, ultimately leading to significant cost savings for insurance companies.

An important aspect of achieving the desired outcome was the iterative refinement of the ASP code based on insights gained from testing each instance. By carefully analyzing the results and identifying areas for improvement, iteratively refined the code to enhance its performance and scalability. This iterative approach enabled us to fine-tune the optimization process, address any potential bottlenecks, and ultimately achieve optimal solutions across a diverse range of scenarios.

Overall, the main results of the testing demonstrate the effectiveness and versatility of the approach in solving the referee assignment problem in insurance cases. Through meticulous modeling, strategic optimization, and iterative refinement, I have developed a powerful tool that empowers insurance companies to efficiently manage resources, optimize case assignments, and drive cost savings.

```
sahilhadke@Sahils-Air project-code % clingo assign-referee.txt example10.txt
clingo version 5.6.2
Reading from assign-referee.txt ...
assign-referee.txt:9:31-41: info: global variable in tuple of aggregate element:
  Referee_id

assign-referee.txt:32:50-60: info: global variable in tuple of aggregate element:
  Referee_id

assign-referee.txt:44:59-69: info: global variable in tuple of aggregate element:
  Referee_id

Solving...
Answer: 1
assign(1,1) assign(2,1) assign(3,3)
Optimization: 8121
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 8121
Calls        : 1
Time         : 0.017s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.013s
sahilhadke@Sahils-Air project-code %
```

Output for Example 10

Conclusion

In conclusion, the results obtained from testing my program with 10 instances validate the effectiveness and robustness of my approach to solving the referee assignment problem in insurance cases. By meticulously modeling the problem constraints and preferences using Answer Set Programming (ASP) and optimizing for various criteria, my program provides insurance companies with a powerful tool for managing resources and optimizing case assignments.

The successful generation of optimal solutions across the tested instances highlights the versatility and scalability of my approach, making it suitable for real-world applications in the insurance industry. By leveraging the capabilities of ASP, my program offers insurance companies a systematic and efficient way to allocate referees to cases, ensuring fairness, efficiency, and cost-effectiveness in their operations.

Overall, my program represents a significant step forward in addressing the complex challenges faced by insurance companies in managing case assignments. With further refinement and deployment, my approach has the potential to streamline operations, improve customer satisfaction, and drive cost savings for insurance companies operating in today's dynamic and competitive market environment.

Opportunities for future work

While my current approach to solving the referee assignment problem in insurance cases has yielded promising results, several opportunities for future work can further enhance the effectiveness and applicability of my solution. These opportunities include:

- **Integration of Real-World Data:** Incorporating real-world data such as historical case data, referee performance metrics, and customer feedback can enhance the accuracy and relevance of my solution. By leveraging data-driven insights, insurance companies can make more informed decisions regarding case assignments and resource allocation.
- **Dynamic Optimization:** Developing algorithms and methodologies for dynamically optimizing referee assignments in real-time based on changing case priorities, resource availability, and organizational objectives. This dynamic approach can improve responsiveness and adaptability to evolving business requirements and market conditions.
- **Advanced Preference Modeling:** Enhancing the modeling of referee preferences by incorporating more sophisticated preference elicitation techniques, such as utility-based methods or machine learning algorithms. This can provide a more nuanced understanding of referee preferences and enable more effective optimization of case assignments.