ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240976

Enabling MCTS Explainability for Sequential Planning Through Computation Tree Logic

Ziyan An[a], Hendrik Baier[b], Abhishek Dubey[a], Ayan Mukhopadhyay[a] and Meiyi Ma[a,*]

[a]Vanderbilt University
[b]Eindhoven University of Technology

Abstract. Monte Carlo tree search (MCTS) is one of the most capable online search algorithms for sequential planning tasks, with significant applications in areas such as resource allocation and transit planning. Despite its strong performance in real-world deployment, the inherent complexity of MCTS makes it challenging to understand for users without technical background. This paper considers the use of MCTS in transportation routing services, where the algorithm is integrated to develop optimized route plans. These plans are required to meet a range of constraints and requirements simultaneously, further complicating the task of explaining the algorithm's operation in real-world contexts. To address this critical research gap, we introduce a novel computation tree logic-based explainer for MCTS. Our framework begins by taking user-defined requirements and translating them into rigorous logic specifications through the use of language templates. Then, our explainer incorporates a logic verification and quantitative evaluation module that validates the states and actions traversed by the MCTS algorithm. The outcomes of this analysis are then rendered into human-readable descriptive text using a second set of language templates. The user satisfaction of our approach was assessed through a survey with 82 participants. The results indicated that our explanatory approach significantly outperforms other baselines in user preference.

1 Introduction

Artificial Intelligence (AI) is now intricately woven into the fabric of our daily lives. AI's influence extends into virtually every sector, redefining the way we work, learn, and interact with the world around us. For example, AI algorithms play a pivotal role in personalized recommendations, virtual social networks, medical diagnosis, transportation management, and city services [12, 26, 28, 2]. As AI becomes ubiquitous, explainability of the underlying algorithmic processes is imperative for fostering trust and ensuring transparency in automated decision-making processes [31, 29], thereby enabling stakeholders to comprehend and effectively scrutinize autonomous and data-driven systems. The importance of developing transparent and explainable AI systems has been highlighted and, to some extent, mandated in public policy—both the European Union and the US have highlighted its importance through legislation [8, 33].

The field of Explainable AI has acknowledged this challenge and developed several novel and effective approaches. Indeed, there are several well-known approaches for interpreting learning-based methods, i.e., methods that use historical data to make estimates about a dependent variable as a function of independent variables, such as SHAP [25] and LIME [34]. Unfortunately, there is a major lack of approaches that can explain sequential decision-making. Such AI-based methods significantly differ from standard supervised learning since the algorithms for sequential decision-making conduct a search through a complex combinatorial space of trajectories (i.e., sequential states and actions) to find optimal policies; as a result, standard approaches that can be used to explain supervised learning are not directly applicable to sequential decision making.

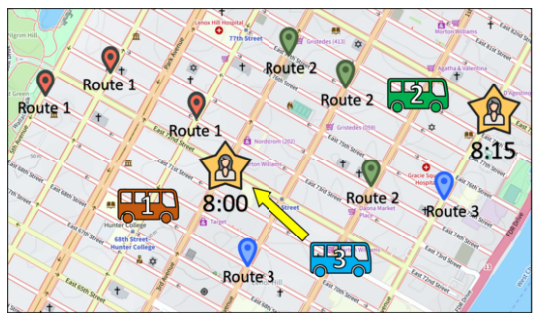


Figure 1. A running example of a trip request and the recommended route plan. Available vehicles are denoted by **bus icons** and the outstanding request is represented by the **passenger icon**. **Stops**, marked by route numbers, will be visited by vehicles with corresponding IDs.

In this paper, we focus on online search, which is at the core of recent advances in autonomous planning and decision-making [35]. Specifically, we focus on Monte Carlo tree search (MCTS) [22], an online and anytime sampling-based general search algorithm that has played a critical role in achieving state-of-the-art performance in games such as Chess and Go [38] and also in many real-world domains such as emergency response [32], public transit [42], and supply chain management [13]. While Baier and Kaisers highlight the need for explainable search [6, 7], there are *no well-established approaches* for explaining decisions computed by MCTS.

To ground our framework, we use a complex real-world domain instead of board games. In particular, we choose public transportation, where algorithmic and data-driven approaches have shown particularly promising results [36, 42]. However, transportation has traditionally involved human operators optimizing decisions manually; as a result, AI-based approaches are often viewed with skepticism even when they demonstrate higher efficiency. This lack of trust primarily stems from a lack of understanding of the algorithms.

Consider a scenario in dynamic vehicle routing. As illustrated in Figure 1, the path planning algorithm is responsible for assigning a

* Corresponding Author: meiyi.ma@vanderbilt.edu

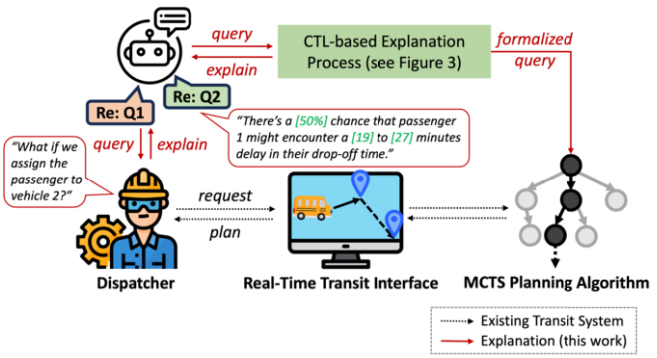


Figure 2. Overview of the CTL-based explainer.

passenger request to one of three available vehicles. The algorithm is equipped with real-time information, including the current locations of all vehicles and their availability status. While generating *plans*, the algorithm is guided by specific objectives, such as maximizing the overall percentage of completed trips. Moreover, it is provided with certain non-negotiable constraints that must not be violated during the planning process. Leveraging its internal mechanisms, the planning algorithm generates the best approximation to an optimal solution it can find and presents the user with the corresponding result, which is to assign the passenger to vehicle 3 (blue).

