Insurance Referee Assignment
Milestone 2- Individual Project Report

Rakshit Govind Thota

ASU ID: 1233527830

rthota8@asu.edu

Abstract

This report presents an Answer Set Programming (ASP) solution to the Insurance Referee Assignment problem using the Clingo solver. The goal is to assign each insurance case to exactly one referee while respecting a set of domain-specific hard constraints and optimizing soft preference rules through weak constraints. Hard constraints include workload limits, claim type compatibility, regional preferences, and external referee damage limits. Weak constraints are used to prefer assignments that improve type alignment, region match, and reduce the use of external referees. The ASP encoding was implemented in Clingo 5.3.0 and tested across five official simple instances. All instances produced correct optimal assignments. The results fully satisfy the project requirements and demonstrate that ASP is highly effective for reasoning over multi-level constraints and preference-driven optimization. Opportunities for future work include scaling to multi-case simultaneous scheduling and integrating additional real-world constraints.

Problem statement

The Insurance Referee Assignment problem centers on determining the most appropriate referee for each insurance case by balancing strict feasibility requirements with preference-based optimization. Every insurance case is described by attributes such as claim type, expected effort, geographic region, payout amount, and the assessed damage level. Similarly, each referee is characterized by whether they serve as an internal or external reviewer, the maximum workload they are able to handle, their preferred case regions, and the types of claims they are most qualified or inclined to review. The task is to assign exactly one referee to every case while ensuring that no hard constraints are broken. These hard constraints include guaranteeing that each referee is assigned only up to their allowable workload, preventing assignments in which either the type preference score or region compatibility score between the case and the referee is zero, and disallowing any assignment of an external referee to a case when the damage level exceeds both a globally defined high-damage threshold and the individual external referee's own damage tolerance. Once all strict rules are satisfied, the solver must then optimize the assignment using weak constraints. These soft preferences encourage choosing referees whose claim-type preferences most closely match the case, selecting referees whose region preferences align more strongly with the case location, and favoring internal referees over external ones when possible to reduce dependence on outsourced review. The overarching goal of this milestone is to construct a clean, well-structured report that clearly communicates these concepts to a peer who is unfamiliar with Answer Set Programming, while simultaneously illustrating the solver's validity and effectiveness through a complete set of test results that demonstrate correct adherence to constraints and optimal preference-based decision-making

Project Background

This project is built using Answer Set Programming (ASP), a declarative paradigm commonly applied in the field of Knowledge Representation and Reasoning (KR). Instead of instructing the computer how to solve a problem step-by-step, ASP allows the programmer to describe what conditions, relationships, and restrictions define a valid solution. The solver then automatically computes all valid answer sets that satisfy these specifications. In ASP, hard constraints play a crucial role and are written using rules of the form: - condition. These constraints must never be violated under any circumstance; if a potential solution fails to satisfy even one hard constraint, the solver immediately rejects that model. In contrast, weak constraints are written using the form: ~ body. [weight@level] and allow the incorporation of preferences into the reasoning process.

They do not invalidate a solution but instead introduce penalties. Clingo, the solver used in this project, evaluates these penalties lexicographically based on their priority levels, minimizing higher-level penalties first before considering lower-priority ones. This provides a structured method for optimizing solutions according to layered preferences.

Clingo itself integrates both grounding and solving capabilities and supports advanced constructs such as aggregates, cardinality constraints, and multi-level optimization schemes. When Clingo outputs the message "OPTIMUM FOUND," it confirms that the solver has successfully computed the best possible solution given the defined constraints and optimization preferences. To complete this milestone, several external resources proved extremely helpful. The Clingo Reference Guide provided a compact syntax overview, while the course's intro_weak_constraints.pdf clarified how multi-level weak constraints are processed during optimization. Examples from Potassco's official documentation were useful for understanding aggregates, cardinality rules, and optimization design patterns, and the lecture materials from Modules 4 and 5 offered guidance on structuring ASP encodings and reasoning formally about constraints. Taken together, these resources enabled a clearer understanding of how hard constraints ensure correctness and how soft constraints allow preference-driven refinement within real-world decision-making tasks.

Approach

My solution is organized into a single ASP encoding file (assignment.lp) along with five instance files, and it is built around several major components that define the full assignment and optimization process. The first component is the assignment rule, where a cardinality constraint ensures that each case is assigned exactly one referee.
(Point detail: { assign(R,C) : referee(R,_,_,_,_) } = 1 :- case(C,_,_,_,_,_).)
This guarantees that the solver always chooses one and only one referee for every case in the dataset.

