INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 0 (2024) 1–34
DOI: 10.1111/itor.13594

Modeling and generating user-centered contrastive explanations for the workforce scheduling and routing problem

Mathieu Lerouge[a,b,*], Céline Gicquel[c], Vincent Mousseau[a] and Wassila Ouerdane[a]

[a] *MICS, CentraleSupélec Université Paris-Saclay, Gif-sur-Yvette 91190, France*
[b] *DecisionBrain, Paris 75010, France*
[c] *LISN, Université Paris-Saclay, Gif-sur-Yvette 91190, France*
E-mail: mathieu.lerouge@centralesupelec.fr[Lerouge]; celine.gicquel@lri.fr[Gicquel]; vincent.mousseau@centralesupelec.fr[Mousseau]; wassila.ouerdane@centralesupelec.fr[Ouerdane]

Received 26 December 2023; received in revised form 9 November 2024; accepted 22 November 2024

Abstract

In the last decade, explainability has been attracting much attention in the machine learning community. However, this research topic extends beyond this field to encompass others such as operations research and combinatorial optimization (CO). This paper addresses this issue in the case of the workforce scheduling and routing problem (WSRP), a CO problem involving human resource allocation and routing decisions. We first introduce a novel mathematical framework that models the process of explaining solutions to the end-users of a WSRP-solving system. Then, we present original algorithmic methods to generate explanation texts employing a high-level vocabulary adapted to such end-users. Explanations are user-centered, local, and contrastive. They are triggered by end-user questions about various topics regarding a solution of a WSRP instance. Both questions and explanations are expressed as texts thanks to templates. Numerical experiments show that the algorithms generating explanation texts have execution times that are mostly compatible with the online use of explanations in an interactive system.

Keywords: combinatorial optimization; artificial intelligence; workforce scheduling and routing problem; user-centered explanations; contrastive explanations

1. Introduction

Nowadays, decision-aid tools based on combinatorial optimization (CO) are used in many professional contexts. For instance, organizations may use them in order to be more efficient at managing their resources and planning their future activities. However, most often, the decision-makers who use these optimization-based tools do not have the necessary background to fully understand their mathematical concepts and algorithmic principles; and even if they do, they may be surprised

*Corresponding author.

by some aspects of the decisions proposed by the systems and thus have some doubts about the relevance of these decisions. In both cases, given the lack of understanding, decision-makers may lose trust in their optimization systems and feel reluctant to apply the decisions proposed by these systems. One way to tackle these issues is to provide explanations to the decision-makers. Such an approach falls under the wide field of eXplainable artificial intelligence (XAI) (Gunning and Aha, 2019).

Over the last decade, explainability has been attracting much attention in the artificial intelligence (AI) community, especially in the machine learning one (Barredo Arrieta et al., 2020). This strong interest has been triggered among others by the XAI research program funded by the Defense Advanced Research Projects Agency (DARPA, 2016) in the United States and by the recent introduction of the General Data Protection Regulations (GDPR, 2016) and Artificial Intelligence Act (AIA, 2021) in Europe. The GDPR gives individuals the right to obtain explanations about how a decision affecting their life and made automatically by an algorithm has been reached. Independently of the AI field they relate to, most works dealing with explanation have similar goals, namely algorithmic transparency, user trust, and bias mitigation (Mohseni et al., 2021). Besides, they often rely on the same fundamental concepts coming from social sciences and philosophy such as the notions of contrastive questions (Lipton, 1990) or counterfactual explanations (Lewis, 1973). Works on explanations from different fields may thus inspire each other. However, the differences between these fields (in terms of problems, use cases, models, inputs, algorithms, etc.) make it difficult to directly transpose an explanation method developed for a given field to another one. For instance, methods generating explanations as saliency heatmaps (a.k.a. sensitivity maps) in deep learning (Simonyan et al., 2014; Zeiler and Fergus, 2014) cannot be easily converted into some equivalent methods that may be used in operations research (OR) contexts and more specifically in CO contexts. Thus, in order to provide explanations to decision-makers using optimization systems, explanation methods specifically designed for CO contexts are needed.

Yet, to the best of our knowledge, there are only a few works dealing with explanations for CO problems (see Ludwig et al., 2018; Čyras et al., 2019; Korikov et al., 2021; Lerouge et al., 2023). The first three papers rely on strong assumptions that limit the applicability of their methods to other CO problems and aim to provide explanations for only a few specific matters of solutions. Focusing on a scheduling problem, Ludwig et al. (2018) seek to provide explanations about the timing of the tasks, that is, explaining why a given task is scheduled at a given time in the solution. However, their explanations tightly depend on the heuristic approach used to solve the problem, making their approach specific to their CO problem and solving heuristic. It also supposes that people receiving their explanations are familiar with the heuristic and agree to solve their instances with this sub-optimal algorithm. Čyras et al. (2019) also consider a scheduling problem but propose an explanation method based on abstract argumentation (Dung, 1995) to explain three matters about solutions: feasibility, local optimality, or satisfaction of a fixed user decision. However, this method seems to be applicable only to a specific class of integer linear programs: programs that can be fully formulated with binary variables which are all involved in clique constraints. Finally, Korikov et al. (2021) focus on another specific class of integer linear programs. Their approach prescribes each explanation to be based on the change of a single input parameter that must be involved in the objective function but not in the constraints, further limiting

the applicability of the approach to other CO problems and the variety of explainable matters. In contrast, Lerouge et al. (2023) may offer a more extendable approach that could be applied to various CO problems beyond the studied scheduling and routing problem. Their approach enables end-users to ask about 15 different questions and receive counterfactual explanations in return. A key strength in favor of the generalization capability of the approach is its relation to neighborhoods, a concept rooted in local search and widely used to solve many CO problems. However, the presented approach lacks of a clear and problem-independent explanation modeling framework to support its potential for generalization. Additionally, while counterfactual explanations are one way to explain results to end-users, other methods, such as contrastive explanations, also exist.

Thus, the development of explanation approaches for CO problems keeps raising several challenging open research issues: How can explanations, in particular contrastive explanations, be defined and modeled in a CO context? How can they be efficiently computed? How can they be effectively communicated to their intended audience? etc. This work is an attempt to addressing these research issues.

Contributions. In this work, we design an original mathematical framework to model the process of giving contrastive explanations about solutions to a CO problem for the end-users of an optimization system. In our approach, end-users trigger explanations by formulating questions based on a template chosen within a predefined bank of questions which satisfy certain assumptions. These questions are interpreted in CO terms and induce mathematical programs. The explanations answering to these questions are then mathematically defined based on the feasibility of the induced programs. Finally, algorithms, which are sought to be executable in a reasonable time (a few seconds), are used for generating explanation texts by relying on template texts. Throughout this work, we apply our method to the same use case as in Lerouge et al. (2023) in which end-users use optimization systems for solving instances of a workforce scheduling and routing problem (WSRP).

Use case. The WSRP involves assigning geographically dispersed tasks to mobile employees and building a route and a schedule for each employee. Figure 1 presents a solution obtained with an optimization system. Each employee is related to a color, for example, the color red for Ellen (El in the figure). The graph on the left represents routes: colored dots correspond to tasks performed by employees and gray ones to nonperformed tasks, while squares correspond to employee starting locations. The Gantt chart on the right depicts schedules: numbered and colored rectangles represent tasks; gray ones represent employee traveling times to go from one task to another; for both groups, the width of a rectangle matches the duration of what it represents.

Focusing on the WSRP has been originally motivated by the needs of our industrial partner DecisionBrain, a French company which develops and sells optimization software products, among which is a decision-aid tool for solving variants of the WSRP. DecisionBrain often receives from its clients questions about the solutions generated by this tool, for example, "Why is Ellen not performing task 27 just after task 17?" after noticing that Ellen passes close to

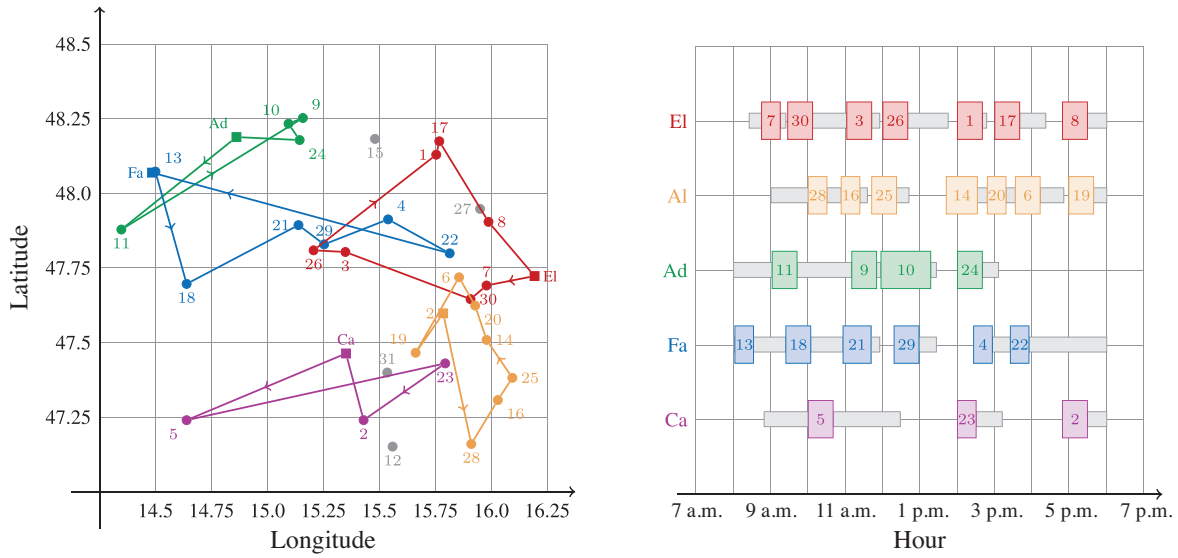

Fig. 1. Representation of routes (left) and schedules (right) of a solution of a WSRP instance.

task 27 between tasks 17 and 8. Answering such questions is time-consuming as it supposes to (1) get familiar with the WSRP instance and solution and (2) find explanatory content to provide to the clients. For these reasons, the automatic generation of explanations in response to clients' questions would be very convenient and useful for both the clients and the company.

More broadly, the WSRP arises in many and various contexts. Lots of applications can be found in the literature: see, for example, Mosquera et al. (2019) for home care services and Chen et al. (2016) for heating, ventilation, and air-conditioning home services. The WSRP is an NP-hard problem as it is a generalization of both vehicle routing and scheduling problems which are NP-hard (Volte et al., 2019). Therefore, decision-makers using an optimization system to solve large instances of the WSRP may feel overwhelmed by the high combinatorial aspect of the problem. Similar to DecisionBrain's clients, they are likely to have difficulties in appreciating why the generated solutions are good ones and, consequently, to feel the need to obtain explanations about these solutions.

Structure of the paper. The remainder of the paper is organized as follows. Section 2 provides an overview of the related literature. It discusses works dealing with explanations in OR, and more broadly in AI, and clarifies where our work stands relative to this literature. Section 3 formally defines the WSRP and provides an integer linear programming (ILP) formulation for this problem. Section 4 presents our mathematical framework modeling the process of explaining solutions to the end-users of WSRP-solving systems. Section 5 describes our algorithmic techniques for computing and narrating explanations, followed by an analysis of the performance of these techniques on large-scale instances and solutions. Finally, conclusions and future works are synthesized in Section 6.

