ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230542

A Logic-Based Framework for Explainable Agent Scheduling Problems

Stylianos Loukas Vasileiou[a;*], **Borong Xu**[a] **and William Yeoh**[a]

[a]Washington University in St. Louis

Abstract. *Agent Scheduling Problems* (ASPs) are common in various real-world situations, requiring explainable decision-making processes to effectively allocate resources to multiple agents while fostering understanding and trust. To address this need, this paper presents a logic-based framework for providing explainable decisions in ASPs. Specifically, the framework addresses two types of queries: *reason-seeking queries*, which explain the reasoning behind scheduling decisions, and *modification-seeking queries*, which offer guidance on making infeasible decisions feasible. Acknowledging the importance of privacy in multi-agent scheduling, we introduce a privacy-loss function that measures the disclosure of private information in explanations, enabling a privacy-preserving aspect in our framework. By using this function, we introduce the notion of *privacy-aware explanations* and present an algorithm for computing them. Empirical evaluations demonstrate the effectiveness and versatility of our approach.

1 Introduction

Agent scheduling problems (ASPs) involve allocating a finite set of resources to multiple agents over a specific time frame. These problems are pervasive in real-world scheduling systems, ranging from personnel shift assignments [34] to machine job allocation [38], and even scheduling awake and asleep periods for Mars rovers [8]. Apart from generating a schedule that allocates resources to agents, it is crucial to ensure that both the schedule and the underlying decision-making process are explainable. An agent may require an explanation for why certain scheduling decisions were not satisfied or why a schedule could not be generated at all. In such cases, understanding the reasons behind these issues is not only enlightening but also necessary for rectifying the problem. Additionally, privacy plays a significant role due to the sensitive nature of personal information that may be included in ASPs, such as agents' constraints and preferences. Preserving privacy helps protect individual agents from potential discrimination or unauthorized access to their information, fostering trust and willingness to participate in the scheduling process. Therefore, incorporating explanation generation modalities with privacy-preserving considerations into ASP systems is highly desirable.

To address this need, this paper presents a logic-based framework aimed at making ASPs explainable. The framework accommodates two types of queries: *reason-seeking queries*, which clarify why a scheduling decision was (or not) derived, and *modification-seeking queries*, which offer guidance on rendering infeasible scheduling decisions feasible. Recognizing the importance of privacy in multi-agent scheduling, we use the concept of agent access rights to distinguish between *public* and *private information*, and introduce a straightforward privacy-loss function to quantify the amount of private information disclosed in explanations. Using this function, we then define the notion of *privacy-aware explanations* and present the *Query Understanding and Efficient Response with Intelligible Explanations of Schedules* (QUERIES) algorithm for computing them. This approach ensures that the explanations provided maintain the confidentiality of sensitive information while still offering valuable insights into the scheduling decisions.

In summary, our framework advances existing explainable scheduling methods, which typically focus on specific scheduling problems [1, 3, 28], by providing a general solution applicable to a broader range of ASPs. Our main contributions are as follows:

- We introduce a general logic-based explanation generation framework for ASPs that addresses both reason-seeking queries and modification-seeking queries.
- We propose a privacy-loss function to quantify the amount of private information included in an explanation and define the concept of privacy-aware explanations.
- We present the QUERIES algorithm for computing explanations. Empirical evaluations demonstrate the effectiveness and versatility of our approach.

2 Motivating Thought Experiment

To better understand the challenges faced by agent scheduling problems and the importance of generating effective explanations, let us engage in a thought experiment inspired by a simplified version of the *employee shift assignment* problem [34]. Consider a scenario based on the *employee shift assignment* problem [34]. In this scenario, an automated scheduling agent named Alice is responsible for assigning shifts to employees at a company. Specifically, there are three shift types – *morning*, *afternoon*, and *evening* – and four employees – *Thanos*, *Irene*, *Vicky*, and *Rose* – who need to be assigned shifts over three days from *Monday* to *Wednesday*.

The scheduling problem consists of the following *domain constraints*:

C_1: *All employees must be assigned a total of two shifts.*
C_2: *Employees cannot be assigned multiple shifts per day.*
C_3: *No two employees can be assigned the same shift the same day.*
C_4: *Employees cannot be assigned a morning shift right after an evening shift.*

* Corresponding Author. Email: v.stylianos@wustl.edu.

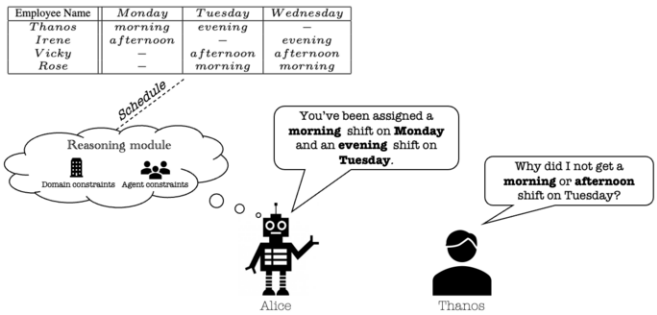


Figure 1: Instance of the thought experiment with Alice and Thanos.

Moreover, each employee has *personal constraints*:

C_T: *Thanos wants only morning or afternoon shifts.*

C_I: *Irene does not want evening shifts.*

C_V: *Vicky wants the afternoon shift on Tue. and Wed.*

C_R: *Rose wants the morning shift on Tue. and Wed.*

Here, Alice's objective is to find a schedule that satisfies all domain constraints and, as much as possible, accommodates the employee constraints according to their weights, which in this example are based on the employees' seniority levels.

Let us assume that Alice finds a feasible schedule, but it does not meet Thanos' constraint of being assigned morning or afternoon shifts. Thanos, in turn, may inquire about the reason for this assignment. To generate an effective explanation, Alice needs a framework that can generate explanations that are informative and tailored to the specific needs of the explainee, that is, Alice must first recognize the nature of the explainee's query.

In our thought experiment, Thanos' query is a *reason-seeking query*, as he wants to know "why" his constraint was unsatisfied in the schedule. In response, Alice should provide a (reason-seeking) explanation that identifies the reasons behind her (scheduling) decision. For example, Alice might explain that due to the constraints of the problem and the higher priority given to the preferences of Rose and Vicky, it was not possible to assign Thanos morning shifts on Tuesday or Wednesday without affecting the overall quality of the allocation.