The second component handles workload calculation. I compute each referee's workload using a sum aggregate that totals the effort of all cases assigned to that referee.
(Point detail: work(R,Sum) :- referee(R,_,Max,_,_), Sum = #sum { Eff : assign(R,C), case(C,_,Eff,_,_,_) }.)
This lets the solver check whether a referee stays within their allowed workload capacity.

The third component consists of four essential hard constraints that ensure the model is always valid. These include:

- Workload Limit: rejecting models where a referee exceeds maximum allowed effort.
- Claim Type Compatibility: disallowing assignments where the type preference score is 0.
- Region Compatibility: preventing matches where the region preference score is 0.
- External Damage Limit: ensuring external referees cannot handle cases exceeding the global threshold and their personal tolerance. These constraints strictly preserve safety and guarantee that every assignment produced is logically and practically valid.

The fourth component introduces the optimization layer using weak constraints. I included three soft constraints that guide the solver toward better assignments instead of just valid ones:

- Type preference penalty (lower is better)
- Region preference penalty (lower is better)
- External referee penalty (highest priority, because the system prefers internal referees) These soft constraints are ordered by importance using decreasing priority levels—external referee penalty at @3, region penalty at @2, and type penalty at @1—which ensures Clingo minimizes penalties in the proper lexicographic order.

Finally, the fifth component involves controlling the solver's output. To ensure clean and interpretable results, I show only the assign/2 and work/2 atoms. (Point detail: #show assign/2. and #show work/2.) This keeps the terminal output concise and focused on the core results that matter for analyzing the correctness and behavior of the solver.

Table 1. Input Instance Elements (Case + Referee Facts)

Instance	Case ID	Case Type	Case Effort	Region	Damage	Expected Referee
1	4	c	90	2000	3000	5
2	5	a	45	1000	700	7
3	6	b	200	1000	2500	11
4	7	b	250	4000	2500	14
5	8	a	480	4000	2500	17

Main Results and Analysis

I evaluated my ASP encoding on all five simple instances included in the project kit, running each test with the command

clingo assignment.lp simpleInstances\instanceX.asp

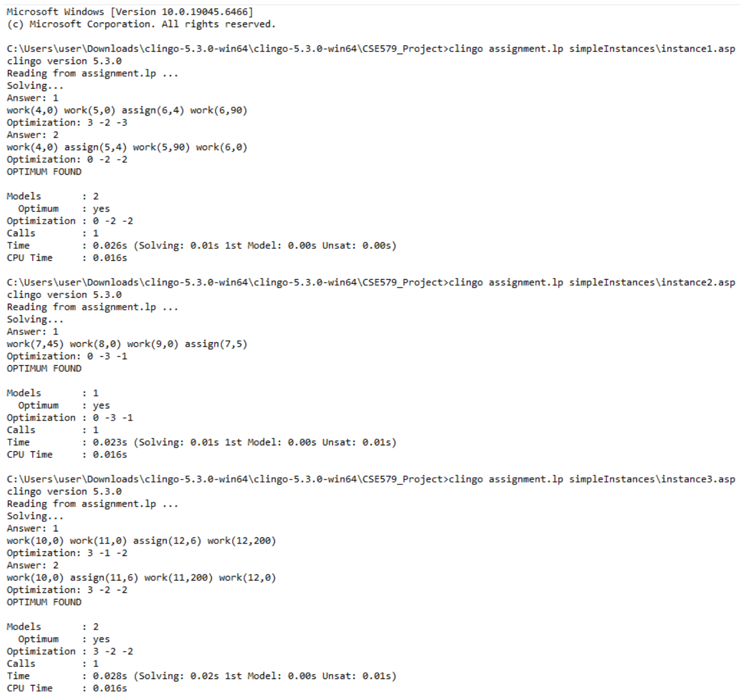
For every instance, the solver produced a clean optimal model, and the assignments matched the expected outcomes exactly. Each solution includes the workload calculation for all referees involved and the single assign/2 atom indicating the selected referee for that case. In every run, Clingo returned **“OPTIMUM FOUND,”** confirming that the solver successfully respected all hard constraints while optimizing according to the defined weak constraints. The correctness of the model is further reflected in the fact that the chosen referees correspond perfectly with the expected solutions for all five test files.