2. Related works

In this section, we first review the literature on explanations in AI by detailing key characteristics of explanation. We then focus on how explanations have been addressed specifically in OR. Finally, we position our work in relation to this literature.

2.1. Design key characteristics of explanations in AI

The concept of explanation in AI may be described according to several key characteristics including target audience, scope, type, trigger, and form. In the following paragraphs, we discuss these characteristics in a cross-disciplinary way, by referring to works from various AI fields, namely expert systems (ES), AI planning (AIP), machine learning, and constraint satisfaction problems (CSP).

Target audience. With the aim of explaining the results or the functioning of an AI system, it is essential to determine the target audience, as the appropriate content and form as well as the intended goals of the explanations that are presented to them may depend on it (Barredo Arrieta et al., 2020; Mohseni et al., 2021). Whatever the considered AI field, the literature generally identifies three target audiences: in ES (Wick and Thompson, 1992), the *end-user* of an expert system, the *domain expert* who is involved in the acquisition of the expert system, and the *knowledge engineer* who designs the expert system; in AIP (Chakraborti et al., 2020), the *end-user* who interacts with the AIP system, the *domain designer*, and the *algorithm designer*—the two latter ones being the AIP twins of the ES domain expert and knowledge engineer; in machine learning (Mohseni et al., 2021), the *AI novice*, the *data expert*, and the *AI expert*. In our work, we assume that the target audience is an end-user of a WSRP-solving system who may not have any expertise in optimization. This audience is analogous to the end-user in ES and AIP or the AI novice in machine learning. In DecisionBrain's context, such an audience corresponds, within their clients' organizations, to the *planners*, that is, people in charge of designing human resources' plannings using the optimization tool as well as the employees affected by the decisions of the planners who may question them. Thus, we seek to adapt the explanations generated by our methods to this audience.

Scope of explanations. Explanations can be classified by their scope, also known as focus (Wick and Thompson, 1992) or interpretation scale (Mohseni et al., 2021). The XAI community generally identifies two scopes of explanations, namely local and global scopes. Originally, in the context of ES, Wick and Thompson (1992) partition explanations into two categories. The first is made of process-related explanations, which involve information on how the system works and generally address "how" questions. The second is made of solution-related explanations, which involve information about the solutions themselves and usually consist of arguments supporting them. In more recent XAI works (Doshi-Velez and Kim, 2017; Guidotti et al., 2018; Liao et al., 2020; Mohseni et al., 2021), the first category is rather referred to as *global* explanations and the second one as *local*

explanations. As noted by Mohseni et al. (2021), local explanations better suit AI novices as they are less overwhelming. Since end-users are the target audience in our work, we design local explanations: our explanations focus on solutions and aim at describing causes justifying some intriguing matters observed in them.

Explanations can always be seen as answers to some questions, even though these questions are not always made explicit. Thus, many works on explanations commonly name explanation types based on characteristics of their corresponding questions or vice versa. Especially, works like Mohseni et al. (2021) name explanation types based on the interrogative forms of their corresponding questions: "how," "why," "why-not," "what-if," "how-to," and "what-else." This categorization is not completely rigorous, since a given explanation type may be suitable for answering questions corresponding to different interrogative forms as noted by Liao et al. (2020). However, it is a convenient and intuitive way to name, compare, and delineate different explanation types. For this reason, we use such a categorization.

Explanation types. As previously mentioned, there are many interrogative forms for the questions to be answered and consequently many explanation types. Among them, two types are widely used in the XAI literature: "why-not" / *contrastive* explanations and "how-to" / *counterfactual* explanations. (See Stepin et al., 2021, for a comprehensive survey of contrastive and counterfactual explanations in XAI.)

- *Contrastive/"why-not" explanations*. Lipton (1990) defines a contrastive question as a question having the following form: "*Why* this observation *rather than* that one" or equivalently "*why not* that other observation *instead of* this one?"—this second version makes the "why-not" name evident. "This observation"/"this one" is called the *fact* and refers to a detail, a property, that can be observed in the result. "That one"/"that other observation" is the *foil* and refers to a hypothetical other aspect that the person who is asking the question would have expected to observe instead. Besides, the foil of the question may sometimes be implicit, especially when the fact is a negation. In this case, the contrastive question becomes simply: "Why is this fact?" As noted by Miller (2019), some authors refer to the foil as the counterfactual case which makes sense as it is literally a counter-fact, that is, an alternative to the fact observed in the output. However, naming the foil this way may be misleading as the term "counterfactual" is also used to name a hypothetical alternative input in counterfactual explanations (see the next bullet point). For this reason, in this work, we adopt the term foil. Explanations answering contrastive questions are usually called contrastive explanations. Miller (2019, 2021) claims that most of the questions asked by people starting with "why" are contrastive, that such questions are usually asked when a surprising or abnormal fact is observed and that contrastive explanations are simpler to deal with for both the questioner and the explainer. Reflecting Miller's position, several works in XAI aim at providing contrastive explanations, whether in AIP (Sreedharan et al., 2018; Cashmore et al., 2019; Lindsay, 2020) or in OR (Čyras et al., 2019; Korikov et al., 2021).
- *Counterfactual/"how-to" explanations*. Consider an input and its corresponding output obtained using an AI system. A counterfactual explanation presents a hypothetical alternative input that

would have resulted in a different output such as a user-specified output (Wachter et al., 2018; Mohseni et al., 2021). An explicit way to ask for such counterfactual explanations is to use questions starting with "how to" (e.g., 'How to obtain this other output?") (Lerouge et al., 2023)—which is the reason why counterfactual explanations are also termed "how-to" explanations. However, counterfactual explanations can also be used for answering contrastive/"why-not" questions of the form "why is this fact and not this foil?" In this case, the counterfactual explanation corresponds to exhibiting a change in the input that would turn the fact mentioned in the question into the foil (Korikov et al., 2021).

This work focuses on providing contrastive explanations.

Trigger of explanations. Many works in XAI concentrate their efforts on modeling, computing, or presenting explanations regardless of any interactions with the target audience and thus do not discuss how explanations are triggered: see, for example, Korikov et al. (2021). Other works take into account interactions with the target audience and specify ways to trigger explanations. For instance, Ludwig et al. (2018) and Čyras et al. (2020) allow the audience to apply *actions* on a graphic user interface (e.g., clicks or drag-and-drops) and assume that these actions implicitly correspond to asking questions about the output. However, such a way to trigger explanations is only possible when there is a limited number of matters that the audience wants to question. In order to allow the end-user to ask a wider range of questions and to obtain explanations on many aspects of the output, it is necessary to consider *questions* explicitly formulated as texts, as done in Swartout and Smoliar (1987), Chandrasekaran et al. (1989), Cashmore et al. (2019) and Lindsay (2020). In our work, we choose to trigger explanations with end-user questions, which allows us to generate explanations on many aspects of the solution.

Form of explanations. There are various ways to present the explanations to the audience (Mohseni et al., 2021). One can resort to *visual* explanations by depicting their explanatory content using visual elements like images, graphs, etc. For instance, in deep learning, some works, for example, Zeiler and Fergus (2014) and Simonyan et al. (2014), use a saliency heatmap to emphasize important features in the input image; in AIP, Krarup et al. (2022) use color to highlight differences between the initial solution and the one computed as part of the explanation. One can also resort to *textual* explanations, which express their explanatory content using words or phrases. A possible approach is then to use template texts and fill them in with computed data (Sqalli and Freuder, 1996; Ludwig et al., 2018). Another approach is to use natural language generation techniques to automate the verbalization of explanations (Forrest et al., 2018; Poli et al., 2021). This paper deals with textual explanations that are expressed based on template texts and does not consider visual explanations.

In the previous paragraphs, we discussed key characteristics of explanation methods in the broad field of AI. In Subsection 2.2, we study how the notion of explanation has been addressed specifically in OR.

2.2. Explanations in OR contexts

We first examine papers dealing with explanations in CSP. We then review works seeking to provide explanations for CO problems, which is the application context of this work.

Explanations in CSP. There are several works on explanations in the field of CSP. In most of them, for example, de Kleer (1986), Ginsberg (1993), Junker (2004), and Cambazard and Jussien (2006), the term explanation (or equivalently no good, removal explanation, conflict set) is used to name a subset of constraints that mathematically justifies either the infeasibility of the CSP instance or, within the solving process, the current state of the variables domains. Some works, for example, Junker (2004), look for the minimal conflict sets; others, for example, de Kleer (1986), Ginsberg (1993), and Cambazard and Jussien (2006), exploit such sets in order to help the solving process. The corresponding explanations are expressed in mathematical terms and may therefore be useful exclusively for the CSP algorithms designers. Actually, few works consider providing explanations to nonexpert end-users; hence, few works consider expressing explanations in a way that is adapted to this audience. Among them, Sqalli and Freuder (1996) and Bogaerts et al. (2021) focus, for instance, on explaining how to solve, step by step, the given CSP instance using verbal or visual elements while Jussien and Ouis (2001) seek to explain infeasibility by naming conflict sets of constraints. In any case, in these works, there is a single matter to explain to the audience: the feasibility/infeasibility of the instance. However, in our context, the end-user is assumed to have at hand a feasible solution and to look for explanations about other matters, for example, "Why is the workforce member Ellen not performing this electricity task while her route goes next to it?" or "Why is Fabian not performing this plumbing task in the morning as his schedule is partially empty?" Consequently, rather than looking for explaining why an instance is feasible, we are interested in explaining why a solution is more relevant than others.

Explanations in CO contexts. Besides these works dealing with explanations in CSP, there are some other OR-related works on user-centered explanations, for example, Ludwig et al. (2018), Čyras et al. (2019), Korikov et al. (2021), and Lerouge et al. (2023). Ludwig et al. (2018) focus on a specific heuristic scheduling system that is based on a greedy algorithm and present a facility that is able to provide to end-users of this system a verbal explanation of the question: "How has this task been scheduled at this time in the returned schedule?" The explanation, which is triggered by clicking on the task of interest on an interface, takes the form of a list of sentences detailing the reasoning steps that have driven the system to schedule the task at a given time. In other words, the explanation consists of describing part of the system execution on an instance, which supposes that end-users understand and agree that the scheduling problem is solved according to the heuristic approach of the system and not another one. In our context, we want the explanations to be independent from the WSRP-solving approach because its functioning is not greedy; therefore, the algorithmic steps may be too technical and overwhelming for end-users and are likely to be updated over time. Čyras et al. (2019) study a minimum makespan scheduling problem. They define a method for explaining why a given schedule is (not) feasible, (not) locally optimal, or (not)

satisfying fixed user decisions, by extracting information from abstract argumentation frameworks (Dung, 1995). However, these argumentation frameworks rely on the implicit assumption that the problem can be formulated as a mathematical program involving exclusively binary variables and that all these variables are involved in clique constraints. This assumption prevents us from applying their method to the WSRP since it is formulated as an integer linear program as it will be presented in Subsection 3.2. Korikov et al. (2021) describe a method based on inverse optimization for producing counterfactual explanations. Each explanation is assumed to be based on the change of a single instance parameter, which must be involved only in the objective function and not in the constraints. This assumption limits the application of their method. Especially, in the case of the WSRP, all the parameters involved in the objective function are also involved in the constraints as it will be presented in Subsection 3.2. Therefore, all these three works have certain limitations making it difficult to apply their methods to our WSRP context and more generally to other OR contexts. In contrast, in a recent work, Lerouge et al. (2023) develop an approach that enables end-users to ask various "how-to" questions about WSRP solutions (e.g., "How to make Ellen perform task 15 in addition to her already-performed tasks?") and receive counterfactual explanations in return (e.g., "By changing the opening time of task 17, from its value 12:30 p.m. in the current input data, to 12:29 p.m. instead, making Ellen perform the task 15 in addition to her already-performed tasks would be possible"). Their approach relies on defining and solving an integer linear program, which simultaneously explores a question-dependent subset of the WSRP solution space and alters the instance parameters within user-fixed ranges, to identify the information needed to answer the question asked by the end-users (e.g., changing the opening time of task 17 from 12:30 p.m. to 12:29 p.m.). However, this integer linear program involves a large number of decision variables used for adjusting the instance parameter alterations, making it potentially difficult to obtain an explanation in a reasonable time (a few seconds). On the other hand, contrastive explanations, which do not require presenting a hypothetical alternate instance, offer an interesting and possibly more efficient alternative to counterfactual explanations. In addition, the presented approach in Lerouge et al. (2023) introduces some key concepts, such as neighboring solutions or closest feasibility, but lacks of formalization to support its potential for generalization. Thus, in this paper, we aim to provide a more rigorous and problem-independent modeling framework for contrastive explanations.