However, providing a reason-seeking explanation alone may not be sufficient in all scenarios. Suppose Alice could not create a feasible schedule at all due to conflicting constraints. In this case, a higher-level employee, such as a manager, may want to understand "how" to adjust the scheduling problem to derive a feasible schedule. This type of query is a *modification-seeking query*, which requires an explanation that helps the manager identify issues preventing a feasible schedule and suggest potential modifications.

In addition to addressing these two types of queries, Alice's explanations should respect the *privacy* of the other employees. To achieve this, Alice could only reveal information according to the employees' *access rights*. In doing so, Alice distinguishes between *public information* (information that can be revealed to employees with access rights) and *private information* (information that cannot be revealed to employees without access rights).

This thought experiment demonstrates some of the challenges of generating explanations in the context of agent scheduling problems. Indeed, in Section 4 we present an explanation generation framework that can handle the complexity of the problem, account for the explainee's needs and access rights, and produce informative explanations.

3 Background

We now provide some background on the satisfiability (SAT) problem, a general agent scheduling problem (ASP) definition, and our logic-based representation of that problem.

3.1 Satisfiability

We assume familiarity with propositional logic. A knowledge base KB is a set of constraints, where each constraint is built up recursively from *literals* (i.e., variables or its negations) using the usual logical connectives.

Satisfiability (SAT) [9] is the prototypical NP-complete problem of finding an assignment of truth values to variables in order to make a knowledge base KB true. If there exists a truth value assignment μ that makes KB true, then we say that μ is a *model* of KB and KB is *satisfiable*, otherwise KB is *unsatisfiable*, denoted by $KB \models \bot$. A KB *entails* a constraint φ, denoted $KB \models \varphi$, iff $KB \cup \{\neg\varphi\} \models \bot$.

Partial weighted MaxSAT [24] is an extension of SAT in which constraints are partitioned into *hard* and *soft* constraints, where each soft constraints is given a weight. Hard constraints must always be satisfied in a solution, whereas soft constraints may not. The goal of MaxSAT is to find an assignment that satisfies the hard clauses and maximizes the sum of weights of the satisfied soft clauses.

3.2 Agent Scheduling Problem

In general, the goal of an *agent scheduling problem* (ASP) is to distribute a set of resources to a set of agents over a scheduling horizon. Formally, it can be defined as a tuple $\mathcal{A} = \langle A, R, S, C \rangle$, where $A = \{a_i\}_{i=1}^{n}$ is a set of agents, $R = \{r_j\}_{j=1}^{m}$ is a set of resources, $S = \{s_t\}_{t=1}^{h}$ is a set of time steps, and C is a set of constraints that consists of *domain constraints*, which are intrinsic and describe the problem's dynamics, as well as *agent constraints*, which are extrinsic and describe the agents' personal constraints.

A solution to an ASP $\mathcal{A}$ is a *schedule* Σ, that is an $|A| \times |R| \times |S|$ matrix, where each cell $\Sigma[i, j, t] = 1$ if agent a_i is assigned resource r_j at time step s_t and $\Sigma[i, j, t] = 0$ otherwise. A schedule is *feasible* if all the domain constraints, which are treated as hard constraints, are satisfied. A schedule is *optimal* if it is feasible and all the agent constraints, which are treated as soft constraints, are maximized.

3.3 Logic-based Agent Scheduling Problems

In this paper, we model an ASP $\mathcal{A}$ as a logic-based problem, that is, we encode $\mathcal{A}$ into a set of logical constraints for which satisfiability can be decided. By using an appropriate logical language, the problem's dynamics are encoded into a knowledge base KB that expresses all the scheduling constraints that a desired schedule should satisfy. Specifically, the knowledge base KB consists of domain constraints C_D and agent constraints C_A, where C_D are treated as hard constraints and C_A as weighted soft constraints. As such, the scheduling problem turns into a MaxSAT problem, where the quality of a feasible schedule depends on the degree to which the soft clauses are satisfied. The objective function of a candidate schedule is then defined as the sum of weights of satisfied soft constraints, and an optimal schedule is the solution with the highest possible objective value. A plethora of scheduling problems has been modeled using logic-based approaches [2, 5, 10, 14, 18, 21, 23, 27].

For ease of presentation, in this paper we will use propositional logic to encode ASPs. We formally define a *logic-based ASP* (L-ASP) as follows:

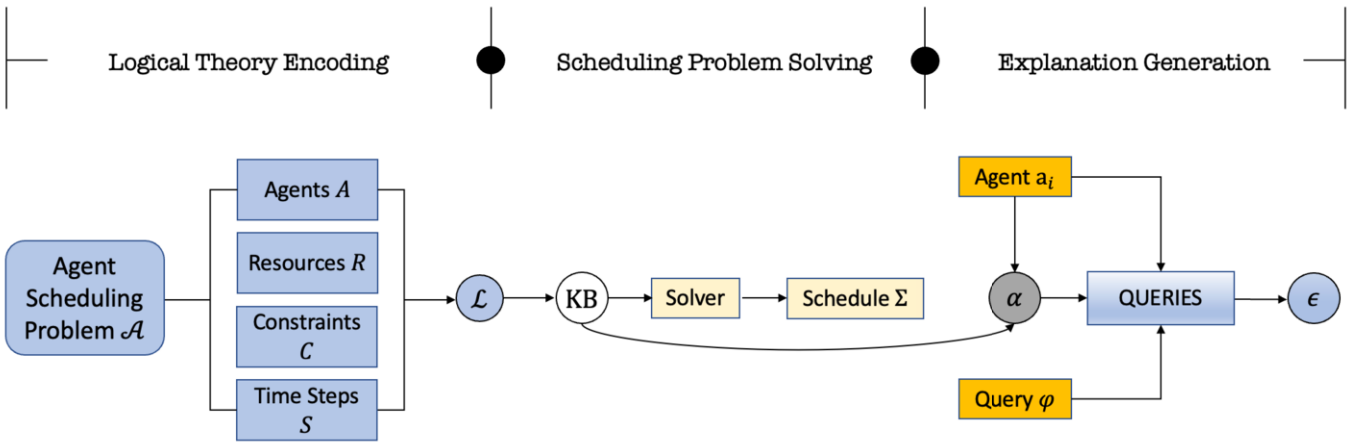


Figure 2: Overview of Our Explainable Logic-based Agent Scheduling Problem Pipeline.