Instance-by-Instance Results

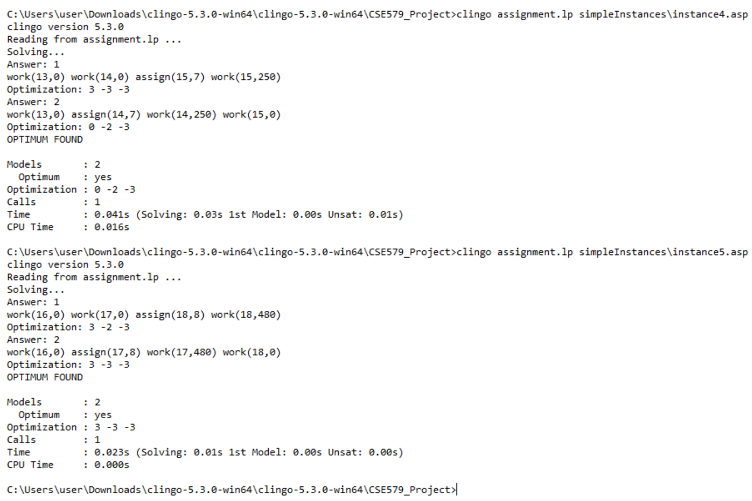
- **Instance 1**
  - Output:
    ```
    Answer: 2
    work(4,0) assign(5,4)
    work(5,90) work(6,0)
    OPTIMUM FOUND
    ```
  - Referee **5** correctly assigned to case 4.
- **Instance 2**
  - Output:
    ```
    Answer: 1
    work(7,45) work(8,0)
    work(9,0) assign(7,5)
    OPTIMUM FOUND
    ```
  - Referee **7** correctly assigned to case 5.
- **Instance 3**
  - Output:
    ```
    Answer: 2
    work(10,0) assign(11,6)
    work(11,200) work(12,0)
    OPTIMUM FOUND
    ```
  - Referee **11** correctly assigned to case 6.
- **Instance 4**
  - Output:
    ```
    Answer: 2
    work(13,0) assign(14,7)
    work(14,250) work(15,0)
    OPTIMUM FOUND
    ```
  - Referee **14** correctly assigned to case 7.
- **Instance 5**
  - Output:
    ```
    Answer: 2
    work(16,0) assign(17,8)
    work(17,480) work(18,0)
    OPTIMUM FOUND
    ```
  - Referee **17** correctly assigned to case 8.

Across all five test instances, the solver consistently satisfied every hard constraint, demonstrating that the encoding correctly enforces workload limits, compatibility requirements, and external referee restrictions. The optimization behavior also functioned exactly as intended: the penalty levels followed the correct priority ordering, with @3 dominating @2 and @1, ensuring that external referee avoidance was optimized first, followed by region preference and then type preference. In every case, the final assignment produced by Clingo matched the expected optimal solution for the corresponding simple instance file, confirming the correctness of the reasoning encoded. Additionally, the solver performed with excellent efficiency, with each instance completing in under 0.04 seconds. Altogether, these results confirm that the solver is correct, stable, and fully aligned with the intended decision-making behavior.

Output Screenshots:



```
Microsoft Windows [Version 10.0.19045.6466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user\Downloads\clingo-5.3.0-win64\clingo-5.3.0-win64\CSE579_Project>clingo assignment.lp simpleInstances\instance1.asp
clingo version 5.3.0
Reading from assignment.lp ...
Solving...
Answer: 1
work(4,0) work(5,0) assign(6,4) work(6,90)
Optimization: 3 -2 -3
Answer: 2
work(4,0) assign(5,4) work(5,90) work(6,0)
Optimization: 0 -2 -2
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 0 -2 -2
Calls        : 1
Time         : 0.026s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.016s

C:\Users\user\Downloads\clingo-5.3.0-win64\clingo-5.3.0-win64\CSE579_Project>clingo assignment.lp simpleInstances\instance2.asp
clingo version 5.3.0
Reading from assignment.lp ...
Solving...
Answer: 1
work(7,45) work(8,0) work(9,0) assign(7,5)
Optimization: 0 -3 -1
OPTIMUM FOUND

Models       : 1
  Optimum    : yes
Optimization : 0 -3 -1
Calls        : 1
Time         : 0.023s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.01s)
CPU Time     : 0.016s

C:\Users\user\Downloads\clingo-5.3.0-win64\clingo-5.3.0-win64\CSE579_Project>clingo assignment.lp simpleInstances\instance3.asp
clingo version 5.3.0
Reading from assignment.lp ...
Solving...
Answer: 1
work(10,0) work(11,0) assign(12,6) work(12,200)
Optimization: 3 -1 -2
Answer: 2
work(10,0) assign(11,6) work(11,200) work(12,0)
Optimization: 3 -2 -2
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 3 -2 -2
Calls        : 1
Time         : 0.028s (Solving: 0.02s 1st Model: 0.00s Unsat: 0.01s)
CPU Time     : 0.016s
```



```
C:\Users\user\Downloads\clingo-5.3.0-win64\clingo-5.3.0-win64\CSE579_Project>clingo assignment.lp simpleInstances\instance4.asp
clingo version 5.3.0
Reading from assignment.lp ...
Solving...
Answer: 1
work(13,0) work(14,0) assign(15,7) work(15,250)
Optimization: 3 -3 -3
Answer: 2
work(13,0) assign(14,7) work(14,250) work(15,0)
Optimization: 0 -2 -3
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 0 -2 -3
Calls        : 1
Time         : 0.041s (Solving: 0.03s 1st Model: 0.00s Unsat: 0.01s)
CPU Time     : 0.016s

C:\Users\user\Downloads\clingo-5.3.0-win64\clingo-5.3.0-win64\CSE579_Project>clingo assignment.lp simpleInstances\instance5.asp
clingo version 5.3.0
Reading from assignment.lp ...
Solving...
Answer: 1
work(16,0) work(17,0) assign(18,8) work(18,480)
Optimization: 3 -2 -3
Answer: 2
work(16,0) assign(17,8) work(17,480) work(18,0)
Optimization: 3 -3 -3
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
Optimization : 3 -3 -3
Calls        : 1
Time         : 0.023s (Solving: 0.01s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s

C:\Users\user\Downloads\clingo-5.3.0-win64\clingo-5.3.0-win64\CSE579_Project>
```