In this case, the algorithm is a black-box when presented to users such as route dispatchers who may not have a technical background in search algorithms. Furthermore, the dispatchers remain uninformed about the potential future status of other vehicles in the vicinity. Without this knowledge, they are unable to determine if alternative nearby vehicles could provide better solutions to the current routing problem, which might also significantly benefit future operations. We tackle this challenge by developing computation tree logic (CTL)-based explainable MCTS. Particularly, our focus is on non-technical route dispatchers as explainees, who are seeking to understand the decision-making outcomes. They may raise inquiries such as "why was this particular action recommended?" or "why wouldn't this alternative action be viable?"

Our explainer categorizes user queries into three varieties: *factual queries* that provide insights into the algorithm's decisions derived from states and actions in the search tree, *contrastive queries* that compare user-suggested decisions with the algorithm's recommendations by identifying CTL violations or inferior rewards, and *alternative plan explanations* that involve extending the search to additional options. Specifically, our explainer first leverages language templates to capture user queries about the outcomes of the MCTS algorithm. This design choice considers the needs of non-technical dispatchers by helping them in formulating the most relevant questions, excluding queries that do not directly relate to the core aspects of the planning problem. Second, each templated natural language query is converted into a logic specification by a many-to-many mapping process. For instance, consider a user query such as, *"Based on the suggested vehicle assignment, is it expected that the passenger will be dropped off too late?"* Our explainer translates this query into CTL as $\phi = AG\,(t_{est.} > t_{allowed})$.

Third, our explainer triggers a formal specification checking mechanism of CTL that takes logic formulas as input and traverses the MCTS tree to generate results. Fourth, the checking and verification results obtained in this step are then translated back into natural language. This conversion, again, leverages pre-defined language templates that are designed to ensure the output is understandable to the user. Continuing from the previous example, a response to the query would be: "The passenger has specified a desired drop-off time of 5:33 PM. The route planning algorithm has simulated approximately 150 potential future scenarios and requests. *Potential Late Arrival:* There's a chance that the passenger might encounter a delay in their drop-off time. This expected delay averages around 23 minutes. The primary reason for this delay is that the proposed vehicle is expected to make stops at about 4 other locations prior to reaching the passenger's drop-off point. However, the delay can be as short as 19 minutes or extend up to 27 minutes. The percentage of times the suggested vehicle doesn't meet the desired drop-off time is about 10%." In summary, we offer the following *contributions:*

- Our explainer dynamically associates non-technical user queries with three types. These queries, containing free variables, are then converted into CTL formulas for further analysis.
- Our explainer utilizes CTL semantics to determine the satisfiability of the logic query. It then translates these results into natural language using linguistic templates, ensuring the algorithm's explanations are easily comprehensible and accessible.
- We recruited a total of 82 participants for the study (approved by the Institutional Review Board (IRB)). The CTL-based explainer significantly outperforms the two baseline methods in reported understanding, satisfaction, completeness, and reliability.

2 Sequential Planning in Public Transit

In this section, we introduce the underlying Markov Decision Process (MDP) that models our specific use case in public transit, and defines the search space for our MCTS implementation. Following related work [19, 42] in using MDPs for transit planning, we define a transit planning task Π as the tuple $\langle S, A, T, R, \gamma \rangle$, where S is the set of environment states, A the set of available actions, and $T(s_t, a_t, s_{t+1})$ defines the probabilities of moving from state s_t to state s_{t+1} under action a_t, where every transition to a next state implies a new predicted request. The reward function $R(s, a)$ and the discount factor γ assign a scalar reward for taking action a in state s.

A state s in the transit planning task is defined by the tuple $\langle \theta, r, V, \{R^{|V|}\} \rangle$, where θ denotes the current route plans for all vehicles, r is an outstanding request, V denotes the locations of all vehicles, and $\{R^{|V|}\}$ is a list of assigned requests to each vehicle [42]. We denote the state at time t with $s_t = \langle \theta_t, r_t, V_t, \{R_t^{|V|}\} \rangle$. An action a_t at time t involves assigning an outstanding trip request r^j to a vehicle v^i. A transit service request r^j is defined by: the time the request was made t_r^j, requested pickup time t_p^j, requested drop-off time t_d^j, pickup location l_p^j, and drop-off location l_d^j, as well as its current status $u^j \in \{\text{waiting, assigned, in-transit, dropped-off}\}$. Each transit vehicle $v^i \in V$ has a fixed capacity c^i and an occupancy p^i that varies based on its current route plan. The designated route plan of each vehicle is denoted as θ^i, specifying a list of future locations that the vehicle is scheduled to visit.

We use MCTS [22] as the decision making algorithm, mapping the current state s_t to the next action to take a_t with the help of sampling-based planning over the MDP model described above. Our transit scenario operates within a dynamic environment where requests can be made at any moment. When a new request r is initiated by a passenger, the planning algorithm is engaged on-the-fly to generate a new action to accommodate this request. This time point is referred to as a "decision epoch" [21, 42], where each decision epoch addresses a new planning problem independently. Specifically,

Table 1. Examples of queries, state variables, and the completed CTL formulas.

Query Type	Query Example	State Variable	CTL Formula
T1: Factual query	**Q1:** Is it expected that the [passenger] will be [picked up] on time? (Efficiency)	t_{est} estimated travel time; t_p request picked up time; t_a allowed time window	$\phi_1 : AG\,(t_{\text{est}} \leq (t_p + t_a))$
T2: Contrastive query	**Q2:** Why wasn't the passenger assigned to [this alternative vehicle]? (Efficiency & Hard constraint)	v_c vehicle capacity; v_o vehicle occupancy; t_d request drop off time; t_{est}; t_p; t_a	$\phi_1; \phi_2 : AG\,(v_o \leq v_c)$; $\phi_3 : AG\,(t_{\text{est}} \leq (t_d + t_a))$
T3: Query with tree expansion	**Q3:** Can you tell me more about [this alternative route]? (Efficiency & Hard constraint & Soundness)	v_{tt} total travel time; v_{rt} reasonable timeframe; t_{est}; t_p; t_d; t_a; v_c; v_o	$\phi_1; \phi_2; \phi_3$; $\phi_4 : AG\,(v_{\text{tt}} \leq v_{\text{rt}})$