Definition 1 (L-ASP). *An* L-ASP *is a tuple* $\mathcal{L} = \langle A, R, S, KB \rangle$, *where* $KB = C_D \cup C_A$ *and:*

- C_D *is the set of domain-specific (hard) constraints. These constraints are intrinsic to the problem and must be satisfied by a solution.*
- $C_A = \bigcup_{i=1}^{n} C_i$ *is the set of agent (weighted soft) constraints. Each* $C_i = \{(w_k, c_k^i)\}_{k=1}^{l}$, *where each* c_k^i *is a constraint associated with agent* a_i *and* w_k *is its corresponding weight.*

A schedule can be derived by using off-the-shelf SAT solvers [4] to search for a model μ of KB that satisfies all of the constraints in C_D and possibly some of the constraints in C_A. If a model μ exists, then a *feasible* schedule Σ_μ is derived by extracting from μ the truth values of the variables corresponding to agents, resources, and time steps. Otherwise, the scheduling problem is *infeasible*, i.e., no feasible schedule exists. Finally, a schedule Σ_μ is deemed *optimal* if a model μ exists and maximizes the cumulative sum of weights of satisfied soft constraints in C_A.

Note that the knowledge base $KB = C_D \cup C_A$ may be unsatisfiable due to inconsistencies in the domain constraints and/or agent constraints. However, if a schedule Σ_μ exists, then that means that Σ_μ logically follows from a satisfiable subset $KB_\mu \subseteq KB$. In the next section, we use KB to denote the knowledge base from which explanations are derived. Depending on the context, KB could refer to either a satisfiable subset of the original knowledge base (i.e., KB_μ) or the overall unsatisfiable knowledge base.

4 Explainable Agent Scheduling Problems

We now present our explanation generation framework for agent scheduling problems. We particularly address the following problem:

> Given a logic-based L-ASP $\mathcal{L} = \langle A, R, S, KB \rangle$ and a *query* φ with respect to KB, the goal is to find an *explanation* for φ that can be inferred from KB.

As discussed in Section 2, we are interested in a framework that can generate explanations for agent scheduling problems that are not only informative but also tailored to the specific needs of the explainee. Such a framework should in principle:

- Address two general types of queries: *reason-seeking queries*, which aim to uncover *why* certain scheduling decisions were (or not) made, and *modification-seeking queries*, which focus on identifying potential modifications to the problem.
- Generate informative and concise explanations for the two query types.
- Preserve the privacy of other agents by only revealing information with respect to *access-rights*.

A general pipeline is shown in Figure 2. We now describe how to generate explanations for the two query types.

4.1 Explaining Reason-Seeking Queries

A reason-seeking query, denoted by φ_r, aims to uncover *why* certain scheduling decisions were made. Recall from Section 2 that Thanos wants to know why Alice did not assign him only morning shifts. Alternatively, a higher-level employee (e.g., a manager) may want to understand why a feasible schedule cannot be generated.

To explain reason-seeking queries, we assume that $KB \models \varphi_r$. There are two possible scenarios to consider:

- **Agent Constraints in a Schedule:** If the query φ_r captures an unsatisfied (or satisfied) agent constraint in a schedule Σ_μ, then $\varphi_r \in \neg C_A$ (or $\varphi_r \in C_A$).[1] In this scenario, an explanation should identify the reasons why the constraint holds true with respect to the schedule. Note that the knowledge base KB here is satisfiable (see Section 3.3).
- **Infeasible Scheduling Problems:** If the query φ_r is aimed at capturing why a problem is infeasible, i.e., why a feasible schedule cannot be generated, then generally $\varphi_r = \bot$. In this case, the explanation should identify the inconsistencies within the scheduling constraints that lead to infeasible schedules. Note that the knowledge base KB here is unsatisfiable, i.e., there is no model of KB from which a feasible schedule can be extracted.

Formally now, an explanation for a reason-seeking query is defined as follows:

Definition 2 (Reason-seeking Explanation). *Given a knowledge base* KB *that encodes an L-ASP* $\mathcal{L}$ *and a reason-seeking query* φ_r, *we consider an explanation* $\epsilon_r \subseteq KB$ *to be a* reason-seeking explanation *for* φ_r *if:*

- ϵ_r *is sufficient:* $\epsilon_r \models \varphi_r$, *meaning that the explanation* ϵ_r *entails the query* φ_r.
- ϵ_r *is minimal: For all proper subsets* $\epsilon_r' \subset \epsilon_r$, $\epsilon_r' \not\models \varphi_r$, *indicating that no smaller subset of* ϵ_r *are sufficient.*

[1] Note that $\neg C_A$ denotes the logical negation of all the constraints in C_A.

These conditions ensure that the reason-seeking explanation is both sufficient and minimal in addressing the query.

4.2 Explaining Modification-Seeking Queries

Modification-seeking queries, denoted by φ_m, focus on identifying potential modifications to a scheduling problem to address specific issues. For example, Thanos may want to know how to incorporate his unsatisfied constraint in Alice's schedule, or a manager may seek ways to adjust the scheduling problem to generate a feasible schedule.

To explain modification-seeking queries, we assume that $KB \not\models \varphi_m$. Specifically, to explain these query types, we seek to identify a set of constraints from the knowledge base KB that, when retracted, $KB \models \varphi_m$. Like before, there are two possible scenarios to consider:

- **Unsatisfied Agent Constraints in a Schedule:** If the query φ_m concerns accommodating an unsatisfied agent constraint in a schedule Σ_μ, then $\varphi_m \in C_A$.
- **Infeasible Scheduling Problems:** If the query φ_m is aimed at explaining how a problem can be modified such that a feasible schedule can be found, then $\varphi_m = \top$.

We now define an explanation for a modification-seeking query as follows:

Definition 3 (Modification-seeking Explanation)**.** *Given a knowledge base KB that encodes an L-ASP $\mathcal{L}$ and a modification-seeking query φ_m, we consider an explanation $\epsilon_m \subseteq KB$ to be a* modification-seeking explanation *for φ_m if:*

- *ϵ_m enables the entailment of φ_m: $KB \setminus \epsilon_m \models \varphi_m$, meaning that the query φ_m is entailed when the constraints in ϵ_m are removed from the knowledge base.*
- *ϵ_m is minimal: For all proper subsets $\epsilon'_m \subset \epsilon_m$, $KB \setminus \epsilon'_m \not\models \varphi_m$, indicating that no smaller subset of ϵ_m can satisfy the query when removed from the knowledge base.*

These conditions ensure that the modification-seeking explanation is both effective and minimal in addressing the query.