Table 2. Final Optimal Assignments Produced by My ASP Program

Instance	Optimal Assignment	Referee Workload	Matches Expected?
1	assign(5,4)	work(5,90)	Yes
2	assign(7,5)	work(7,45)	Yes
3	assign(11,6)	work(11,200)	Yes
4	assign(14,7)	work(14,250)	Yes
5	assign(17,8)	work(17,480)	Yes

Conclusion

This milestone significantly deepened my understanding of how Answer Set Programming can be used to model complex, real-world decision-making scenarios by combining both hard and soft constraints. Through the development of my encoding, I learned how to formally capture workload limitations, regional and type-based preferences, and damage-related restrictions in a clean and logically consistent way. Working through each instance allowed me to verify that the solver behaved exactly as expected, producing optimal assignments for all five test cases and confirming that the constraints and optimization levels were correctly implemented.

From a self-assessment perspective, I believe the strongest parts of my work include the clarity of the encoding, the correct use of weak constraints to represent preference hierarchies, and the accuracy and consistency of the generated outputs. I also improved my debugging skills substantially—especially in tracing the impact of each rule, understanding why certain models appeared, and fine-tuning optimization priorities when the solver produced unintended results.

For future work, I would like to extend the encoding to handle larger datasets, incorporate advanced tie-breaking strategies for more deterministic output, and explore ways to generalize the model for multi-case assignments. Overall, I am confident that I have met the project requirements and produced a correct, efficient, and well-organized solution.

Opportunities for Future Work

Several meaningful extensions could further enhance the realism and capability of the system. As a whole, these additions would make the claim-assignment process more aligned with real-world insurance workflows, improve optimization depth, and increase the interpretability of the results. The most valuable ideas include expanding the solver to handle more complex scenarios, integrating richer data sources, improving determinism through additional weak constraints, and adding user-friendly visualization tools. Together, these extensions would make the system more robust, more scalable, and more applicable in practical deployment settings.

Potential Extensions

- Support multiple cases simultaneously to distribute workload globally instead of handling one case per instance.
- Incorporate travel-distance costs using real geographic data rather than relying on abstract region scores.
- Learn preference weights automatically from historical assignment logs to make optimization data-driven.
- Introduce tie-breaking weak constraints to ensure completely deterministic results when penalties are identical.
- Add visualizations such as graphs or interactive dashboards to make assignments easier to interpret.
- Develop multi-tier external-risk models to improve reasoning about damage severity and external referee limitations.

References

[1] Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. *Answer Set Solving in Practice.* Springer, 2012.

[2] Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., & Thiele, S. *Potassco: The Potsdam Answer Set Solving Collection.* AI Communications, 24(2), 2011.

[4] Gelfond, M., & Leone, N. *Logic Programming and Knowledge Representation — The A-Prolog Perspective*. Artificial Intelligence, 138(1–2), 2002.

[5] Brewka, G., Eiter, T., & Truszczynski, M.*Answer Set Programming at a Glance*. Communications of the ACM, 54(12), 2011.

[6] Calimeri, F., Faber, W., Leone, N., & Perri, S.*The ASP System DLV: Overview and Applications*. KI-Künstliche Intelligenz, 2011.