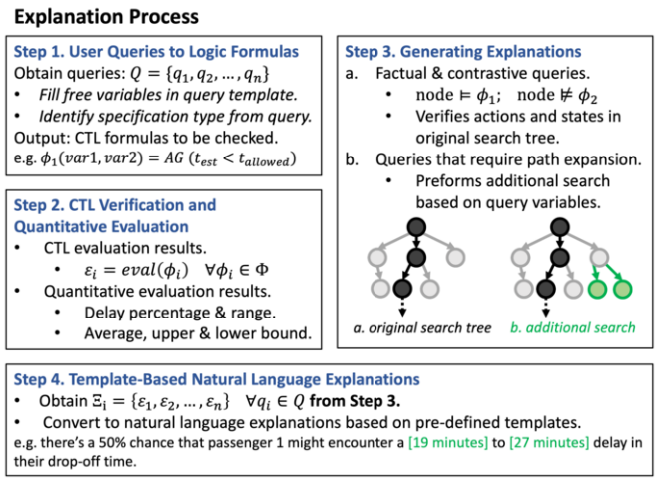


Figure 3. Illustration of the complete explanation process.

MCTS seeks to find or approximate the optimal action a_t that allocates the current request to one of the vehicles $v^i \in V$. Potential actions in this use case must adhere to two hard constraints: the capacity constraint $p_t^i \leq c^i, \ \forall v_i \in V$ and the timing constraint $\left| r^j_{\text{dropped-off}} - r^j_{\text{in-transit}} \right| \leq T_{\max}, \ \forall r^j \in R$, where p_t^i is the number of passengers for vehicle v^i at time t and T_{max} is the maximum allowed en-route time for a trip request, $r^j_{\text{dropped-off}}$ is the actual drop off time and $r^j_{\text{in-transit}}$ is the actual pick up time of request r^j. A request is successfully assigned to a vehicle if the action complies with all hard constraints, which can extend beyond the above and are explicitly defined by the user or the algorithm engineer prior to the planning task. To simplify the problem, we do not consider swapping route plans between vehicles. Therefore, the action space of each vehicle is restricted to one per single request by inserting the pickup and drop-off locations of r into its existing route at the index where the time used to travel between two stops is minimized. Assume that there are $|V|$ vehicles available, the maximum number of potential actions for assigning a particular request is $|V|$.

The reward of the MCTS planning algorithm is $R(s,a) = \gamma_1 \cdot \frac{N_t+N_d}{|R^{|V|}|} + \gamma_2 \cdot \sum_{i=1}^{N_t+N_d}(t_p^i - r^i_{\text{in-transit}}) + \gamma_3 \cdot \sum_{i=1}^{N_d}(t_d^i - r^i_{\text{dropped-off}})$, where N_t is the total number of in transit requests, N_d is the number of dropped off requests, γ_1, γ_2, and γ_3 are weights.

3 CTL-Based Explainable MCTS

The goal of CTL [14]-based explainable MCTS is to help users (e.g., transit operators) understand the relationship between the system state, the desired outcome, non-negotiable constraints, and the route plan produced by MCTS. In each decision epoch of the planning task Π, the user is presented with the current system state s_t and the action a_t proposed by MCTS. The user also has knowledge of a predefined set of hard constraints that the recommended actions must consistently adhere to. Given this information, the user poses a set of queries, denoted as $Q = \{q_1, q_2, \cdots\}$. Following the running example in Figure 1, one such query can be: "Why wasn't the passenger assigned to vehicle 2's route?" (Q2, Table 1).

Given the query, the explainer's goal is to provide explanations, denoted as Ξ, concerning recommended actions a_t in response to specific queries. The explainer must deduce answers to these questions based on criteria that include the goal of the search, the stringent state requirements such as vehicle capacity violations, and user-proposed criteria like additional efficiency requirements. In Figure 3, we provide a overview of this process. In the following sections, we describe the detailed methods employed to derive this response from the MCTS search tree.

3.1 Formulating User Queries

Dispatchers lacking technical knowledge in MCTS can inquire about various aspects of the planning result, such as constraint compliance, route plan efficiency, and its soundness. To address these varying interests, we organize user queries into three distinct categories. Table 1 demonstrates one example query of each category. In principle, our explanation framework can handle any query that can be structured in this way. In our examples, free variables are indicated by brackets. Furthermore, these queries are initially left empty and are associated with predefined criteria such as efficiency and soundness. When a user submits a query in natural language, the first step involves identifying the completed variables within the query. For instance, if a query asks about an alternative action, we map the free variables in the query to the corresponding state variables of the alternative action in question. Additionally, we incorporate these state variables and the query criteria into automatically generated CTL formulas, denoted as $\Phi = \{\phi_1, \phi_2, \cdots\}$. In Section 3.3, we describe how variables in queries are processed, incorporated into CTL formulas, and how their handling differs within the search tree.

3.2 CTL Verification and Quantitative Evaluation

Before discussing how specific queries and criteria are translated into CTL, we detail our approach to employing the branching-time logic extensively applied in model checking [10]. The CTL state quantifiers evaluate the breadth of the computation tree, where the A (for all) operator examines every child of a given state, and the E (exists) operator focuses on at least one child, requiring the formula's validity for any one of them. In contrast, the path quantifiers consider the tree's depth. The X (next) operator checks whether a specified condition holds in the next immediate state. The F (future) operator is concerned with a condition's eventual occurrence and the G (globally) operator requires that a condition holds true in all future states. The syntax of CTL is shown in Definition 1, where p is an atomic proposition, incorporating relational and arithmetic operators.