4.3 Privacy-Aware Explanations

It is reasonable to assume that individuals might prefer explanations for scheduling decisions that only encompass public information, as they could perceive these as more satisfying and equitable compared to explanations that incorporate private information as well. To explore this possibility and incorporate potential privacy preferences into our framework, we propose that agents have *access rights* on the different pieces of information about the scheduling problem. Specifically, we assume an *access-rights* function:

$$\alpha : A \times KB \to \{0, 1\} \tag{1}$$

that determines whether an agent $a_i \in A$ has *access rights* to a constraint $c \in KB$, returning 1 if a_i has access to c and 0 otherwise.

While we have motivated access rights through the lens of privacy, note that the function can also encode access rights through other means as well (e.g., security clearances and other administrative compartmentalization protocols).

Given an agent a_i and the function α, we define the *privacy loss* ρ_i of an explanation ϵ with regard to the agent as the count of constraints inaccessible to it:

$$\rho_i(\epsilon) = |\epsilon| - \sum_{c \in \epsilon} \alpha(a_i, c) \tag{2}$$

Lastly, we define an explanation ϵ_i as being *privacy-aware* in relation to agent a_i and query φ if it incurs the least privacy loss among all possible explanations E for the query φ:

$$\epsilon_i = \operatorname*{argmin}_{\epsilon \in E} \rho_i(\epsilon) \tag{3}$$

4.4 Illustrating Example

Consider the employee shift assignment problem presented in Section 2. To represent the problem using (propositional) logic, we employ Boolean decision variables $x_{i,j,t}$ for all $a_i \in A$, $r_j \in R$, and $s_t \in S$, where each variable is set to true if and only if agent a_i is assigned shift r_j on day s_t. Otherwise, it is set to false. These variables comprise the domain constraints C_D and agent constraints C_A which make up the knowledge base KB. Note that we assume the following weights for employee constraints C_A: $w(C_R) = w(C_V) > w(C_T) > w(C_I)$.[2]

Recall from Section 2 that Alice has generated a schedule (see Figure 1) that does not satisfy Thanos' constraint, prompting him to ask Alice a reason-seeking query. In our logic-based framework, this translates to the query $\varphi_r = \{\neg x_{1,1,2} \vee \neg x_{1,2,2}\}$. There are two reason-seeking explanations for this query:

- $\epsilon_{r1} = \{x_{4,1,2}, \neg x_{4,1,2} \vee \neg x_{1,1,2}\}$, stating that only one employee can be assigned a morning shift on the same day (domain constraint) and that Rose's preference was given a higher priority that day.
- $\epsilon_{r2} = \{x_{3,2,2}, \neg x_{3,2,2} \vee \neg x_{1,2,2}\}$, stating that only one employee can be assigned an afternoon shift on the same day (domain constraint) and that Vicky's preference was given a higher priority that day.

Now, assume that the access-rights function α is defined such that Thanos has access-rights to the domain constraints and Rose's constraints, but not to the constraints of other agents. In this case, the privacy loss ρ_1 of both explanations would be calculated as follows:

- $\rho_1(\epsilon_{r1}) = |\epsilon_{r1}| - \sum_{c \in \epsilon_{r1}} \alpha(1, c) = 2 - 2 = 0$, since Thanos has access to Rose's information.
- $\rho_1(\epsilon_{r2}) = |\epsilon_{r2}| - \sum_{c \in \epsilon_{r2}} \alpha(1, c) = 2 - 1 = 1$, since Thanos does not have access to Vicky's information.

As $\rho_1(\epsilon_{r1}) < \rho_1(\epsilon_{r2})$, the privacy-aware explanation in this case would be ϵ_{r1}.

5 QUERIES: Computing Explanations

We now present the *Question Understanding and Efficient Response with Intelligible Explanations of Schedules* (QUERIES) algorithm, which generates privacy-aware explanations ϵ_i^* for reason-seeking and modification-seeking queries φ of an agent a_i. The core of QUERIES is based on *reasoning via inconsistency*. In particular, it leverages a set of methods that are directly applicable to logic-based explanation generation problems, namely, *minimal unsatisfiable sets* (MUS) and *minimal correction sets* (MCS) [25, 29], both of which emerge when a set of clauses is unsatisfiable. Particularly, an MUS

[2] For more details on the encoding, please refer to the supplement available at https://github.com/YODA-Lab/QUERIES.

Algorithm 1: QUERIES Algorithm

Input: $KB, \varphi, a_i, \alpha, k$
Result: *privacy-aware explanation* ϵ for φ for a_i

```
forall c ∈ KB do
    if α(a_i, c) = 1 then
        assign weight k to c
if φ is a reason-seeking query then
    ε ← getMUS(KB, φ)
else if φ is a modification-seeking query then
    ε ← getMCS(KB, φ)
return ε
```

can be interpreted as explaining why a set of clauses is unsatisfiable by identifying a minimal set of conflicting clauses that cause the unsatisfiability. An MUS can then be used to find a reason-seeking explanation:

Proposition 1. *Given a knowledge base KB and a reason-seeking query φ_r, $\epsilon_r = M \setminus \{\neg\varphi_r\}$ is a reason-seeking explanation for φ_r if M is an MUS of $KB \cup \{\neg\varphi_r\}$.*

PROOF (SKETCH). The existence of a reason-seeking query φ_r implies that $KB \models \varphi_r$, which in turn implies that $KB \cup \{\neg\varphi_r\} \models \bot$ according to the definition of entailment. That is, the negation of φ_r is inconsistent with a set of constraints from KB and, as such, an MUS M of $KB \cup \{\neg\varphi_r\}$ exists. If $\neg\varphi_r \in M$, then $M \setminus \{\neg\varphi_r\}$ is satisfiable and $M \setminus \{\neg\varphi_r\} \models \varphi_r$. Therefore, $M \setminus \{\neg\varphi_r\}$ is a reason-seeking explanation for φ_r. □

Similarly, an MCS explains how to restore consistency in an inconsistent KB by identifying a minimal set of clauses from KB such that when removed, KB becomes satisfiable. A modification-seeking explanation can be then be generated via an MCS:

Proposition 2. *Given a knowledge base KB and a modification-seeking query φ_m, C is a modification-seeking explanation for φ_m if C is an MCS of $KB \cup \{\varphi_m\}$ and $\varphi_m \notin C$.*

The proof of Proposition 2 follows from the fact that a modification-seeking explanation for φ_m is indeed an MCS of $KB \cup \{\varphi_m\}$.

Algorithm 1 presents the pseudocode of QUERIES, which generates explanations for an agent a_i. At a high level, it iterates over all constraints in KB and assigns large weights $k >> 1$ to constraints that are public to agent a_i with respect to access-rights function α. Then, the MUS (or MCS) solver prioritizes the constraints with the largest weights, which means that the output of the solver is a set of constraints with the largest cumulative sum of weights (i.e., privacy-aware explanation).