2.3. Our positioning in this work

We conclude this section by positioning our work relatively to the examined literature. First, we identify three main audiences for the explanations in an OR context: the end-users of an optimization system, the business analysts, that is, people in charge of acquiring this system who define its specifications, and the algorithm designers specialized in optimization. Among them, we primarily target the *end-users* of an optimization system solving a WSRP. However, the explanations generated by our method may also be helpful for business analysts and algorithm designers. Then, we deal with explanations that are *local* by focusing on a given solution of a WSRP instance, *contrastive* by answering to question like "why this fact instead of this foil?" and *verbal* by taking the form of *texts* built from templates. Finally, we assume that explanations are *triggered by questions*, also formulated using template texts, which allows us to study various end-user matters about their solutions.

3. Workforce scheduling and routing problem

This section focuses on the WSRP for which we aim at developing an explanation approach. First, we give various definitions related to our WSRP use case. Then, we formulate it as a bi-objective ILP model. It will be exploited later, in Section 4, not for finding optimal solutions, but for defining explanations about the solutions.

3.1. Definitions

The *WSRP* can be briefly stated as follows. Given a set of mobile employees and a set of geographically dispersed tasks, the problem consists of building and assigning to each employee a pair of route and schedule, which defines the tasks they should perform, in what order and at what times over a certain horizon. The objective is to design a family of route–schedule pairs of minimum cost, which accommodates as many tasks as possible while satisfying a set of constraints. For a literature review of the WSRP, see, for example, Castillo-Salazar et al. (2016).

Instance. In our use case, we consider a scheduling horizon of one day, that is, 1440 minutes. Times are expressed in minutes from 12:00 a.m. (e.g., 8:00 a.m. $\equiv$ 480). An instance $\mathcal{I}$ involves a set of n mobile employees $\mathcal{E} = \{1, \ldots, n\}$ and a set of m tasks $\mathcal{T} = \{1, \ldots, m\}$. Each employee $i \in \mathcal{E}$ is characterized by a name, a skill level $s_i \in \mathbb{N}$, a home location, and a working time window $[\underline{w}_i, \overline{w}_i] \subseteq [0, 1440]$. Each task $j \in \mathcal{T}$ is characterized by a required skill level $r_j \in \mathbb{N}$, a location, a performing duration $d_j \in \mathbb{N}$, and an availability time window $[\underline{a}_j, \overline{a}_j] \subseteq [0, 1440]$. In addition, as each employee i departs from their home location at the beginning of their working day and returns to it at the end of the day, we introduce the notations b_i and e_i for referring, respectively, to the departure and return events of i. Observing that tasks, departures, and returns play similar roles, we introduce the notion of activity j to refer either to a task, a departure, or a return. Then, for each employee i, we define a personal set of activities $\mathcal{A}_i = \mathcal{T} \cup \{b_i, e_i\}$. Finally, assuming that all employees travel at the same speed, we note $t_{jk} \in \mathbb{N}$ the travel time needed by any employee to go from activity j to activity k.

Plannings and solution. Given an instance $\mathcal{I}$, a solution is a family $\mathcal{S} = ((\mathcal{R}_i, \mathcal{C}_i))_{i \in \mathcal{E}}$ mapping each employee $i \in \mathcal{E}$ to a planning, that is, a route–schedule pair $(\mathcal{R}_i, \mathcal{C}_i)$. The route $\mathcal{R}_i$ is a sequence of activities of $\mathcal{A}_i$ that begins with b_i and ends with e_i. It defines the tour done by i. The schedule $\mathcal{C}_i$ is a sequence of start times $p_j \in \mathbb{N}$ that defines when i starts performing each activity j in $\mathcal{R}_i$. Let $p \in \mathbb{N}$ be the number of tasks performed by i, then $\mathcal{R}_i = (b_i, j_1, j_2, \ldots, j_p, e_i)$ and $\mathcal{C}_i = (p_{b_i}, p_{j_1}, p_{j_2}, \ldots, p_{j_p}, p_{e_i})$—where we do not indicate that $j_1, j_2, \ldots, j_p$ depend on i for readability purpose.

Throughout this paper, we will often use the solution represented in Fig. 1 as a small illustrative example. Its corresponding instance is detailed in Table B1 of Appendix B, and its routes and schedules are described in Table A1 in Appendix A. For example, the planning $(\mathcal{R}_1, \mathcal{C}_1)$ of employee 1, named Ellen, is given by $\mathcal{R}_1 = (b_1, 7, 30, 3, 26, 1, 17, 8, e_1)$ and $\mathcal{C}_1 = (485, 525, 567, 662, 720, 840, 900, 1009, 1080)$.

Total working and travel times. Various quantitative criteria can be used to compare solutions. In our use case, we consider two main criteria: the total working time and the total travel time. Given a solution $\mathcal{S}$, they are, respectively, noted $f^p(\mathcal{S})$ and $f^t(\mathcal{S})$, and computed as follows:

$$f^p(\mathcal{S}) = \sum_{i \in \mathcal{E}} \sum_{j \in \mathcal{T} \cap \mathcal{R}_i} d_j \qquad \text{and} \qquad f^t(\mathcal{S}) = \sum_{i \in \mathcal{E}} \sum_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} t_{jk}.$$

We consider that maximizing the total working time is more important than minimizing the total traveling time. This allows us to define the following order over the set of solutions. Given two solutions $\mathcal{S}$ and $\mathcal{S}'$, $\mathcal{S}$ is considered better than $\mathcal{S}'$, noted $\mathcal{S} \geq \mathcal{S}'$, if $(f^p(\mathcal{S}), -f^t(\mathcal{S})) \geq (f^p(\mathcal{S}'), -f^t(\mathcal{S}'))$ with a lexicographic order.

Finally, we end this subsection with the notion of *route-equal* plannings (or solutions), which will be used in Section 4 to describe groups of solutions and in Section 5 when computing explanations.

Route-equal plannings and solutions. For a given route $\mathcal{R}_i$, one can build several schedules, which differ from each other by the starting time values. We say that these plannings are route-equal. This notion can then be extended to solutions. Two solutions $\mathcal{S} = ((\mathcal{R}_i, \mathcal{C}_i))_{i \in \mathcal{E}}$ and $\mathcal{S}' = ((\mathcal{R}'_i, \mathcal{C}'_i))_{i \in \mathcal{E}}$ are said to be route-equal if, for each employee i, the plannings $(\mathcal{R}_i, \mathcal{C}_i)$ in $\mathcal{S}$ and $(\mathcal{R}'_i, \mathcal{C}'_i)$ in $\mathcal{S}'$ are route-equal.

For example, consider the solution $\mathcal{S}$ of the illustrative example represented in Fig. 1. Let $(\mathcal{R}'_1, \mathcal{C}'_1)$ be such that $\mathcal{R}'_1 = (b_1,\ 7,\ 30,\ 3,\ 26,\ 1,\ 17,\ 8,\ e_1)$ and $\mathcal{C}'_1 = (541, 581, 620, 715, 768, 873, 925, 1009, 1080)$. Then, $(\mathcal{R}'_1, \mathcal{C}'_1)$ could be another planning for employee 1, Ellen, that is route-equal to her planning $(\mathcal{R}_1, \mathcal{C}_1)$ in $\mathcal{S}$.

3.2. ILP model of our WSRP use case

In this subsection, we provide a formulation of our WSRP use case as a bi-objective ILP model, which is presented in Model 1. We call it the *main model* and note it M. We now detail its decision variables, its objective function, and its constraints.

Decision variables. Two sets of decision variables are used in M. The first set is related to the temporal dimension of the WSRP. For each task $j \in \mathcal{T}$, the integer decision variable T_j defines the time at which j starts to be performed by an employee—if j is performed. The second set of decision variables is related to the spatial dimension of the WSRP. For each employee $i \in \mathcal{E}$ and each pair of activities $(j, k) \in (\mathcal{A}_i \setminus \{e_i\}) \times (\mathcal{A}_i \setminus \{b_i, j\})$, the binary decision variable U_{ijk} is equal to 1 if i performs the activity j and then moves to the activity k, and to 0 otherwise. Note that, using such binary variables, we can build quantities expressing whether task

j is performed by employee i and whether task j is performed by any employee, respectively, as follows:

$$\sum_{\substack{k \in \mathcal{A}_i \\ k \neq b_i, j}} U_{ijk} \quad \text{and} \quad \sum_{i \in \mathcal{E}} \sum_{\substack{k \in \mathcal{A}_i \\ k \neq b_i, j}} U_{ijk}.$$

These quantities are involved several times in the objective function and the constraints of M.

Bi-objective function. The objective function (1) of M is a bi-objective one that is maximized according to a lexicographic order: the first objective equals the total working time, and the second objective is the opposite of the total travel time.

Constraints. The constraints of M are described below by groups.

- *Flow constraints* (2)–(4) ensure that each employee starts their working day from their home location, goes from activities to others without splitting into multiple directions, and ends their day at their home location. Note that these constraints alone do not prevent subtours, that is, loops connecting only tasks, assigned to employees as part of their routes. However, sequencing constraints defined below will prohibit such subtours.
- *Skill constraints* (5) ensure that an employee $i \in \mathcal{E}$ can be assigned to a task $j \in \mathcal{T}$ only if i has a skill level s_i that is higher than r_j the minimum one required for performing j.
- *Occurrence constraints* (6) limit each task $j \in \mathcal{T}$ to be performed no more than once, that is, to occur at most once within all employee routes.
- *Availability constraints* (7) and (8) ensure that, if a task $j \in \mathcal{T}$ is performed, then it must be started and ended within its availability time-window $[\underline{a}_j, \overline{a}_j]$.
- *Working hours and sequencing constraints* (9)–(11) ensure that if an employee $i \in \mathcal{E}$ performs two consecutive activities $j \in \mathcal{A}_i$ and $k \in \mathcal{A}_i \setminus \{j\}$, then i must do so within their working time-window $[\underline{w}_i, \overline{w}_i]$ and i must have enough time to travel from j to k, after ending j and before starting k. The constraints (9)–(11) correspond, respectively, to the foloptimization system b_i and $k \in \mathcal{T}$; $j \in \mathcal{T}$ and $k \in \mathcal{T} \setminus \{j\}$; $j \in \mathcal{T}$ and $k = e_i$.