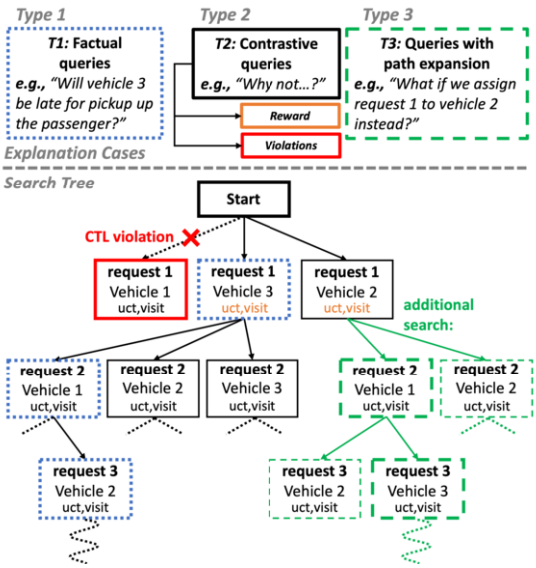


Figure 4. Illustration of different types of queries.

Definition 1 (Syntax of CTL formulas)**.**

$$\Phi ::= \top \mid \bot \mid p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid$$
$$AX\Phi \mid EX\Phi \mid AF\Phi \mid EF\Phi \mid AG\Phi \mid EG\Phi$$

Given a CTL formula and a MCTS search tree, the process of verifying whether the nodes and paths satisfy the formula is conducted through CTL semantics, known as CTL verification or specification checking. The input for this specification checking module is a finite sequence of possible states of the system $\{s_i\}$, where $i < L$ and L represents the length of the longest path within the data structure. This sequence describes discrete transitions $s_i \curvearrowright s_{i+1}$ with a time delay between two successive states, such that $t_{i+1} = t_i + d$ [1]. In our planning task, the search tree is conceptualized as a Kripke structure [23] where nodes symbolize states s_i in the decision-making process, and edges represent the available choices or actions at each node, leading to the next state s_{i+1} in the sequence. The CTL specification checking module yields a binary outcome, indicating whether the evaluated condition is satisfied or violated. To provide a quantitative assessment of each state where the CTL formula evaluates to false, we record the specific details of the violation, such as the extent of delay in minutes, at that particular state. This state is then marked with a flag to indicate a violation. This approach facilitates a thorough identification of non-compliant states, both qualitatively and quantitatively. The described process corresponds to lines 5-6 of Algorithm 2. It also enables for a more detailed transition to natural language explanations in subsequent processes, ensuring a comprehensive understanding of each violation.

3.3 Mapping Queries to CTL Formulas

Factual Queries — "Why?" Factual queries relate to reasons that indicate why a particular action is recommended [15, 37]. We specifically address the following types of questions in which individuals seek to determine whether the specific user-specified criteria will be met. For instance, a common question is, "Based on the current vehicle assignment, is it expected that passenger 1 will be dropped off too late?" For such queries, we employ the following structured language template: "Based on the current vehicle assignment, is it expected that [*passenger number*] will be [*action*] [*time*]?", where the *action* can be either dropped off or picked up, and *time* corresponds to either late or early. For the example query, we decompose the template-based query into the following tuple: [passenger 1, drop off, late]. As shown in Table 1, the specification type for this factual query is "Efficiency", and the queried state variable is "drop off" or t_d. Thus, the corresponding CTL formula regarding efficiency would be $\phi_1 : AG\,(t_{\text{est}} \leq (t_d + t_{\text{allowed}}))$, where the variable t_{est} represents the estimated time required for dropping off passenger 1. The CTL operator AG checks for all states and all future paths starting from the root. The relation $\leq$ in the formula is used to determine whether passenger 1 will arrive late, based on the comparison of the estimated travel time with the allowed time window.

Contrastive Queries — "Why Not?" The second type of query in our explainer involves alternative plans specified by technical users. In this case, we aim to explain why the planning algorithm did not recommend the alternative action a'_t as proposed by the user. We formulate the type of user queries about alternative plans with the following language template: "Why wasn't [*passenger*] assigned to [*another vehicle*] located at [*location*]?", where the missing variables can be represented as [passenger, another vehicle, location]. For example, a common query is "Why wasn't passenger 1 assigned to vehicle 1 (which is closer)?" In line with the nature of contrastive queries, the evaluation criteria for this query involve asking about potential violations of hard constraints and efficiency. More precisely, when querying about an alternative action a'_t, we can provide two types of reasons why MCTS did not produce a'_t. First, a'_t could be infeasible due to violations of hard constraints. Second, a'_t could be less efficient than the near-optimal solution a_t generated by MCTS. This evaluation includes several CTL formulas: $\phi_1 : AG, (t_{\text{est}} \leq (t_p + t_a))$, which examines efficiency in pickup times; $\phi_2 : AG, (t_{\text{est}} \leq (t_d + t_a))$, which addresses efficiency in drop-off times; and $\phi_3 : AG, (v_o \leq v_c)$, which checks for violations of capacity constraints. Initially, the search tree nodes are checked against these hard constraints. If no violations are found, the efficiency formulas are then evaluated.

Queries with Tree Expansion — "What If?" MCTS focuses on exploring the most promising paths, intentionally leading to less exploration of other, lower-scoring branches. Therefore, while extracting information from the search tree suffices for explaining recommended plans, it can provide limited insights for alternative plans, which might not have been explored enough [6]. For example, consider a user querying about an alternative action, such as assigning request 1 to vehicle 2 in Figure 1. In this case, while there are no direct violations, the user might question whether the algorithm considers the action less optimal due to insufficient exploration depth. To accommodate such queries, the MCTS explainer enables users to query about a broader set of potential actions that were neither recommended nor sufficiently explored. We formulate them as "Can you tell me more about assigning the [*passenger*] to [*another vehicle*] ?". For instance, a user may ask, "Can you tell me more about assigning passenger 1 to vehicle 2?" In this format, the free variables are encapsulated in the tuple: [passenger 1, vehicle 2]. This query type requires additional search and exploration to provide comprehensive explanations [7], as illustrated in Figure 4. A pseudocode of this process is provided in Appendix A1[1]. At a high level, to address such queries, we locate the under-explored node $N(s, a)$ in the original tree, where a is the alternative action being queried and s is the state

[1] Appendix available at https://arxiv.org/pdf/2407.10820.