The completeness of QUERIES lies in the assumption we made for the two query types, which is that an explanation for both query types always exists. The correctness of QUERIES lies in the correctness of the MUS and MCS solvers and the assumption that k is sufficiently large such that explanations with the largest cumulative sum of weights are privacy-aware explanations.

6 Empirical Evaluations

We now empirically evaluate our approach both in simulated computational experiments as well as in a human user study.

6.1 Computational Evaluation

We now present a computational evaluation of QUERIES for the following four queries, two for each query type, where C_a is an agent's clause and Σ an infeasible schedule:[3]

- Reason-seeking query (agent): *Why is C_a unsatisfied?*
- Modification-seeking query (agent): *How to satisfy C_a?*
- Reason-seeking query (schedule): *Why is Σ infeasible?*
- Modification-seeking query (schedule): *How to make Σ feasible?*

We ran our experiments on a MacBook Pro machine comprising an M1 Max processor with 32GB of memory. The time limit was set to $500s$. Our implementation of QUERIES is written in Python and integrates calls to MUS and MCS oracles through the PySAT toolkit [20].[4]

To comprehensively evaluate our approach, we ran three sets of experiments: (1) To demonstrate the scalability of our approach, we evaluated it on our motivating employee shift assignment problem of varying size; (2) To demonstrate the impact of privacy or access rights, we evaluated our algorithm on the same scheduling problem, but agents have varying access rights; and (3) To demonstrate the generality of our approach, we evaluated it on an SMT-based encoding of the job-shop scheduling problem.

Experiment 1: Scalability: In this experiment, we vary the scale and complexity of the agent scheduling problem by varying the number of agents $|A|$, resources $|R|$, and time steps $|S|$ in the problem. Specifically, we created 14 random instances, where each instance has $|A| = 10 \cdot i$ agents, $|R| = 10 \cdot i$ resources, and $|S| = 10$ time steps, with i taking the values $1, 1.5, 2, \ldots, 7.5$. For the domain constraints, we extended the ones described in Section 2 to include more agents, shift types, and time steps, as well as included an additional constraint describing the maximum number of consecutive shifts an employee can undertake without a day off. For the agent constraints, we generated 5 types of constraints to reflect different kinds of preferences similar to those presented in Section 2, and randomly assigned them to the agents. We set the fraction $p = 0.5$ of agents that each agent has access rights to. If an agent a_i has access rights to agent a_j, then a_i is aware of all of agent a_j's constraints.

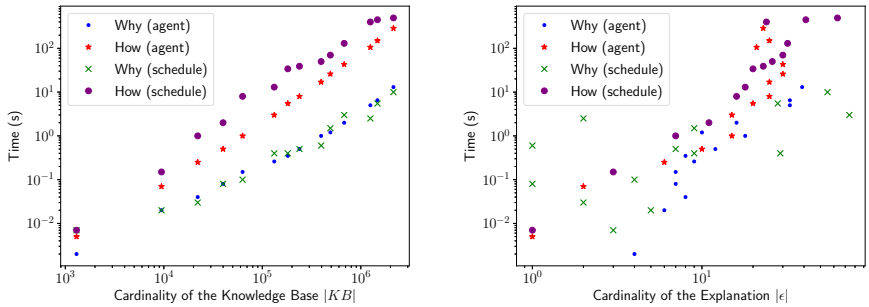
Figures 3(a) and 3(b) plot the runtimes of QUERIES as a function of the cardinalities of the knowledge base $|KB|$ and the explanation $|\epsilon|$ found, respectively. Unsurprisingly, the runtimes increase as the cardinalities increase. The reason is that the search space grows with $|KB|$, also reflected in $|\epsilon|$. Also, modification-seeking queries took longer to solve than reason-seeking queries. The reason is that our off-the-shelf MCS solver, used for modification-seeking queries, is less efficient than our off-the-shelf MUS solver, used for reason-seeking queries.

Experiment 2: Access Rights: In this experiment, we use the same employee shift assignment problem, where we set the number of agents $|A| = 40$, resources $|R| = 40$, and time steps $|S| = 5$. We vary the fraction $p = \{0, 0.1, 0.2, \ldots, 1\}$ of other agents that each agent has access rights to.

Figures 4(a), 4(b), and 4(c) plot, as a function of access rights fraction p, the runtimes of QUERIES, privacy losses $\rho_i(\epsilon)$ of explanations, and cardinality of explanations $|\epsilon|$, respectively. Similar to the previous experiment, the runtimes are larger for modification-

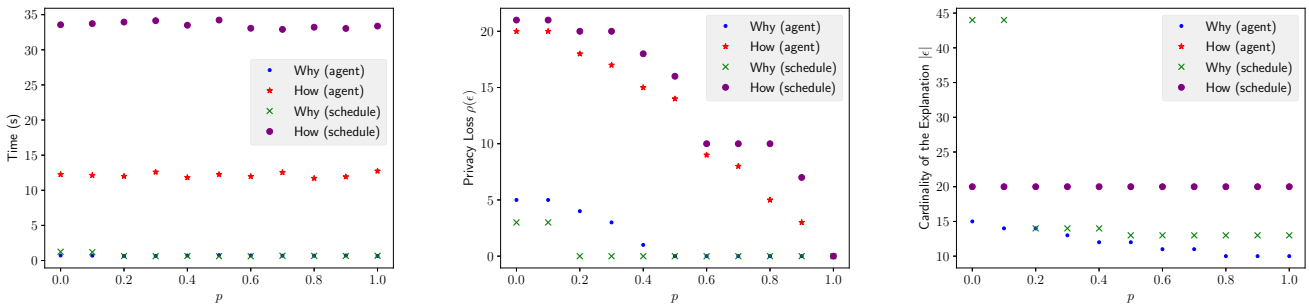
[3] C_a was randomly selected from a pool of unsatisfied clauses of agent a and Σ was generated by randomly flipping 20% of the values of a feasible schedule.

[4] The code repository is available at https://github.com/YODA-Lab/QUERIES.