We end this subsection by discussing the relations between two ways of characterizing solutions.

Two solution characterizations. In Subsection 3.1, we define a solution $\mathcal{S}$ of an instance $\mathcal{I}$ as a *family of plannings* $((\mathcal{R}_i, \mathcal{C}_i))_{i \in \mathcal{E}}$. But, we could also define a solution of $\mathcal{I}$ as the result of M, that is, the *assignment* of a value to each decision variable involved in M. The first characterization rather corresponds to an end-user way of representing a solution, while the second to an optimization system way. In most OR papers, these two characterizations are implicitly equated. However, in our context, it is important to clearly distinguish between them. Namely, as we aim at generating explanations about solutions for end-users of an optimization system, we will have to navigate from the end-user characterization of solutions to the optimization system one and vice versa. Thus, from

now on, we call *ILP solution* an assignment of M decision variables and note it $\mathcal{X}$, while we save the term *solution* for naming a family of employee plannings, noted $\mathcal{S}$.

$$\text{lex max} \quad \Big(\sum_{i\in\mathcal{E}} \sum_{j\in\mathcal{T}} \sum_{k\in\mathcal{A}_i,\, k\neq b_i,j} U_{ijk}\, d_j, \quad -\sum_{i\in\mathcal{E}} \sum_{j\in\mathcal{A}_i,\, j\neq e_i} \sum_{k\in\mathcal{A}_i,\, k\neq b_i,j} U_{ijk}\, t_{jk} \Big), \tag{1}$$

s.t.

$$\sum_{k\in\mathcal{A}_i,\, k\neq b_i} U_{i,b_i,k} = 1 \qquad \forall\, i\in\mathcal{E}, \tag{2}$$

$$\sum_{j\in\mathcal{A}_i,\, j\neq e_i} U_{i,j,e_i} = 1 \qquad \forall\, i\in\mathcal{E}, \tag{3}$$

$$\sum_{j\in\mathcal{A}_i,\, j\neq k,e_i} U_{ijk} = \sum_{j'\in\mathcal{A}_i,\, j'\neq b_i,k} U_{ikj'} \qquad \forall\, i\in\mathcal{E},\ \forall\, k\in\mathcal{T}, \tag{4}$$

$$\sum_{k\in\mathcal{A}_i,\, k\neq b_i,j} U_{ijk} \leq \mathbb{1}_{\{r_j\leq s_i\}} \qquad \forall\, i\in\mathcal{E},\ \forall\, j\in\mathcal{T}, \tag{5}$$

$$\sum_{i\in\mathcal{E}} \sum_{k\in\mathcal{A}_i,\, k\neq b_i,j} U_{ijk} \leq 1 \qquad \forall\, j\in\mathcal{T}, \tag{6}$$

$$\sum_{i\in\mathcal{E}} \sum_{k\in\mathcal{A}_i,\, k\neq b_i,j} U_{ijk}\, \underline{a}_j \leq T_j \qquad \forall\, j\in\mathcal{T}, \tag{7}$$

$$T_j \leq \sum_{i\in\mathcal{E}} \sum_{k\in\mathcal{A}_i,\, k\neq b_i,j} U_{ijk}\, (\overline{a}_j - d_j) \qquad \forall\, j\in\mathcal{T}, \tag{8}$$

$$\sum_{i\in\mathcal{E}} U_{i,b_i,k}(\underline{w}_i + t_{b_ik}) \leq T_k \qquad \forall\, k\in\mathcal{T}, \tag{9}$$

$$T_j + d_j + \sum_{i\in\mathcal{E}} U_{ijk}\, t_{jk} \leq T_k + \Big(1 - \sum_{i\in\mathcal{E}} U_{ijk}\Big)\overline{a}_j \qquad \forall\, (j,k),\in\mathcal{T}^2,\ j\neq k \tag{10}$$

$$T_j + d_j \leq \sum_{i\in\mathcal{E}} U_{i,j,e_i}(\overline{w}_i - t_{je_i}) + \Big(1 - \sum_{i\in\mathcal{E}} U_{i,j,e_i}\Big)\overline{a}_j \qquad \forall\, j\in\mathcal{T}, \tag{11}$$

$$U_{ijk}\in\{0,1\} \qquad \forall\, i\in\mathcal{E},\ \forall\, j\in\mathcal{A}_i\setminus\{e_i\},\ \forall\, k\in\mathcal{A}_i\setminus\{b_i,j\},$$

$$T_j\in\mathbb{N} \qquad \forall\, j\in\mathcal{T},$$

Model 1: Main model M.

Bijection between solution characterizations. Consider instance $\mathcal{I}$ and its corresponding M. Provided that an ILP solution $\mathcal{X}$ satisfies flow constraints (2)–(4) and does not involve any subtour, it can be easily mapped into a solution $\mathcal{S}$. We note φ such a mapping, which is a bijection. In short, given an ILP solution $\mathcal{X}$, one can build a solution $\mathcal{S} = \varphi(\mathcal{X})$ as follows: the routes $(\mathcal{R}_i)_{i\in\mathcal{E}}$ can be deduced from the values of spatial binary variables (U_{ijk}) and the schedules $(\mathcal{C}_i)$ from the values of the temporal variables (T_j). Conversely, by following the inverse reasoning, one can build an ILP solution $\mathcal{X} = \varphi^{-1}(\mathcal{S})$ from a solution $\mathcal{S}$.

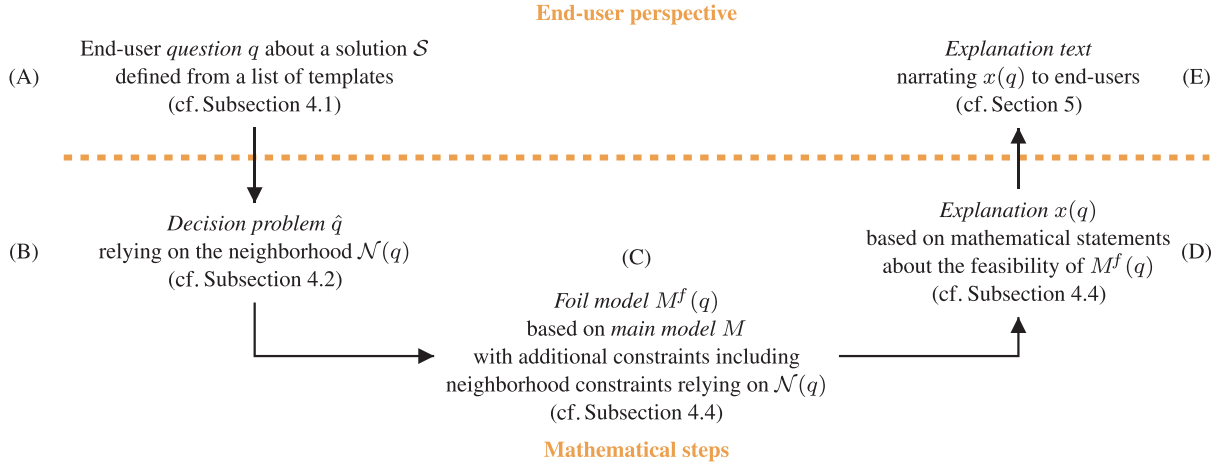

Fig. 2. Overview of the conceptual method developed for explaining a solution $\mathcal{S}$.

We illustrate this mapping with the example of solution $\mathcal{S}$ represented in Fig. 1. By applying φ^{-1} on $\mathcal{S}$, we obtain an ILP solution $\mathcal{X} = \varphi^{-1}(\mathcal{S})$. We give below the part of $\mathcal{X}$ related to employee 1.

- $T_7 = 525,\ T_{30} = 567,\ T_{26} = 720,\ T_1 = 840,\ T_{17} = 900,\ T_8 = 1009$;
- $U_{1,b_1,7} = U_{1,7,30} = U_{1,30,3} = U_{1,3,26} = U_{1,26,1} = U_{1,1,17} = U_{1,17,8} = U_{1,8,e_1} = 1$ and $U_{1jk} = 0$ for all other couples of activities $(j, k) \in (\mathcal{A}_1 \setminus \{e_1\}) \times (\mathcal{A}_1 \setminus \{b_1, j\})$.

Feasible solution and feasible planning. We described above the constraints that an ILP solution must satisfy to be feasible. Using the bijection φ, we can transpose the notion of feasibility from ILP solutions to solutions. We say that a solution $\mathcal{S}$ is feasible if its corresponding ILP solution $\mathcal{X} = \varphi(\mathcal{S})$ is feasible. We also say that a planning $(\mathcal{R}_i, \mathcal{C}_i)$ is feasible if it is part of a feasible solution $\mathcal{S}$.

In this section, we introduced various definitions and notations about our WSRP use case. They will be used, in the following sections, in order to model and generate explanations about solutions.

4. Framework for modeling the explanation process about WSRP solutions

In this section, we describe our framework for modeling the process of explaining WSRP solutions. It consists of the sequence of steps depicted in Fig. 2. As represented in the top part of the figure, from an end-user perspective, our framework involves (A) asking a question q about a solution $\mathcal{S}$ and (E) receiving an explanation in the form of a text. However, as illustrated in the lower part of the figure, multiple consecutive mathematical steps (B)–(D) lead from the initial question q to the ultimate explanation. The following subsections are organized as follows. Subsection 4.1 focuses on step (A). It specifies the nature of the end-user questions handled by our framework and describes how questions can be formulated by end-users. Subsection 4.2 delves into step (B). It presents how

end-user questions formulated in a common language can be interpreted in mathematical terms involving decision problems. Subsection 4.3 deals with step (C). It especially introduces the concept of the foil model, which exploits the main model, that is, the ILP formulation of our WSRP use case introduced in the previous section. Finally, Subsection 4.4 details step (D). It ends this section with a formal definition of explanations. The generation of explanation texts, which correspond to step (E), will be developed in Section 5.

Throughout Section 4, we assume that end-users have $\mathcal{S}$, a feasible solution of an instance $\mathcal{I}$, that they have typically obtained using a WSRP-solving system.

4.1. End-user questions

In this subsection focusing on step (A) in Fig. 2, we specify the nature of the end-user questions that our approach deals with as well as the way end-users can formulate such questions.

Assumptions about the end-user questions as follows:.

- *Scope of explanations*. As mentioned in Subsection 2.1, explanations may have local or global scopes. Our approach deals with *local* explanations, more precisely explanations focusing on a WSRP solution like $\mathcal{S}$.
- *Type of explanations*. Also mentioned in Subsection 2.1, there are various types of explanations (including contrastive and counterfactual explanations) which can be associated with interrogative forms (respectively, "why-not" and "how-to"). In our approach, we aim at providing explanations to *contrastive/"why-not"* questions. Such questions involve a fact and a foil, corresponding, respectively, to a solution feature that end-users observe in $\mathcal{S}$ and one that they would have expected to observe in $\mathcal{S}$ (e.g., the fact that a given task is not performed by a given employee and the foil that the given task may be performed by the given employee). Thus, we do not seek to explain $\mathcal{S}$ but we rather focus on parts of it allowing us to reduce the risks of overloading end-users with information.
- *Neighborhood-related questions*. Explaining the observed facts in $\mathcal{S}$ by contrasting them with expected foils amounts to comparing $\mathcal{S}$ to other solutions that are close to $\mathcal{S}$. In other words, it amounts to comparing $\mathcal{S}$ with neighboring solutions. Therefore, we require that any end-user question about $\mathcal{S}$ be related to a *neighborhood* of $\mathcal{S}$, that is, a subset of solutions that can be obtained from $\mathcal{S}$ by applying a given transformation (e.g., inserting a given task in a given employee planning) as defined in the local search optimization terminology. Thus, the end-user questions can be interpretable as requests for examining neighborhoods of $\mathcal{S}$. Given a question q, we note $\mathcal{N}(q)$ the neighborhood related to q.