Algorithm 1 CTL-based Explainable MCTS

for each decision epoch **do**
  $s_t = \langle \theta_t, r_t, V_t, \{R_t^{|V|}\} \rangle$
  `/* calculate route plan with MCTS */`
  a_t = MCTS(s_t)
  Obtain user queries $Q_t = \{q_1, q_2, \cdots\}$
  `/* process user queries */`
  for $q_i \in Q_t$ **do**
    $\Phi \leftarrow \{\}, \Xi \leftarrow \{\}$
    Identify specification type from query type
    $\Phi \leftarrow$ specification_type(q_i), query_variable(q_i)
    Calculate results: $\varepsilon_i = \textit{ExpGen}(\phi_i); \forall \phi_i \in \Phi$
    Append each ε_i to Ξ; output Ξ
  end for
  Apply a_t to route θ
end for

in the query. We then create a sub-tree by designating the previous path as the parent node and execute MCTS, continuing to explore possible scenarios and outcomes in response to the query.

3.4 Generating Natural Language Explanations

As outlined in Algorithm 1, the explainer is invoked once at the end of each decision epoch if there are user queries. Following the steps specified in Algorithm 2, the CTL-based explainer checks the search tree against each formula upon formulating the set of CTL formulas Φ. Each explanation in Ξ is obtained through CTL specification checking and quantitative evaluation. Specifically, at line 10 of Algorithm 2, we derive two key quantitative insights from the aggregated list of violations across all expanded nodes within the MCTS tree. First, we compute the violation percentage and the average degree of these violations. Second, for timing-related violations, we compute the temporal range of these violations, identifying both the earliest (lower bound) and latest (upper bound) occurrences. These results are then translated into a set of natural language explanations, denoted as $\Xi = \{\varepsilon_1, \varepsilon_2, \cdots\}$. The details of language templates are provided in Appendix A2.6. An example of these templates to address the query in the running example is: "Based on the set of scenarios examined by MCTS, there's a chance that the passenger might encounter a delay in their drop-off time. This expected delay averages around [23 minutes]. The primary reason for this delay is that the proposed vehicle is expected to make stops at about [4 other locations] prior to reaching the passenger's drop-off point. However, the delay can be as short as [19 minutes] or extend up to [27 minutes]. The percentage of times the suggested vehicle doesn't meet the desired drop-off time is about [10%]."

4 Evaluation

We evaluate the effectiveness of the explanations generated by our explainable framework in the context of five distinct vehicle routing scenarios [42]. through an IRB-approved user study. We aim to assess the overall quality and user-friendliness of the natural language explanations from the perspective of end users, comparing them with two other baseline methods. Additionally, the study aims to understand the participants' preferences among three types of queries.

4.1 Study Design

Testing Environment. Participants in this questionnaire-based user study first receive a comprehensive overview of the paratransit problem, which is designed to familiarize them with the key concepts. Then, participants are presented with five different scenarios of the route-planning problem. Each scenario involves different conditions and variables relevant to the problem, simulating real-world traffic planning situations. In the next step, the decision computed by the MCTS planning algorithm is shown to the participants. Finally, participants review three types of MCTS explanations regarding this decision. For each type of explanation, participants are asked to rate their quality using a pre-defined questionnaire. For all questions that require a rating, participants will be prompted to give their assessment using a 5-point Likert scale, where a rating of 1 is *strongly disagree* and a rating of 5 is *strongly agree*. Illustrative examples of the questionnaire is shown in Appendix A2.5.

Algorithm 2 *ExpGen*: Generate Explanations for One Query

1: Obtain ϕ
2: Initialize ε using explanation templates
3: Initialize potential violations $vio \leftarrow \{\}$
4: **for** each child node $n = (s, N, Q)$ in MCTS tree **do**
5: **if** $n \not\models \phi$ **then**
6: Add ϕ to potential violations vio
7: **end if**
8: **end for**
9: **if** violations vio are not empty **then**
10: Calculate explanations for vio and apply to templated ε
11: **end if**
12: **return** ε

Participants. The complete details and statistics of the participants were provided in Appendix A2.2. We recruited a total number of 82 eligible participants. Among the participating clients, 40.3% of the participants reported having no knowledge of the MCTS algorithm, while 32.8% indicated they have a basic understanding. The remaining 26.9% of participants are either very familiar with the algorithm or have hands-on experience with MCTS.

4.2 Explanations Quality Assessment

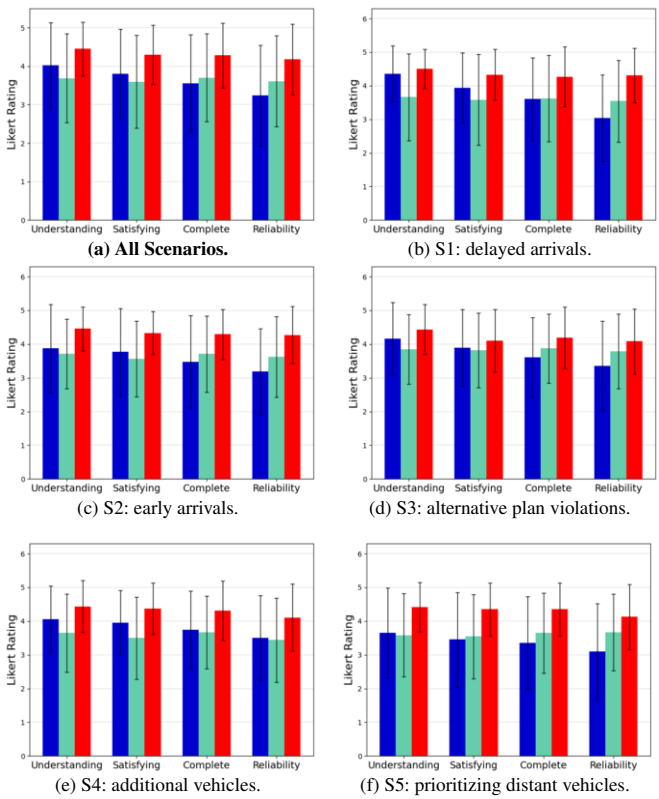
Evaluation Metrics. We designed the questionnaire following the explainable artificial intelligence metrics proposed by Hoffman et al. [20]. Specifically, our questions evaluate the following four *criteria*: **(1) Understandability:** I understand this explanation of the planning algorithm result. **(2) Satisfaction:** This explanation of the planning result is satisfying. **(3) Completeness:** This explanation of the planning result seems complete. **(4) Reliability:** This explanation helps me to assess the reliability of the planning algorithm.