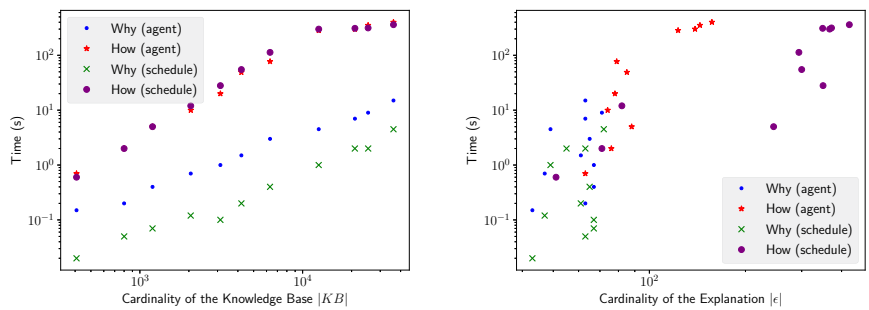
(a) (b)

Figure 3: Results of Experiment 1 on the Scalability of QUERIES



(a) (b) (c)

Figure 4: Results of Experiment 2 on the Impact of Privacy and Access Rights



(a) (b)

Figure 5: Results of Experiment 3 on SMT-based Encoding of Job-Shop Scheduling

seeking queries than reason-seeking queries. However, unlike the previous experiment, there is a significant difference in $|\epsilon|$ for the different queries in this experiment. As the modification-seeking queries required longer explanations, they took longer to solve than reason-seeking queries.

Additionally, the runtimes stay relatively constant for all values of p, reflecting the fact that the runtimes for the MCS and MUS computations are independent of the weights of the clauses. Also, as expected, the privacy loss decreases as p increases since fewer clauses are private as p increases. Finally, as p increases, $|\epsilon|$ either decreases or remains constant, indicating that the solver can find shorter (i.e., better) explanations when the explanation space expands with larger values of p.

Experiment 3: SMT and Job-Shop Scheduling: Finally, to demonstrate that our explainable scheduling framework and algorithm can be generalized to other scheduling problems as well as other types of logic aside from propositional logic, we evaluate our approach on a *Satisfability Modulo Theory* (SMT) encoding of the *job-shop scheduling problem* [30]. SMT is a decision problem that extends Boolean logic and allows for richer representations of real-world problems with logical formulae that are based on a combination of background theories such as integers and reals [13].

The job-shop scheduling problem involves assigning a set of jobs, each with its own processing time, to machines in a way that ensures all jobs are completed. We encoded this problem in Python using the Z3 solver [12], and generated 11 instances by varying the number of jobs, processing times, and machines. For the MUS and MCS solvers, we used off-the-shelf implementations available within Z3. Similar to the previous experiment, we generated queries with an unsatisfied constraint and an infeasible schedule.

Figures 5(a) and 5(b) plot the runtimes of QUERIES as a function of the cardinalities $|KB|$ and $|\epsilon|$, respectively. We observed trends similar to those in Experiment 1, attributable to the same reasons described earlier."

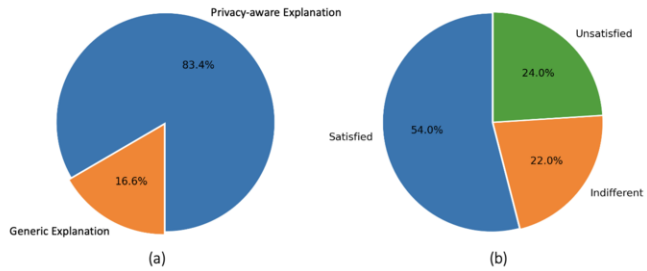


Figure 6: Human user study results from 60 users: (a) Percentage of users that selected generic and privacy-aware explanations; and (b) Percentage of users that were satisfied, indifferent, or unsatisfied with the privacy-aware explanation.

6.2 *Human User Study*

We now present a user study aimed at examining the assumptions made in our framework. In particular, we hypothesize:

> *Within agent scheduling problems, individuals prefer explanations containing only public information (e.g., publicly acknowledged rules and constraints) over those including private information (e.g., other employees' names and personal constraints), as they perceive them as more satisfactory.*

To evaluate this hypothesis, we conducted a human user study involving 60 English-speaking participants recruited through the online platform Prolific [26]. The study is centered around the employee shift assignment problem introduced earlier, with participants engaging in a thought experiment by assuming the role of an employee in a hypothetical company.

We informed the participants that Alice, an automated scheduling agent, was responsible for creating a schedule under the previously described domain constraints, ensuring that this information was public and known to all users. Participants were asked to choose a personal constraint from four available options, making them aware of only their own personal constraint, while the remaining agent constraints were considered private information. The participants then received their shift assignments, and were notified that their personal constraint was not satisfied in Alice's schedule.

Their primary task was to select an explanation out of two options: a *generic explanation*, which contained another employee's name and private constraint as the reason for their unsatisfied constraint, and a *privacy-aware explanation*, which included only a public domain constraint. Participants then answered questions about their choice of explanation and their satisfaction levels.

Figure 6 presents the main results of the study. The majority (83.4%) of participants preferred the privacy-aware explanation (Figure 6(a)). Among those who chose the privacy-aware explanation, 54% were satisfied, while the remaining participants were either indifferent (22%) or unsatisfied (24%), as shown in Figure 6(b). In the analysis of responses to the justification question, i.e., "why they selected the particular explanation", we observed a common trend: the privacy-aware explanation was considered more "informative" and "equitable" to all employees. Here, informative meant that it contained well-justified rules (i.e., constraints known to them), while "equitable" implied that it was not personal in the sense that it did not disclose other employees' information. Finally, when asked whether an explanation for a scheduling decision should include only public information, only private information, or a combination of both, the vast majority (88%) responded that only public information should be included, while the remaining participants (12%) suggested a combination of both public and private information.

In conclusion, our study supports the hypothesis that individuals prefer explanations containing only public information, which they perceive as not only more satisfactory but also more equitable.. Based on these findings, our explanation generation framework is designed to align with people's expectations for a scheduling decision explanation in this particular context.

7 Related Work

There is a small body of literature on *explainable scheduling*, with EXPRES [28] being the most relevant related work. It uses a MILP to find explanations for unsatisfied user preferences. Nevertheless, it is limited to only identifying a set of reasons for unsatisfied user preferences, thus lacking the ability to address and explain other types of queries, such as how (or why) a schedule can be (or is) (in)feasible. With regards to privacy, EXPRES preserves privacy by post-processing explanations to remove identifying reference to agents. In contrast, we give a more thorough treatment on this issue as we found that it is key to users in our user study. On a similar thread, Cyras *et al.* [11] proposed an argumentation-based approach for explaining why a schedule is (or not) feasible and why a preference was unsatisfied in the schedule, as we also tackle in this paper. The key differences between their approach and ours is that they do not consider any privacy preservation strategies, they are restricted to makespan scheduling problems, and they did not provide any experimental evaluation of their approach. Finally, Agrawal *et al.* [1] and Bertolucci *et al.* [3] also consider the problem of explaining scheduling decisions, however, their scope is limited to specific domain applications – scheduling Mars rovers and operating rooms, respectively.