To sum up, we design an approach providing explanations in response to neighborhood-related contrastive questions about a solution. Dealing with such questions can be advantageous for both end-users and explanations designers: it helps explanation designers to restrict their examination and propose relevant explanations to end-users in real time.

Although making these assumptions may seem restrictive, the list of end-user questions shows that our approach is able to address a significant number of questions.

Table 1
Nonexhaustive list of question templates.

Label	**Question template text**
(Ins, C)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ just after ⟨activity k^*⟩?"
(Ins, P, a)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ between two consecutive activities of their planning?"
(Ins, P, b)	"Why is ⟨employee i^*⟩ not performing any nonperformed task between two consecutive activities of their planning?"
(Ins, P, c)	"Why is any employee not performing ⟨task j^*⟩ between two consecutive activities of their planning?"
(Ins, E)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ in addition to the activities of their planning (even if it means changing the order of the activities in the planning)?"
(Ex, C)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ rather than ⟨task k^*⟩?"
(Ex, P, a)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ rather than any other task of their planning?"
(Ex, P, b)	"Why is ⟨employee i^*⟩ not performing any nonperformed task rather than ⟨task k^*⟩?"
(Ex, P, c)	"Why is any employee not performing ⟨task j^*⟩ rather than any other task of their planning?"
(Ex, E)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ rather than any other task of their planning (even if it means changing the order of the activities in the planning)?"
(Ord, C, a)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ later in their planning, just after ⟨task k^*⟩?"
(Ord, C, b)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ earlier in their planning, just before ⟨task k^*⟩?"
(Ord, P, a)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ at a later stage in their planning?"
(Ord, P, b)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ at an earlier stage in their planning?"
(Ord, P, c)	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ at any other stage in their planning?"
(Ord, E)	"Why is ⟨employee i^*⟩ not performing the activities of their planning in a different order?"

List of end-user question templates. Table 1 is a nonexhaustive list of end-user question templates that satisfy the above-mentioned assumptions. In order to build this list, we have enumerated various questions that planners, that is, persons in charge of designing employee plannings using the WSRP optimization tool, may have. Each question template text of Table 1 has a label. For example, (Ins, C) template text is "Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ just after ⟨activity k^*⟩?"

- Each template text contains one or several symbols ⟨.⟩, indicating fields that end-users have to specify using data from instance $\mathcal{I}$ (an employee name, a task id, etc.). For example, from the (Ins, C) template, end-users can define the question $q =$ "Why is employee Ellen not performing task 27 just after task 17?" introduced in Section 1. We annotated with a superscript * the indices

of employees, tasks, and activities involved in the fields ⟨.⟩ in order to differentiate them from generic indices.

- Each label is a tuple '(X, Y, Z)' such that
  - 'X' is either 'Ins' (insertion), 'Ex' (exchange), or 'Ord' (reordering). It indicates the kind of transformation to apply to $\mathcal{S}$ in order to obtain the neighboring solutions. For example, for "Why is employee Ellen not performing task 27 just after task 17?" based on the (Ins, C) template, the solutions in $\mathcal{N}(q)$ are obtained from $\mathcal{S}$ by *inserting* task 27 between tasks 17 and 8 in Ellen's route—and by setting new start times in Ellen's schedule.
  - 'Y' is either 'C' (constant), 'P' (polynomial), or 'E' (exponential). Depending on the template on which a question q is based, the number of groups of route-equal solutions in $\mathcal{N}(q)$ may be constant, polynomial (nonconstant), or exponential in n (number of employees) and m (number of tasks). For example, for $q =$ "Why is employee Ellen not performing task 27 between two consecutive tasks of her planning?" based on the (Ins, P, a) template, the solutions in $\mathcal{N}(q)$ are obtained from $\mathcal{S}$ by inserting task 27 between a pair of consecutive activities in Ellen's route—and by setting new start times in Ellen's schedule. Therefore, the number of route-equal solutions in $\mathcal{N}(q)$ is *polynomial* (even linear) in m. In other words, 'Y' relates to the size of $\mathcal{N}(q)$. When 'Y' is 'C' (resp. 'P' or "E"), we say that $\mathcal{N}(q)$ has a *constant* size (resp. *polynomial* or *exponential*).
  - 'Z' is a letter (e.g., 'a', 'b' or 'c') referring to a variant of the transformation whose category is 'X' and neighborhood size is 'Y'; if the pair 'X' and 'Y' defines a unique transformation, then there is no letter for 'Z' and the label is simply '(X, Y)'

Several comments can be made about this list of question templates. First, while the (Ex, P, a) and (Ex, E) templates may appear similar, there is a nuanced difference between them. In (Ex, E), as mentioned in parentheses, the order of the activities in the planning of i^* can be altered if it helps in inserting j^*. Conversely, in (Ex, P, a), the absence of such mention in parentheses implies that the order of activities in the planning of i^* remains unchanged. Then, most of the neighborhoods associated with the templates of Table 1 apply changes to essentially one employee planning while leaving the rest of the current solution unchanged. This will be leveraged for the computation of explanations in Section 5. Finally, it is important to acknowledge that this list is not exhaustive. For example, "Why is ⟨employee i^*⟩ not performing ⟨task j_1^*⟩ and ⟨task j_2^*⟩ instead of ⟨task k^*⟩?" is a question satisfying the above-mentioned assumptions which are not part of Table 1.

In this subsection, we described step (A) of Fig. 2. We specified the nature of the questions addressed by our framework (local, contrastive, and neighborhood-related questions) and presented a list of templates allowing end-users to formulate such questions. In the next subsection, we focus on step (B).

4.2. Decision problems related to end-user questions

From now on, we assume that end-users have a question q about $\mathcal{S}$, based on one of the templates of Table 1. As q is contrastive, it has the following form "Why is *this fact* instead of *that foil*?" (e.g., "Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ just after ⟨activity k^*⟩ [instead of the oppo-

site]?" where the mention in [] is usually implicit). In order to provide an explanation answering q, reasons must be found for why, indeed, *that foil* ("⟨employee i^*⟩ is performing ⟨task j^*⟩ just after ⟨activity k^*⟩") has not occurred in $\mathcal{S}$ instead of *this fact* ("⟨employee i^*⟩ is not performing ⟨task j^*⟩ just after ⟨activity k^*⟩"). We propose to look for such reasons by using optimization-related concepts such as solution feasibility, optimality, and neighborhoods. However, the text of q does not explicitly refer to any optimization concept. Thus, in our framework, the first step towards determining an explanation consists of translating q into a decision problem that is expressed in mathematical terms, related to the field of optimization. This corresponds to step (B) in Fig. 2. We now define such a decision problem.

Decision problem. We call the decision problem related to q, noted $\hat{q}$, the following yes-no question: "Is there a neighboring solution in $\mathcal{N}(q)$ that is feasible and better than $\mathcal{S}$?" $\hat{q}$ can be seen as the translation of q into a decision problem using mathematical terms from the field of optimization. However, in contrast to q, $\hat{q}$ explicitly states what we need to find for answering q: we need to determine whether there exists a neighboring solution $\mathcal{S}'$ that is feasible and better than the original solution $\mathcal{S}$. There are two possible outcomes to $\hat{q}$.

- A *negative outcome* arises when the answer to $\hat{q}$ is "no," that is, when there is no neighboring solution in $\mathcal{N}(q)$ that is feasible and better than $\mathcal{S}$.
- A *positive outcome* arises when the answer to $\hat{q}$ is "yes," that is, when there is at least one neighboring solution $\mathcal{S}'$ in $\mathcal{N}(q)$ that is feasible and better than $\mathcal{S}$.

Note that if $\mathcal{S}$ has been obtained using an optimization system, $\mathcal{S}$ should be a good solution, possibly even an optimal one. Therefore, the negative outcome is more likely to happen than the positive one. Still, since the WSRP is NP-hard, WSRP-solving systems are generally based on approximate optimization methods, which produce good but suboptimal solutions. In this case, it may be possible to improve these solutions by applying transformations (e.g., an insertion) such as the ones implicitly suggested in the end-user questions. Thus, we cannot exclude the possibility of getting a positive outcome.

Even though $\hat{q}$ is expressed in mathematical terms, we still need to figure out how to find the reasons explaining why a negative case occurs. This is the purpose of step (C) in Fig. 2, which consists of rephrasing $\hat{q}$ into another question involving an ILP model. The next subsection focuses on this step.

4.3. Foil models related to end-user questions

Exploiting the fact that the answer to the decision problem $\hat{q}$ is either "yes" or "no," that is, a binary answer, we construct an ILP model, which we refer to as foil model, such that the binary information about its feasibility (whether it is feasible or not) relates to the answer to $\hat{q}$. This corresponds to step (C) in Fig. 2. We recall that we introduced in Subsection 3.2 a bi-objective ILP model of our WSRP use case that we called the main model and noted M.

Foil model. The foil model related to q, noted $M^f(q)$, is defined as the ILP model obtained by extending the main model M (cf. Model 1) so that the answer to $\hat{q}$ matches the one to "Is $M^f(q)$ feasible?" As shown in Model 2, $M^f(q)$ is obtained from M by keeping the same decision variables, bi-objective function, and constraints but adding new constraints defined as follows.

- *Neighborhood constraints* (12) are considered in $M^f(q)$ in order to ensure that its solutions are neighboring ones, that is, solutions satisfying the foil of q—hence, the name of this model. Expressing these constraints in terms of decision variables (T_j) and (U_{ijk}) depends on the structure of $\mathcal{N}(q)$, that is why we simply write them in short $\varphi(\mathcal{X}) \in \mathcal{N}(q)$.
- *Improvement constraints* (13) are considered in $M^f(q)$ to force its solutions to be better than $\mathcal{S}$.

Model 2: Foil model $M^f(q)$ induced by a question q.

Let us illustrate neighborhood constraints as an example. Consider again $\mathcal{S}$ represented in Fig. 1, a feasible solution of an instance $\mathcal{I}$, as well as the question q "Why is Ellen not performing task 27 just after task 17?" Then, neighborhood constraints in $M^f(q)$ are the following:

- For $i \in \mathcal{E} \setminus \{1\}$, variables (U_{ijk}), as well as (T_j) such that j is in $\mathcal{R}_i$, are set to their value in $\varphi^{-1}(\mathcal{S})$.
- All the variables (U_{1jk}) are set to their value in $\varphi^{-1}(\mathcal{S})$ except for $U_{1,17,8}$ which is set to 0 and for $U_{1,17,27}$ and $U_{1,27,8}$ which are set to 1.

With $M^f(q)$, it is clearer where to find reasons when a negative outcome arises: we should look for an infeasible subset of constraints making the foil model infeasible. Thus, we can now move on to step (D) and define the notion of explanation in our context.