Baselines. Our method is benchmarked against two baselines. The first baseline is designed to represent the information typically displayed in current route dispatching interfaces, while the second focuses on the states, actions, and scores stored within the MCTS search tree. **Baseline (1):** A map visualization that integrates passenger and vehicle locations, with indications of rule violations. **Baseline (2):** A detailed visualization depicting the states, actions, and scores within the search tree.

Assessment on explanation quality by query type. We assess the effectiveness of each query type using the following questions [20]: **(1) Detail:** This explanation of the planning result has sufficient detail. **(2) Irrelevance:** This explanation of the planning result contains irrelevant details. **(3) Accuracy:** This explanation says how accurate the planning algorithm is. The second assessment question focuses on the irrelevance of the information presented in the explanations.

Table 2. Performance of the explanation framework for different query types across five scenarios.

Query Type	All Scenarios			Scenario 1			Scenario 2			Scenario 3			Scenario 4			Scenario 5		
	Det.	Irr.	Acc.	Det.	Irr.	Acc.	Det.	Irr.	Acc.	Det.	Irr.	Acc.	Det.	Irr.	Acc.	Det.	Irr.	Acc.
Factual Queries	4.33	**2.40**	3.96	4.39	**2.39**	3.99	4.29	**2.62**	4.02	4.07	**2.11**	3.43	**4.28**	**2.12**	**4.03**	4.13	**2.52**	3.66
Contrastive Queries	4.28	2.50	3.98	**4.43**	2.42	4.11	**4.40**	2.79	3.96	**4.54**	2.36	**4.26**	4.14	2.41	3.92	4.35	2.75	4.01
Tree Expansions	**4.35**	2.64	**4.08**	4.29	2.65	**4.17**	4.15	2.69	**4.06**	3.95	2.33	3.69	4.11	2.61	3.95	**4.39**	2.74	**4.13**



(a) All Scenarios. (b) S1: delayed arrivals. (c) S2: early arrivals. (d) S3: alternative plan violations. (e) S4: additional vehicles. (f) S5: prioritizing distant vehicles.

Figure 5. Comparative analysis across five scenarios (S1-S5). Left bars represent *baseline 1 with map visualization*; middle bars represent *baseline 2 with search tree visualization*; and right bars represent *the proposed approach*. Plot (a) displays the aggregated results across all scenarios.

Therefore, in our evaluation metrics, for the categories of detail and accuracy, a *higher* score correlates with a better explanation. In contrast, for the category of irrelevance, a *lower* score is more desirable, suggesting that the explanations are concise.

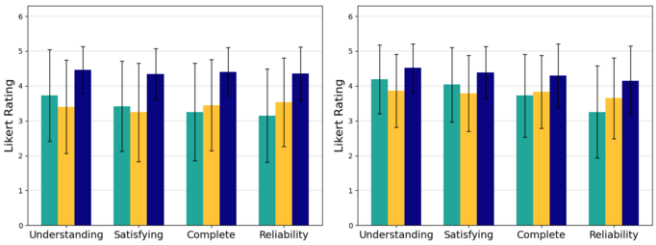
User preferences by technical background. Route dispatchers, despite their knowledge in the paratransit domain, generally lack technical expertise in MCTS. Consequently, to assess the effectiveness of our explanations for both non-technical and technical users, we divided the participant population based on their technical background and familiarity with MCTS. Individuals with no knowledge about the algorithm are considered non-technical users; the rest are technical users. In the left plot of Figure 6, we present the user ratings of three baselines as evaluated by non-technical users, where left bars (green) represent baseline 1, middle bars (yellow) represent baseline 2, and right bars (deep blue) represent our explanations. The right plot of Figure 6 showcases the feedback from technical users.

4.3 Result Analysis

Comparison of CTL-based explainable MCTS with baselines. In Figure 5, we present a comparative evaluation and analysis across all scenarios. Particularly, subplot (a) aggregates and averages the Likert ratings for each criterion across the five scenarios. This result shows that, on average, our explanations outperformed both baselines in all four evaluation criteria. While participants considered baseline 1 with map visualization as more understandable and satisfying than baseline 2 with search tree visualization, the results show that the natural language-based explanations of our approach were superior in terms of understandability, satisfaction, and completeness. Furthermore, these explanations were most effective in helping participants to assess the reliability of the route planning algorithm. Although approximately 60% of the participants have a basic or higher level of understanding of the MCTS algorithm, our proposed approach significantly outperformed the search tree visualization baseline in all four evaluation criteria. This performance improvement can be observed even with a technically informed participant group. Subplots (b)-(f) in Figure 5 showcase the results for each individual scenario. Our proposed method outperforms the other two baseline approaches across all evaluation criteria in all scenarios. Among these criteria, the most significant improvement offered by our method is in aiding participants to evaluate the reliability of the MCTS route planning algorithm. This improvement is likely attributable to the comprehensive level of detail provided by our method, as further shown by the subsequent evaluations focusing on each type of query.