A related research area is *explainable planning*, which has a larger body of work. Most of the approaches in this area aim at explaining planning-specific queries, such as why a plan is feasible/optimal and why a particular action is (or not) included in a plan [7, 16, 31, 32, 37, 39]. Closely related is the work by Vasileiou *et al.* [35], which also uses minimal correction sets (MCS) and minimal unsatisfiable sets (MUS) to find explanations. However, the key differences between their approach and ours is that they do not consider privacy preservation and they take a philosophically different approach of finding explanations by reconciling the differences between the mental models of the explainer and explainee. Finally, for a further exposition on the relationship between our approach and previous works such as diagnosis and MUS generation, we refer the reader to the work by Vasileiou *et al.* [35, 36].

8 Discussion

Privacy: Despite optimizing for privacy, explanations may still contain private constraints with respect to the explainee. As such, privacy leakage can occur when these explanations are relayed to the explainee. To address this issue and preserve the agents' privacy, we can post-process the explanation by abstracting away the remaining private constraints. This process can take different forms, such as *masking* all identifying references to the agents' whose private constraints are included in the explanation or by completely *retracting* the private constraints from the explanation.

As an example, consider that Thanos has no access rights to any of the agent constraints. Then, the reason-seeking explanation

$\epsilon_r = \{x_{4,1,2}, \neg x_{4,1,2} \vee \neg x_{1,1,2}\}$ that is generated for him unfortunately includes Rose's identity and private constraint (= $x_{4,1,2}$). Post-processing ϵ_r will allow us to retract $x_{4,1,2}$ from ϵ_r and mask the identity of Rose from the remaining clause $\neg x_{4,1,2} \vee \neg x_{1,1,2}$, for example, by transforming the clause to its generalized form $atmost_1(\{x_{1,j,t}, x_{2,j,t}, x_{3,j,t}, x_{4,j,t}\})\ _{\forall r_j \in R, s_t \in S}$ (domain constraint C_3).

Explanation Delivery: After the (potential) abstraction phase, the (post-processed) explanation needs to be communicated to the agent. Unless the explainee agent is a domain expert, the explanation should not be communicated in a logical representation, but rather in a human-understandable format such as natural language. A trivial direction could be to leverage the expressivity and symbolic nature of logic. That is, we can define natural language templates and use them to map the generated explanations. In particular, notice that each constraint "symbolizes" a specific constraint type and is grounded on (propositional) variables, with each variable denoting a scheduling element such as an agent, a resource, or a time step. For instance, $\epsilon_r = \{x_{4,1,2}, \neg x_{4,1,2} \vee \neg x_{1,1,2}\}$ says that Rose is assigned the morning shift on Tuesday ($x_{4,1,2}$), and that either Rose or Thanos can be assigned a morning shift on Tuesday ($\{\neg x_{4,1,2} \vee \neg x_{1,1,2}\}$). As such, a logic-based explanation can be transformed into a natural language explanation by identifying and mapping the constraints to their respective pre-defined, natural language templates. Another possibility is to leverage Large Language Models (LLMs) [6] to translate logical explanations into natural language. However, the accuracy of such translations will need to be validated through additional research as LLMs have been shown to have hallucination issues [40]. Another approach is through visualization systems [22, 33], though these systems will likely need to be crafted with significant domain expertise.

Ethical Considerations: It is paramount to assess the ethical implications of our work. In our context, two ethical considerations emerge – the explanation unavoidably involves private information, and the fair resolution of conflicting agent constraints. The former concern can be addressed by the post-processing mechanisms described above. For the latter, while we do not address the issue directly in our work, we imagine that fairness could be achieved by employing multi-objective optimization techniques [15, 17, 19] that seek a balance among conflicting constraints.

Although our current framework does not present definitive solutions to these complex issues, these potential directions could guide the future trajectory of research in this field. Subsequent iterations should integrate these considerations, working towards not just practical but also ethically robust AI explanation systems.

9 Conclusions

In this paper, we tackled the challenge of generating explanations for agent scheduling problems. We proposed a logic-based framework capable of generating privacy-aware explanations for reason-seeking and modification-seeking queries. To the best of our knowledge, our framework is the first to present a general approach that tackles a broad spectrum of agent scheduling problems while quantifying and optimizing for privacy. Our experimental results demonstrate the efficacy of our framework, and our user study supports the importance of privacy, fairness, and informativeness in explanation generation for scheduling systems.

Acknowledgments

This research is partially supported by the National Science Foundation under awards 1812619 and 2232055. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the United States government.

References

[1] Jagriti Agrawal, Amruta Yelamanchili, and Steve Chien, 'Using explainable scheduling for the Mars 2020 rover mission', *arXiv preprint arXiv:2011.08733*, (2020).

[2] Carlos Ansótegui, Miquel Bofill, Miquel Palahí, Josep Suy, and Mateu Villaret, 'Satisfiability modulo theories: An efficient approach for the resource-constrained project scheduling problem', in *Proceedings of the Symposium on Abstraction, Reformulation and Approximation (SARA)*, pp. 2–9, (2011).

[3] Riccardo Bertolucci, Carmine Dodaro, Giuseppe Galatà, Marco Maratea, Ivan Porro, and Francesco Ricca, 'Explaining ASP-based operating room schedules', in *Proceedings of the Workshop on Explainable Logic-Based Knowledge Representation*, (2021).

[4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, *Handbook of Satisfiability*, volume 336, IOS press, 2021.

[5] Miquel Bofill, Marc Garcia, Josep Suy, and Mateu Villaret, 'MaxSAT-based scheduling of B2B meetings', in *Proceedings of the International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR)*, pp. 65–73, (2015).

[6] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al., 'On the opportunities and risks of foundation models', *arXiv preprint arXiv:2108.07258*, (2021).

[7] Tathagata Chakraborti, Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati, 'Plan explanations as model reconciliation: Moving beyond explanation as soliloquy', in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 156–163, (2017).

[8] Wayne Chi, Steve Chien, and Jagriti Agrawal, 'Scheduling with complex consumptive resources for a planetary rover', in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 348–356, (2020).