4.4. Explanations

In line with the potential positive or negative outcomes of the decision problem, we define two cases of explanations: negative and positive. To get some intuitions about these categories, consider that end-users ask a question q about $\mathcal{S}$. A positive explanation ensues when there exists a neighboring solution $\mathcal{S}'$ in $\mathcal{N}(q)$ that is feasible and better than $\mathcal{S}$. Therefore, a positive explanation essentially confirms that end-users were right to wonder about the foil of q. Conversely, a negative explanation arises when none of the neighboring solutions is feasible and better than $\mathcal{S}$. In essence, a negative explanation basically confirms that $\mathcal{S}$ is of good quality and aims at justifying that. Furthermore, we delineate two subcases of negative explanations: proof-like and argument-like. The first aims at providing a comprehensive justification by relying on an infeasible subset of constraints (i.e., a subset of constraints such that the set of solutions satisfying these constraints is empty): it aims at providing a proof. The second seeks to provide a convincing example to illustrate the negative outcome, not a mathematical proof of it: it seeks to provide an argument. Subsequently, we provide the definitions for each of these cases.

Proof-like negative explanation. When $M^f(q)$ is infeasible, a proof-like negative explanation $x(q)$ can be defined as follows:

$$\mathcal{U} \text{ is an infeasible subset of constraints of } M^f(q) \textit{ therefore} \text{not}(\exists\, \mathcal{S}' \in \mathcal{N}(q) \text{ feasible},\ \mathcal{S}' \geq \mathcal{S}).$$

Note that, in this definition, the infeasible subset of constraints $\mathcal{U}$ is not required to be inclusion-wise minimal. Actually, since we ultimately aim at narrating $x(q)$, we are rather interested in finding a subset that we can describe in a few sentences rather than one that is inclusion-wise minimal. Besides, in order to find such $\mathcal{U}$, we will not directly solve and assess the feasibility of the foil model, but rather resort to a polynomial algorithm, when possible. These considerations about computing and narrating infeasible subsets of constraints will be discussed in Section 5.

Argument-like negative explanation. When $M^f(q)$ is infeasible, an argument-like negative explanation $x(q)$ can be defined as follows:

$$\text{not}(\exists\, \mathcal{S}' \in \mathcal{N}(q) \text{ feasible},\ \mathcal{S}' \geq \mathcal{S}). \textit{ Forexample}, \mathcal{S}' \in \mathcal{N}(q) \text{ but not(feasible and } \mathcal{S}' \geq \mathcal{S}).$$

Note that, in this definition, the neighboring solution $\mathcal{S}'$ that is used as an example is not required to satisfy some kind of optimality. However, in order to be convincing, we need to find a striking solution example for $\mathcal{S}'$. In Section 5, we will discuss how we choose and compute it.

Positive explanation. When $M^f(q)$ is feasible, a positive explanation $x(q)$ is defined as follows:

$$\begin{aligned} &\textit{because} \mathcal{S} \text{ was not an optimal solution, the foil of } q \text{ was not observed in } \mathcal{S} \\ &\quad \textit{however} \text{ this foil can be observed in } \mathcal{S}' \text{ with } \mathcal{S}' \text{ feasible and } \mathcal{S}' \geq \mathcal{S}, \end{aligned}$$

where $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{X}$ feasible ILP solution of $M^f(q)$.

In this section, we described steps from (A) to (D) in the explanation process depicted in Fig. 2. These lead from an end-user question q to an explanation $x(q)$. A proof-like negative explanation is based on an infeasible subset of constraints, while an argument-like negative explanation or a positive explanation relies on a specific neighboring solution. Two notable remarks emerge. First, explanations are essentially formulated in mathematical terms, as it is particularly evident in the case of proof-like negative explanations that rely on infeasible subsets of constraints—a concept lacking practical meaning for nonexperts. Therefore, these explanations are not suitable information to communicate to end-users. Second, we have not yet addressed how to compute the information involved in the explanations, namely the infeasible subsets or the specific neighboring solutions. Thus, in Section 5, we will describe how explanations can be both computed and narrated into intelligible texts for end-users.

5. Generating explanation texts

In the previous section, we gave the definition of explanations in mathematical terms. As represented by step (E) of Fig. 2, given an end-user question q about a solution $\mathcal{S}$, our ultimate goal

is to provide an explanation in the form of intelligible text. Besides, we aim at computing such texts within a time frame that is compatible with an online use of explanations in an interactive system. In this section, we describe our method for generating explanation texts and conducting numerical experiments on large-scale instances and solutions to measure its performances in terms of computation time.

Algorithmic structure. The generation of an explanation text involves three phases.

- *Phase 1*. Preliminary checks are carried out. They verify, in polynomial time and without the need to construct any neighboring solution, whether necessary conditions, for $\mathcal{N}(q)$ to contain feasible solutions that are better than $\mathcal{S}$, are met. For example, whenever the transformation from $\mathcal{S}$ to any neighboring solution of $\mathcal{N}(q)$ requires one task j^* to be newly assigned to one employee i^*, we carry out two preliminary checks: (a) i^* must be skilled enough to perform j^*; (b) if j^* was the unique task assigned to i^*, i^* must be able to feasibly perform it. If these checks are successful, the algorithm proceeds to Phase 2 then Phase 3; otherwise, it proceeds directly to Phase 3.
- *Phase 2*. This phase explores $\mathcal{N}(q)$ to search for a neighboring solution that is both feasible and better than $\mathcal{S}$ or to identify any conflicts preventing the existence of such a solution.
- *Phase 3*. Based on the results of the two preceding phases, an explanation text is built, which corresponds either to a proof-like negative, an argument-like negative, or a positive explanation.

For the sake of brevity, we will not provide additional details about the preliminary checks of Phase 1. The coming Subsections 5.2 and 5.3 will focus on Phases 2 and 3. However, before delving into these phases, we dedicate the following Subsection 5.1 to the definition of the *support solution*. Intuitively, this solution is the "best" (or "most convincing") solution in $\mathcal{N}(q)$, upon which a relevant explanation text can be constructed by exploiting its feasibility/infeasibility and its objective values. This support solution will play a key role in Phases 2 and 3.

5.1. Definition of the support solution

To begin with this subsection, we first provide preliminary comments and introduce several concepts related to solution transformations as prerequisites for defining the support solution.

Consistent task insertion in solution transformations. Let q be a question about a solution $\mathcal{S}$ based on a template '(X, Y, Z)'. When 'X' is 'Ins,' building any solution $\mathcal{S}'$ in $\mathcal{N}(q)$ from $\mathcal{S}$ requires, exactly once, to insert a task in an employee planning (while either maintaining or permuting the order of tasks within this planning along with the insertion). When 'X' is 'Ex' or 'Ord,' $\mathcal{S}'$ in $\mathcal{N}(q)$ is obtained from $\mathcal{S}$ by removing a task from an employee planning and inserting another task or the same one in the same planning. Thus, regardless of the template, the operations to build any $\mathcal{S}'$ in $\mathcal{N}(q)$ from $\mathcal{S}$, consistently involve exactly one task insertion in a planning. In other words,

each subset of route-equal solutions of $\mathcal{N}(q)$ is associated with a task insertion in a planning. Two solutions in $\mathcal{N}(q)$ may not be obtained by inserting the same task in the same planning; however, each of them is obtained from $\mathcal{S}$ by inserting, exactly once, a task in a planning as part of the transformation operations.

By definition of our WSRP use case (see Subsections 3.1 and 3.2), removing a task from an employee planning is always feasible. Therefore, regardless of the template of q, the critical operation involved in the transformation from $\mathcal{S}$ to $\mathcal{S}'$ in $\mathcal{N}(q)$, is this consistent task insertion. In the subsequent paragraphs, we introduce the concepts of backward-feasible earliest start time, forward-feasible latest start time, and feasibility gap which we use for evaluating the feasibility/infeasibility of a task insertion in a planning.

Backward-feasible earliest start time and forward-feasible latest start time. Consider a feasible planning $(\mathcal{R}_i, \mathcal{C}_i)$, two consecutive activities (k_1, k_2) in $\mathcal{R}_i$, and a task j not in $\mathcal{R}_i$. We define the notions of Backward-feasible Earliest start Time (BET) noted $\underline{p}_j$ (resp. Forward-feasible Latest start Time (FLT) noted $\overline{p}_j$) of j as follows. Suppose one wants to insert j in $\mathcal{R}_i$ between k_1 and k_2. $\underline{p}_j$ is the earliest time (resp. $\overline{p}_j$ is the latest time) at which i can start performing j such that the activities before k_1 (resp. after k_2) can be associated with start times satisfying time constraints (i.e., availability, working hours, and sequencing constraints).

Feasibility gap. Inserting a task j between a pair of consecutive activities (k_1, k_2) of $\mathcal{R}_i$ is feasible if and only if $\underline{p}_j \leq \overline{p}_j$, that is, if and only if $\max(\underline{p}_j - \overline{p}_j, 0) = 0$. We call feasibility gap the term $\max(\underline{p}_j - \overline{p}_j, 0)$. This term measures the infeasibility of inserting j.

As mentioned earlier, the task insertion in a planning is both a consistent operation and the critical one involved in the transformation from $\mathcal{S}$ to any $\mathcal{S}'$ in $\mathcal{N}(q)$. Therefore, evaluating the feasibility of the transformation amounts to evaluating the feasibility of its inherent task insertion, and the feasibility gap related to this insertion can be used as a quantity evaluating the (in)feasibility of the whole transformation. This enables us to define the notions of the nearest-to-feasibility solution and support solution.

Nearest-to-feasibility solution. For every neighboring solution $\mathcal{S}'$ in $\mathcal{N}(q)$, building $\mathcal{S}'$ from $\mathcal{S}$ requires to apply an insertion whose feasibility can be evaluated using the feasibility gap. Then, we say that a neighboring solution in $\mathcal{N}(q)$ is a nearest-to-feasibility solution (relatively to $\mathcal{I}$) if it minimizes the feasibility gap.

Support solution. We define a support solution, noted $\mathcal{S}^\star$, as a solution in $\mathcal{N}(q)$, such that

- if $\mathcal{N}(q)$ does not contain any feasible solutions, then $\mathcal{S}^\star$ is a nearest-to-feasibility solution in $\mathcal{N}(q)$;
- else, $\mathcal{S}^\star$ is the best feasible solution in $\mathcal{N}(q)$ (in terms of bi-objective function values).

Note that there may be several solutions minimizing the feasibility gap, and therefore several nearest-to-feasibility solutions and several support solutions. That is why in the defini-

tions above we wrote *a* nearest-to-feasibility solution (resp. *a* support solution) rather than *the* nearest-to-feasibility solution (resp. *the* support solution). However, in this document, we sometimes abuse language and write *the* instead of a as in such cases these solutions are considered equivalent.

In the next subsection, we present our methodology for computing a support solution.

5.2. Phase 2: computing a support solution

The algorithm computing a support solution $\mathcal{S}^\star$ in relation to q must explore the neighborhood $\mathcal{N}(q)$. The size of $\mathcal{N}(q)$ has a direct impact on the nature of such an algorithm. When $\mathcal{N}(q)$ has a constant or polynomial size, we propose a *polynomial-time* algorithm. It explores $\mathcal{N}(q)$ by examining each of its subsets of route-equal solutions, given that there is a polynomial number of such subsets. However, when $\mathcal{N}(q)$ has an exponential size, examining an exponential number of such subsets, one by one, would be computationally intractable. Therefore, we resort to an *ILP-based* algorithm which implicitly explores $\mathcal{N}(q)$ by solving an ILP model. We discuss in what follows both kinds of algorithms.