Evaluation of each query type. In Table 2, we present the average Likert ratings for three key evaluation criteria across all five scenarios. Note that in assessing irrelevance, a lower rating is more favorable. Across all five scenarios, the result reveals that contrastive queries are considered as having the most sufficient detail, particularly in explaining why certain requests were not assigned to a vehicle. Moreover, participants rated the factual queries as the most relevant, highlighting their ability to convey essential information without including unnecessary details. About the evaluation of the accuracy of the MCTS algorithm, participants showed a preference for queries that require tree expansion. This preference suggests that they find the information through additional search, beyond what is available in the original search tree, is helpful in further understanding the accuracy of MCTS.

User preferences by technical background. Data in these figures reveal that our explainer received higher ratings across all four evaluation criteria from both user groups. Notably, the advantage of our explainer is more pronounced in the feedback from non-technical users, indicating its effectiveness and adaptability in meeting the needs of users with little to no technical expertise.



(a) Non-technical users. (b) Technical users.

Figure 6. User preferences based on technical background

4.4 Discussion

Learning curve. In Figure 5, we observe that the performance gap between our proposed method and the two baseline methods widens progressively from scenario 1 to scenario 5. This could be attributed to participants experiencing a learning curve in using our system and gaining increased familiarity with the route planning task. Initially, the map visualization might seem adequate for understanding why a particular request is assigned to a vehicle, especially for non-technical users. However, as participants look deeper into the more detailed natural language explanations provided by our explainer, they begin to understand more about the complexity of the scenarios, which is often more complicated than initially perceived. The increasing complexity of the scenarios, especially in the last two tasks where the number of paratransit vehicles grows, likely further emphasizes the superiority of our proposed method over the map visualization. Interestingly, we observe that participants' comprehension of the search tree visualization does not show marked improvement with increased exposure. This insight underscores the need for explanations that can adapt to the complexity of the scenario and the user's growing understanding.

Additional feedback. We included additional open-ended questions at the end of the survey. One participant positively highlighted our proposed system, stating that it "provides me more useful information on how to interpret what the algorithm is doing." This feedback emphasizes the effectiveness of our system in enhancing user comprehension of the algorithm's processes. Several participants highlighted the effectiveness of our explainer in clarifying the routing decisions in scenario 5, particularly why the closer vehicle with ID 1 was not chosen, while a more distant vehicle with ID 3 was assigned instead. For instance, one participant remarked, "I like the solution that was given this time; it clearly explains that vehicle 3 should be taken over vehicle 1." These comments underscore the explainer's ability to provide reasons behind complex decision-making processes in a comprehensible way.

5 Related Work

Explainable AI research has its roots in the need to comprehend, trust, and enhance AI algorithms [5]. Recent studies have highlighted the importance of assisting human users in comprehending and trusting the decisions made by a wide range of models such as regression trees and neural networks (e.g. [16, 18, 4, 30, 3, 27, 41]). Our approach is situated within the broader context of prior research centered on explaining plans and decisions generated by computer algorithms. While there is currently no comprehensive taxonomy for this research domain, we can broadly categorize these efforts as follows.

The first category is related to result-oriented explanations, where the objective of explainability algorithms is to reveal the rationale behind a specific decision post hoc by considering the attributes of that decision. Previous work, such as Langley [24], discusses the desired behaviors of post-hoc explainability modules at a high level, encompassing features like explaining the objectives of the planning task and presenting alternative plans. A comprehensive survey, Sreedharan et al. [39], formally defines distinct primary considerations of explainability systems. These considerations are roughly categorized based on the intended audience (the "explainee") and the nature of the planning results, which can encompass plans or policies. Furthermore, Fox et al. [17] identifies four key types of questions that should be addressed by an explanation system. They illustrate the practical application of the system with two examples, demonstrating how it should function in practice. Nonetheless, the majority of these papers discuss post-hoc explainability systems in a general manner. They often lack specific examples and use cases that showcase explanations in action. Thus, the development of meaningful solutions motivated by real-world problems remains a significant challenge. In contrast, our work is the first to develop an explanation framework for a public transportation-related planning system in the real world, and evaluates the system with human users. The second category aims to generate explanations that are included as a sub-goal of the planning algorithm. For instance, the explainability feature may influence the planning process, where the objective is to also account for system interactions with other system components [40]. Chakraborti et al. [11] views the planning problem as a system with two components: the first component is the AI algorithm, and the second component is the human agent. The goal of the AI algorithm is not to make decisions but to provide suggestions to the human. Similar approaches can also be found in the field of multi-agent reinforcement learning (MARL). Boggess et al. [9] focuses on explaining MARL policies by providing policy summaries in natural language. To ensure comprehensive explanations for user queries, their system conducts additional guided rollouts of the policy to generate informative explanations. Our work shares similar practices in providing explanations for under-explored tree branches with the help of additional computation. However, our approach not only generates more fine-grained explanations leveraging CTL and quantitative evaluations, but also derives explanations from the search tree of an online planner as opposed to a fixed RL policy.

6 Conclusions

We present a CTL-based explainer specifically designed for the MCTS algorithm, focusing on the paratransit route planning application. Our explainer classifies user queries into three categories: factual queries, contrastive queries, and queries requiring path explanations. It then dynamically generates CTL formulas based on the free variables, the type of query and its specifications. After checking different branches of the existing MCTS search tree or even the results of newly initiated searches, the explainer converts the outcomes of its specification verification into natural language explanations with the help of language templates. This approach is designed to enhance the usability of the system for both technical and non-technical audiences. The effectiveness of our method is evidenced by a user study involving 82 participants, which shows notable improvements in user comprehension and satisfaction compared to two baseline methods. We believe that our explainer is versatile enough to be seamlessly adapted to other MCTS implementations in various application fields, broadening its potential for impact.

Acknowledgments

This material is based upon work supported by the National Science Foundation (NSF) under Award Numbers 2028001, 2220401, CNS-2238815 and CNS-1952011, AFOSR under FA9550-23-1-0135, and DARPA under FA8750-23-C-0518. The authors acknowledge the support and guidance of Philip Pugliese from Chattanooga Area Regional Transportation Authority.

References

[1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Information and computation*, 104(1):2–34, 1993.