[9] Stephen Cook, 'The complexity of theorem-proving procedures', in *ACM Symposium on Theory of Computing (STOC)*, pp. 151–158, (1971).

[10] James Crawford and Andrew Baker, 'Experimental results on the application of satisfiability algorithms to scheduling problems', in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 1092–1097, (1994).

[11] Kristijonas Cyras, Dimitrios Letsios, Ruth Misener, and Francesca Toni, 'Argumentation for explainable scheduling', in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2752–2759, (2019).

[12] Leonardo De Moura and Nikolaj Bjørner, 'Z3: An efficient SMT solver', in *Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 337–340, (2008).

[13] Leonardo De Moura and Nikolaj Bjørner, 'Satisfiability modulo theories: introduction and applications', *Communications of the ACM*, **54**(9), 69–77, (2011).

[14] Emir Demirović, Nysret Musliu, and Felix Winter, 'Modeling and solving staff scheduling with partial weighted MaxSAT', *Annals of Operations Research*, **275**, 79–99, (2019).

[15] Michael Emmerich and André Deutz, 'A tutorial on multiobjective optimization: Fundamentals and evolutionary methods', *Natural Computing*, **17**(3), 585–609, (2018).

[16] Maria Fox, Derek Long, and Daniele Magazzeni, 'Explainable planning', *arXiv preprint arXiv:1709.10256*, (2017).

[17] Nyoman Gunantara, 'A review of multi-objective optimization: Methods and its applications', *Cogent Engineering*, **5**(1), 1502242, (2018).

[18] Stefaan Haspeslagh, Tommy Messelis, Greet Vanden Berghe, and Patrick De Causmaecker, 'An efficient translation scheme for representing nurse rostering problems as satisfiability problems', in *Proceedings*

of the International Conference on Agents and Artificial Intelligence (ICAART), pp. 303–310, (2013).

[19] Carlos Hernández, William Yeoh, Jorge A. Baier, Han Zhang, Luis Suazo, Sven Koenig, and Oren Salzman, 'Simple and efficient bi-objective search algorithms via fast dominance checks', *Artificial Intelligence*, **314**, 103807, (2023).

[20] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva, 'PySAT: A Python toolkit for prototyping with SAT oracles', in *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pp. 428–437, (2018).

[21] Miyuki Koshimura, Hidetomo Nabeshima, Hiroshi Fujita, and Ryuzo Hasegawa, 'Solving open job-shop scheduling problems by SAT encoding', *IEICE Transactions on Information and Systems*, **93**(8), 2316–2318, (2010).

[22] Ashwin Kumar, Stylianos Loukas Vasileiou, Melanie Bancilhon, Alvitta Ottley, and William Yeoh, 'VizXP: A visualization framework for conveying explanations to users in model reconciliation problems', in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 701–709, (2022).

[23] Sudip Kundu and Sriyankar Acharyya, 'Stochastic local search approaches in solving the nurse scheduling problem', in *Proceedings of the International Confernece on Computer Information Systems - Analysis and Technologies (CISIM)*, pp. 202–211.

[24] Chu Min Li and Felip Manya, 'MaxSAT, hard and soft constraints', in *Handbook of Satisfiability*, 903–927, IOS press, (2021).

[25] João Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov, 'On computing minimal correction subsets', in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 615–622, (2013).

[26] Stefan Palan and Christian Schitter, 'Prolific: A subject pool for online experiments', *Journal of Behavioral and Experimental Finance*, **17**, 22–27, (2018).

[27] Jose Pinto and Ignacio Grossmann, 'A logic-based approach to scheduling problems with resource constraints', *Computers & Chemical Engineering*, **21**(8), 801–818, (1997).

[28] Alberto Pozanco, Francesca Mosca, Parisa Zehtabi, Daniele Magazzeni, and Sarit Kraus, 'Explaining preference-driven schedules: the expres framework', in *Proceedings of the International Conference on Automated Planning and Scheduling*, pp. 710–718, (2022).

[29] Alessandro Previti and João Marques-Silva, 'Partial MUS enumeration', in *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pp. 818–825, (2013).

[30] Sabino Francesco Roselli, Kristofer Bengtsson, and Knut Åkesson, 'SMT solvers for job-shop scheduling problems: Models comparison and performance evaluation', in *International Conference on Automation Science and Engineering (CASE)*, pp. 547–552, (2018).

[31] Tran Cao Son, Van Nguyen, Stylianos Loukas Vasileiou, and William Yeoh, 'Model reconciliation in logic programs', in *European Conference on Logics in Artificial Intelligence (JELIA)*, pp. 393–406, (2021).

[32] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati, 'The emerging landscape of explainable automated planning & decision making', in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4803–4811, (2020).

[33] Karthik Valmeekam, Sarath Sreedharan, Sailik Sengupta, and Subbarao Kambhampati, 'RADAR-X: an interactive mixed initiative planning interface pairing contrastive explanations and revised plan suggestions', in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 508–517, (2022).

[34] Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck, 'Personnel scheduling: A literature review', *European Journal of Operational Research*, **226**(3), 367–385, (2013).

[35] Stylianos Loukas Vasileiou, Alessandro Previti, and William Yeoh, 'On exploiting hitting sets for model reconciliation', in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 6514–6521, (2021).

[36] Stylianos Loukas Vasileiou, William Yeoh, and Tran Cao Son, 'On the relationship between KR approaches for explainable planning', *arXiv preprint arXiv:2011.09006*, (2020).

[37] Stylianos Loukas Vasileiou, William Yeoh, Tran Cao Son, Ashwin Kumar, Michael Cashmore, and Daniele Magazzeni, 'A logic-based explanation generation framework for classical and hybrid planning problems', *Journal of Artificial Intelligence Research*, **73**, 1473–1534, (2022).

[38] Jean-Paul Watson, J. Christopher Beck, Adele Howe, and L. Darrell Whitley, 'Problem difficulty for tabu search in job-shop scheduling', *Artificial Intelligence*, **143**(2), 189–217, (2003).

[39] Yu Zhang, Sarath Sreedharan, Anagha Kulkarni, Tathagata Chakraborti, Hankz Hankui Zhuo, and Subbarao Kambhampati, 'Plan explicability and predictability for robot task planning', in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pp. 1313–1320, (2017).

[40] Terry Yue Zhuo, Yujin Huang, Chunyang Chen, and Zhenchang Xing, 'Red teaming ChatGPT via jailbreaking: Bias, robustness, reliability and toxicity', *arXiv preprint arXiv:2301.12867*, (2023).