Polynomial-time algorithms for computing support solutions. Let q be a question based on a 'Y'='C' or 'Y'='P' template (i.e., '(X, Y, Z)' template with 'Y' being 'C' or 'P'). The neighborhood $\mathcal{N}(q)$ has a constant or polynomial size. Then, we design a polynomial-time algorithm for computing a support solution $\mathcal{S}^\star$ in relation to q. This algorithm consists essentially of evaluating the (in)feasibility and the quality of solutions in each subset of $\mathcal{N}(q)$ containing route-equal solutions in order to identify the "best" neighboring solution. More precisely, it consists of computing the feasibility gap of the task insertion as well as a neighboring solution corresponding to each of these subsets (as explained previously in Subsection 5.1), and in selecting the best one: if a feasible task insertion is found and if the resulting feasible neighboring solution is the best found so far (i.e., with the best bi-objective values), then the algorithm saves the solution as support solution $\mathcal{S}^\star$; if none of the task insertions is feasible, then the algorithm saves as $\mathcal{S}^\star$ the neighboring solution whose cor-

responding task insertion has the smallest feasibility gap. This whole algorithmic procedure runs in polynomial time.

$$\text{lex}\ \min\ \Big(\underline{T}_{j^*} - \overline{T}_{j^*},\quad -\sum_{j\in\mathcal{T}} \sum_{\substack{k\in\mathcal{A}_i \\ k\neq b_i, j}} U_{ijk}\, d_j,\quad \sum_{\substack{j\in\mathcal{A}_{i^*} \\ j\neq e_{i^*}}} \sum_{k\in\mathcal{A}_{i^*},\, k\neq b_{i^*}, j} U_{i^*jk}\, t_{jk} \Big), \tag{1'}$$

s.t.

$$\sum_{k\in\mathcal{A}_{i^*},\, k\neq b_{i^*}} U_{i^*b_{i^*}k} = 1, \tag{2'}$$

$$\sum_{j\in\mathcal{A}_{i^*},\, j\neq e_{i^*}} U_{i^*je_{i^*}} = 1, \tag{3'}$$

$$\sum_{j\in\mathcal{A}_{i^*},\, j\neq k, e_{i^*}} U_{i^*jk} = \sum_{j'\in\mathcal{A}_{i^*},\, j'\neq b_{i^*}, k} U_{i^*kj'} \qquad \forall\, k\in\mathcal{T}^\star, \tag{4'}$$

$$\sum_{k\in\mathcal{A}_{i^*},\, k\neq b_{i^*}, j} U_{i^*jk} \leq 1 \qquad \forall\, j\in\mathcal{T}^\star, \tag{6'}$$

$$\underline{a}_j \leq T_j \qquad \forall\, j\in\mathcal{T}^\star\setminus\{j^*\}, \tag{7.a'}$$

$$\underline{a}_{j^*} \leq \underline{T}_{j^*}, \tag{7.b'}$$

$$T_j \leq \overline{a}_j - d_j \qquad \forall\, j\in\mathcal{T}^\star\setminus\{j^*\}, \tag{8.a'}$$

$$\overline{T}_{j^*} \leq \overline{a}_{j^*} - d_{j^*}, \tag{8.b'}$$

$$\underline{w}_{i^*} + t_{b_{i^*}k} \leq T_k \qquad \forall\, k\in\mathcal{T}^\star\setminus\{j^*\}, \tag{9.a'}$$

$$\underline{w}_{i^*} + t_{b_{i^*}j^*} \leq \underline{T}_{j^*}, \tag{9.b'}$$

$$T_j + d_j + U_{i^*jk}\, t_{jk} \leq T_k + \Big(1 - U_{i^*jk}\Big)\overline{a}_j \qquad \forall\, j\neq k\in\mathcal{T}^\star\setminus\{j^*\}, \tag{10.a'}$$

$$\overline{T}_{j^*} + d_{j^*} + U_{i^*j^*k}\, t_{j^*k} \leq T_k + \Big(1 - U_{i^*j^*k}\Big)\overline{a}_{j^*} \qquad \forall\, k\in\mathcal{T}^\star\setminus\{j^*\}, \tag{10.b'}$$

$$T_j + d_j + U_{i^*jj^*}\, t_{jj^*} \leq \underline{T}_{j^*} + \Big(1 - U_{i^*jj^*}\Big)\overline{a}_j \qquad \forall\, j\in\mathcal{T}^\star\setminus\{j^*\}, \tag{10.c'}$$

$$T_j + d_j \leq U_{i^*je_{i^*}}(\overline{w}_{i^*} - t_{je_{i^*}}) + \Big(1 - U_{i^*je_{i^*}}\Big)\overline{a}_j \qquad \forall\, j\in\mathcal{T}^\star\setminus\{j^*\}, \tag{11.a'}$$

$$\overline{T}_{j^*} + d_{j^*} \leq U_{i^*j^*e_{i^*}}(\overline{w}_{i^*} - t_{j^*e_{i^*}}) + \Big(1 - U_{i^*j^*e_{i^*}}\Big)\overline{a}_{j^*}, \tag{11.b'}$$

$$\underline{T}_{j^*} - \overline{T}_{j^*} \geq 0, \tag{12.a'}$$

$$\overline{T}_{j^*} \leq T_{j^*} \leq \underline{T}_{j^*}, \tag{12.b'}$$

$$\varphi(\mathcal{X}) \in \mathcal{N}(q), \tag{13'}$$

$$U_{i^*jk} \in \{0,1\} \qquad \forall\, j\in\mathcal{A}_{i^*}\setminus\{e_{i^*}\},\ \forall\, k\in\mathcal{A}_{i^*}\setminus\{b_{i^*}, j\},$$

$$T_j \in \mathbb{N} \qquad \forall\, j\in\mathcal{T}^\star,$$

$$\underline{T}_{j^*},\ \overline{T}_{j^*} \in \mathbb{N}.$$

Model 3: Transformation model $M^t(q)$ associated with a question q.

ILP-based algorithms for computing support solutions. Let q be a question based on a 'Y'='E' template (i.e., '(X, Y, Z)' template with 'Y' being 'E'). The neighborhood $\mathcal{N}(q)$ has an exponential size. Then, we design an ILP-based algorithm for computing a support solution $\mathcal{S}^\star$ in relation to q. It consists of solving Model 3, an ILP model whose results can be used to build $\mathcal{S}^\star$ from $\mathcal{S}$. We call it the *transformation model* and note it $M^t(q)$. It is based on the main model M; however, it differs from M on various aspects which we detail below. Before that, note that each of (Ins, E), (Ex, E), and (Ord, E) templates specifies an employee of interest i^* within their text. In addition, each of (Ins, E) and (Ex, E) also specifies a task of interest j^*, which corresponds to the task that must be inserted in the planning of i^*. In the case of (Ord, E), we decide to note j^* the task positioned in the middle of the planning of employee i^*.

- *General differences.* $M^t(q)$ aims at finding the support planning $(\mathcal{R}^\star_{i^*}, \mathcal{C}^\star_{i^*})$, which must replace $(\mathcal{R}_{i^*}, \mathcal{C}_{i^*})$ in the solution $\mathcal{S}$ in order to obtain the support solution $\mathcal{S}^\star$. Therefore, $M^t(q)$ focuses on optimizing only one planning, namely the one of i^*, whereas M deals with a whole solution. Let $\mathcal{T}^\star$ be a subset of $\mathcal{T}$ containing only the tasks that are involved in the planning of employee i^* as well as possible other tasks that are relevant for the transformation to apply to $\mathcal{S}$ (e.g., the task to insert in the planning of i^*). Focusing the optimization on only one planning has various consequences:
  - Not all the decision variables of M are involved in $M^t(q)$: only the path decision variables U_{ijk} with $i = i^*$ and the temporal ones T_j with $j \in \mathcal{T}^\star$ are indeed necessary for $M^t(q)$. In other words, moving from M to $M^t(q)$ can be seen as fixing a lot of decisions variables involved in M and focusing the optimization over the set of the remaining variables.
  - There are no sums over $\mathcal{E}$ or constraints repeated over $\mathcal{E}$ in $M^t(q)$; sums indexed over $\mathcal{T}$ in M are indexed over $\mathcal{T}^\star$ in $M^t(q)$; constraints repeated over $\mathcal{T}$ in M are repeated over $\mathcal{T}^\star$ in $M^t(q)$.
  - The size of $M^t(q)$ decreases drastically compared to M, facilitating the faster resolution of $M^t(q)$.
- *Decision variables*. In addition to the path decision variables and the temporal ones inherited from M, two new integer variables $\underline{T}_{j^*}$ and $\overline{T}_{j^*}$ are introduced in $M^t(q)$. Like T_{j^*}, they are related to the time at which the task j^* starts to be performed by i^*: $\underline{T}_{j^*}$ (resp. $\overline{T}_{j^*}$) corresponds to the time at which i^* can start to perform j^* while having all the time constraints related to the activities performed by i^* before (resp. after) j^* satisfied and while respecting the lower (resp. upper) bound of the availability window of j^*. We call $\underline{T}_{j^*}$ and $\overline{T}_{j^*}$, *BET* and *FLT decision variables*, as they are the decision-variable equivalents of the BET $\underline{p}_j$ and FLT $\overline{p}_j$. Due to the way they are involved in $M^t(q)$, $\underline{T}_{j^*}$ and $\overline{T}_{j^*}$ guarantee the feasibility of $M^t(q)$: either $\underline{T}_{j^*} > \overline{T}_{j^*}$ and it means that inserting j^* in the planning of i^* is infeasible; or $\underline{T}_{j^*} = \overline{T}_{j^*}$ and it means that the insertion is feasible since all the time constraints of the activities performed by i^* before and after j^* are satisfied.
- *Multiobjective function*. Similarly to M, a multiobjective function (1') is minimized according to a lexicographic order in $M^t(q)$. However, the objectives are not the same as in M.
  1. The first objective aims at minimizing the difference $\underline{T}_{j^*} - \overline{T}_{j^*}$, since having $\underline{T}_{j^*} = \overline{T}_{j^*}$ means that the insertion of j^* in the planning of i^* is feasible. Note that constraint (12.a') prevents the difference $\underline{T}_{j^*} - \overline{T}_{j^*}$ from being negative.
  2. The second objective, maximizing the planning total working time, relates to the first of M.

3. The third objective, minimizing the planning total traveling time, relates to the second of M.

- *Constraints*. Most constraints of M still apply in $M^t(q)$ but must be adapted.
  - Flow constraints (2)–(4) correspond to (2') to (4'); however, (2')–(4') zoom on employee i^* and subset $\mathcal{T}^\star$ while (2)–(4) involve the whole sets $\mathcal{E}$ and $\mathcal{T}$.
  - No equivalent for the skill constraint (5) appears in the $M^t(q)$ as i^* is supposed to have a higher skill level than the ones of all the tasks of $\mathcal{T}^\star$ (as preliminary checks are met).
  - Occurrence constraint (6) coincides with (6') but (6') zooms on employee i^* and subset $\mathcal{T}^\star$.
  - Availability, working hours, and sequencing constraints spanning labels from (7) to (11) in M are also involved in $M^t(q)$, with some changes, and correspond to constraints spanning from (7.a') to (11.b'). Each original constraint of M (e.g., constraint (7)) is split into two or three constraints in $M^t(q)$ (e.g., constraints (7.a') and (7.b')) in order to separate the case of j^*, which involves $\underline{T}_{j^*}$ and $\overline{T}_{j^*}$, from the one of any other task j in $\mathcal{T}^\star$, which involves T_j.