[2] Z. An, X. Wang, T. T. Johnson, J. Sprinkle, and M. Ma. Runtime monitoring of accidents in driving recordings with multi-type logic in empirical models. In *International Conference on Runtime Verification*, pages 376–388. Springer, 2023.

[3] Z. An, T. Johnson, and M. Ma. Formal logic enabled personalized federated learning through property inference. In *The 38th Annual AAAI Conference on Artificial Intelligence (AAAI 24)*, 2024.

[4] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.

[5] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58: 82–115, 2020.

[6] H. Baier and M. Kaisers. Explainable search. In *2020 IJCAI-PRICAI Workshop on Explainable Artificial Intelligence*, page 178, 2020.

[7] H. Baier and M. Kaisers. Towards explainable mcts. In *2021 AAAI Workshop on Explainable Agency in AI*, page 178, 2021.

[8] J. Biden. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, 2023.

[9] K. Boggess, S. Kraus, and L. Feng. Toward policy explanations for multi-agent reinforcement learning. *arXiv preprint arXiv:2204.12568*, 2022.

[10] P. Bouyer, F. Laroussinie, N. Markey, J. Ouaknine, and J. Worrell. Timed temporal logics. *Models, Algorithms, Logics and Tools: Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, pages 211–230, 2017.

[11] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. *arXiv preprint arXiv:1701.08317*, 2017.

[12] Z. Chen, X. Sun, Y. Li, and M. Ma. Auto311: A confidence-guided automated system for non-emergency calls. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 21967–21975, 2024.

[13] D. Claes, F. Oliehoek, H. Baier, and K. Tuyls. Decentralised online planning for multi-robot warehouse commissioning. In *AAMAS'17: PROCEEDINGS OF THE 16TH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS*, volume 1, pages 492–500. ACM, 2017.

[14] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on logic of programs*, pages 52–71. Springer, 1981.

[15] A. Darwiche and C. Ji. On the computation of necessary and sufficient explanations, 2022.

[16] A. Das and P. Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[17] M. Fox, D. Long, and D. Magazzeni. Explainable planning. *arXiv preprint arXiv:1709.10256*, 2017.

[18] D. Georgiev, P. Barbiero, D. Kazhdan, P. Veličković, and P. Liò. Algorithmic concept-based explainable reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6685–6693, 2022.

[19] T. Gupta, A. Kumar, and P. Paruchuri. Planning and learning for decentralized mdps with event driven rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[20] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

[21] W. Joe and H. C. Lau. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 394–402, 2020.

[22] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[23] S. A. Kripke. Semantical considerations on modal logic. *Acta philosophica fennica*, 16, 1963.

[24] P. Langley. Explainable agency in human-robot interaction. In *AAAI fall symposium series*, 2016.

[25] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[26] M. Ma, S. M. Preum, and J. A. Stankovic. Cityguard: A watchdog for safety-aware conflict detection in smart cities. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pages 259–270, 2017.

[27] M. Ma, S. Preum, M. Ahmed, W. Tärneberg, A. Hendawi, and J. Stankovic. Data sets, modeling, and decision making in smart cities: A survey. *ACM Transactions on Cyber-Physical Systems*, 4(2):1–28, 2019. doi: 10.1145/3355283.

[28] M. Ma, E. Bartocci, E. Lifland, J. Stankovic, and L. Feng. Sastl: Spatial aggregation signal temporal logic for runtime monitoring in smart cities. In *Proc. of ICCPS 2020*, pages 51–62. IEEE, 2020. doi: 10.1109/ICCPS48487.2020.00013.

[29] M. Ma, J. Gao, L. Feng, and J. A. Stankovic. Stlnet: Signal temporal logic enforced multivariate recurrent neural networks. In *NeurIPS 2020*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/a7da6ba0505a41b98bd85907244c4c30-Abstract.html.

[30] M. Ma, J. Stankovic, E. Bartocci, and L. Feng. Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems. *ACM Transactions on Embedded Computing Systems*, 20(5s):1–25, 2021.

[31] M. Ma, J. A. Stankovic, and L. Feng. Toward formal methods for smart cities. *Computer*, 54(9):39–48, 2021.

[32] A. Mukhopadhyay, G. Pettet, C. Samal, A. Dubey, and Y. Vorobeychik. An online decision-theoretic pipeline for responder dispatch. In *Proceedings of the 10th ACM/IEEE international conference on cyber-physical systems*, pages 185–196, 2019.

[33] C. Panigutti, R. Hamon, I. Hupont, D. Fernandez Llorca, D. Fano Yela, H. Junklewitz, S. Scalzo, G. Mazzini, I. Sanchez, J. Soler Garrido, et al. The role of explainable ai in the context of the ai act. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 1139–1150, 2023.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[35] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[36] S. Shah, M. Lowalekar, and P. Varakantham. Neural approximate dynamic programming for on-demand ride-pooling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 507–515, 2020.

[37] A. Shih, A. Choi, and A. Darwiche. A symbolic approach to explaining bayesian network classifiers. *arXiv preprint arXiv:1805.03364*, 2018.

[38] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[39] S. Sreedharan, T. Chakraborti, and S. Kambhampati. The emerging landscape of explainable automated planning & decision making. IJCAI, 2020.

[40] S. Sreedharan, U. Soni, M. Verma, S. Srivastava, and S. Kambhampati. Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with inscrutable representations. *arXiv preprint arXiv:2002.01080*, 2020.

[41] H. D. Wang, N. Khan, A. Chen, N. Sarkar, P. Wisniewski, and M. Ma. Microxercise: A micro-level comparative and explainable system for remote physical therapy. In *2024 IEEE/ACM Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 73–84. IEEE, 2024.

[42] M. Wilbur, S. U. Kadir, Y. Kim, G. Pettet, A. Mukhopadhyay, P. Pugliese, S. Samaranayake, A. Laszka, and A. Dubey. An online approach to solve the dynamic vehicle routing problem with stochastic trip requests for paratransit services. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*, pages 147–158. IEEE, 2022.