  In addition, three new constraints are introduced in $M^t(q)$.
  - As mentioned earlier, (12.a') prevents the difference $\underline{T}_{j^*} - \overline{T}_{j^*}$ from being negative.
  - (12.b') is simply used for controlling T_{j^*} value which is no longer involved in any constraint.
  - (13') corresponds to the neighboring constraints. First introduced for the foil model $M^f(q)$ in Subsection 4.3, they are written implicitly as "$\varphi(\mathcal{X}) \in \mathcal{N}(q)$" since they depend on q. However, in practice, they are constraints involving path variables and enforcing task assignments, insertions, replacements, etc.

Now that we described our methodology for computing a support solution $\mathcal{S}^\star$, we can outline in the following subsection how we then build an explanation text answering a question q.

5.3. Phase 3: building an explanation text

Based on the results of Phases 1 and 2, we build an explanation text answering question q. We comment below on how we proceed depending on whether the preliminary checks are successful.

Unsuccessful preliminary checks. If one of the preliminary checks is unsuccessful, we build a proof-like explanation text from predefined explanation template text. Such template text must be prepared for each preliminary check in case they appear to be unsuccessful.

For example, let q be the (Ins,E) question: "Why is Carlotta not performing task 12 in addition to the tasks of her planning?" The preliminary check related to skill constraints is not met: Carlotta has a skill level of 1 while task 12 requires one of 2. The prepared explanation template text is "⟨*The unexpected fact is observed*⟩ because ⟨employee i^*⟩ has a skill level of $\langle s_{i^*} \rangle$ while ⟨task j^*⟩ has a skill level of $\langle r_{j^*} \rangle$." Once filled with values, this becomes a proof-like negative explanation text: "*Carlotta is not performing task 12 because Carlotta has a skill level of 1 while task 12 has a skill level of 2*."

Neighborhood exploration. If none of the preliminary checks is unsuccessful, then a support solution $\mathcal{S}^\star$ is computed in relation to the question q. There are three different cases.

1. Either $\mathcal{S}^\star$ is infeasible due to time constraints, and we must build a negative explanation text about the infeasibility of the neighboring solutions of $\mathcal{N}(q)$.
2. Either $\mathcal{S}^\star$ is feasible but not better than $\mathcal{S}$, and we must build a negative explanation text about the nonimprovement of $\mathcal{N}(q)$.
3. Or $\mathcal{S}^\star$ is feasible and better than $\mathcal{S}$, and we must build a positive explanation text.

In cases 1 and 2, if $\mathcal{N}(q)$ has a constant size, that is, all neighboring solutions are route-equal, precisely because all these solutions are route-equal, we can design a proof-like negative explanation text which works for all the solutions of $\mathcal{N}(q)$. On the opposite, if $\mathcal{N}(q)$ has a polynomial or exponential size because neighboring solutions are not all route-equal to each other, we rather resort to an argument-like negative explanation where we use $\mathcal{S}^\star$ as an example of neighboring solution ($\mathcal{S}'$ in the definition).

For example, let q be the (Ins, C) question: "Why is Ellen not performing task 27 just after task 17?" The corresponding support solution $\mathcal{S}^\star$ falls into the first case because $\underline{p}_{27} + d_{27} =$ 4:37 p.m. which is later than $\overline{a}_{27} =$ 3:00 p.m.. The prepared explanation template text is "⟨Employee i^*⟩ *would end* ⟨task j^*⟩ at the earliest at $\langle \underline{p}_{j^*} + d_{j^*} \rangle$ while ⟨task j^*⟩ is not available after $\langle \overline{a}_{j^*} \rangle$, that is why ⟨*the* unexpected fact is *observed*⟩." Once filled with values, this template text becomes a proof-like negative explanation text: "*Ellen would end* task 27 *at the earliest at 4:37 p.m. while* task 27 *is not available after 3:00 p.m., that is, why Ellen is not performing* task 27 *just after* task 17."

Finally, in the next subsection, we conduct numerical experiments on large-scale instances and solutions to measure the performances of our explanation text generation algorithms.

5.4. Numerical study about computation times to generate explanation texts

This subsection presents the numerical experiments that we conduct to assess the computation times for generating explanation texts on large-scale WSRP instances and solutions. First, we describe the instances and solutions used for these experiments. Then, we present our experimental setting as well as the obtained results. Finally, we analyze these results and provide some insights about our approach.

Instances and solutions. We carry out our numerical experiments over 96 pairs of instances and solutions, which are available in an online repository (Lerouge, 2023b). Instances are built from real data provided by our industrial partner DecisionBrain. These data are processed to be anonymous and adapted to our WSRP use case. The number of employees ranges from 17 to 80, the number of tasks from 55 to 2014, and skill levels from 1 to 4. For each instance, a solution was computed using a heuristic algorithm solving WSRP instances. Thus, solutions may not be optimal and explanations may be positive.

Experimental setting. In Section 4, we presented a list of question templates in Table 1. Given a template, many questions can be defined by filling its fields with different values (e.g., different employee names). In our numerical experiments, for each instance–solution pair and for each template of Table 1, we define a random sample of questions based on this template. We randomly

Table 2
Statistics about the explanation computation times using polynomial algorithms

Question template	Computation times			
	Median (seconds)	Third quartile (seconds)	Maximum (seconds)	Average (seconds)
(Ins, C)	0.022	0.035	0.162	0.025
(Ins, P, a)	0.023	0.038	0.104	0.026
(Ins, P, b)	0.114	0.273	3.395	0.213
(Ins, P, c)	0.061	0.093	0.180	0.067
(Ex, C)	0.023	0.035	0.208	0.025
(Ex, P, a)	0.023	0.037	0.091	0.026
(Ex, P, b)	0.115	0.282	3.855	0.222
(Ex, P, c)	0.063	0.095	0.192	0.070
(Ord, C, a)	0.023	0.037	0.100	0.027
(Ord, C, b)	0.024	0.038	0.100	0.027
(Ord, P, a)	0.027	0.041	0.108	0.030
(Ord, P, b)	0.027	0.041	0.120	0.031
(Ord, P, c)	0.030	0.045	0.116	0.034

Table 3
Results of the explanation computation times using ILP-based algorithms

Question template	Completed computation rate (%)	Computation times		
		First quartile (seconds)	Median (seconds)	Third quartile (seconds)
(Ins, E)	69.84	0.126	1.176	> 15
(Ex, E)	75.61	0.094	0.594	12.823
(Ord, E)	91.03	0.047	0.084	0.302

draw a sample of 40 different questions for each template. Moreover, we limit the time allowed for computing the explanation text answering each of these questions to 15 seconds, which we consider as a reasonable time limit—other time limit could be chosen depending on the application context. Then, for each question, we compute the explanation text by running one of the algorithms described in the previous subsections and interrupt it if the computation lasts for more than 15 seconds. The computations lasting for less than 15 seconds are counted as *completed* computations, and their times are saved. The others are counted as *interrupted* computations.

Algorithms were implemented in Python 3.9 and Gurobi Optimizer 9.5 for solving ILP models. Experiments were run in MacOS 10.15.7 on a 2.3-GHz Quad-Core Intel i7 processor with 16 GB RAM. Note that slightly better results could be expected with later versions of Python (3.11) and Gurobi Optimizer (11.0), which were released after our experiments but are also compatible with our implementation.

Results. We separate the results of computation times int two tables: Table 2 for the computations related to polynomial-time algorithms, and Table 3 for the ones related to ILP-based algo-

rithms. In both tables, each line provides statistics about the computation times associated with one template. To calculate these statistics, we group together the computation times obtained for all the instance–solution pairs and for all the random samples of questions by template. All the computations using polynomial-time algorithms have been completed in less than 15 seconds while part of the ones using algorithms relying on ILP have been interrupted. Therefore, we do not provide the same statistics in the two tables.

In Table 2, for each template, we compute the median, third quartile, maximum, and average of its corresponding computation times. For example, the generated explanation texts answering questions based on the (Ins, C) template have been computed on average in 0.025 seconds; among these texts, at least 50% are computed in less than 0.022 seconds, 75% in less than 0.035 seconds, and the longest to be computed has required 0.162 seconds.

In Table 3, for each template, we first compute its completed computations rate that is, the proportion of completed computations (i.e., computed in less than 15 seconds) among the computations of explanation texts related to this template; we then compute the first quartile, median, and third quartile of its computation times. For example, among all the computations of explanation texts answering questions based on the (Ins, E) template, 69.84% are completed in less than 15 seconds, at least 25% have lasted for less than 0.126 seconds, and at least 50% for less than 1.176 seconds. Note that, for the (Ins, E) template, the third quartile value is given as "> 15 seconds" since less than 75% of its related computations have been completed in less than 15 seconds.

Analysis and insights. Computational experiments show that explanation texts produced using polynomial-time algorithms are computed in a very short time: most of them are computed in less than 0.3 seconds. Such performances were expected due to the polynomial algorithmic complexity of the algorithms. The experiments also show that, within templates related to a given transformation family (insertion, exchange, or reordering), explanation texts related to 'Y'='P' templates generally require more time to be computed than the ones related to 'Y'='C' templates (e.g., on average, (Ins, P, c) texts require 0.067 seconds to be computed while (Ins, C) texts require 0.025 seconds). Such a trend could be anticipated since the size of the sets of neighboring solutions increases from 'Y'='C' to 'Y'='P' templates.

Regarding the explanation texts produced using ILP-based algorithms, whatever the template, experiments show that more than 50% of the explanation text computations last for a short time: less than 1.2 seconds. However, a significant proportion of these computations have been interrupted as they have reached the time limit of 15 seconds. Depending on the question template, this proportion varies from 8.97% to 30.16%.

Thus, execution times required for computing explanation texts are mostly compatible with the online use in an interactive system. However, for questions related to 'Y'='E' templates, we have no guarantee that they can be answered in a reasonable amount of time, shorter than 15 seconds.

6. Conclusion

We addressed the explanation of solutions stemming from an optimization system for end-users, in the context of a WSRP modeled as a bi-objective ILP. In our approach, explanations are triggered by user questions. These questions, assumed to be local, contrastive, and neighborhood-

interpretable, are facilitated by predefined templates. Our mathematical framework defined explanations, grounded in the feasibility of a foil model: they are either negative based on infeasible constraints or positive based on feasible solutions. Polynomial and ILP-based algorithms for explanation text generation were proposed. Performance evaluations on large-scale WSRP instances demonstrated that most explanation texts could be computed in less than 1.2 seconds, indicating suitability for online interactive use. However, our experiments highlighted certain limitations, particularly in ensuring that ILP-based algorithms can consistently provide explanations within a 15-second time frame.

In future works, various open questions could be investigated.

- The definition of the WSRP is not unique. Various WSRP complicating features have been investigated in the literature, for example, Goel and Meisel (2013) consider task precedence constraints, Bredström and Rönnqvist (2008) workforce synchronization constraints, and Chen et al. (2016) experience-based service times in the context of a multiday horizon. It would be interesting to study whether the approach developed in this paper may be extended to generate explanations for other WSRP variants.
- More generally, an interesting research perspective would be to assess the level of generalization capability of our approach by investigating its potential applicability to other CO problems. Even if our approach cannot be straightforwardly applied to any other CO problem, we have reasons to believe that it can be transposed to a variety of CO problems. First, our approach for modeling explanations hinges on solution neighborhoods, which is a concept rooted in local search. Second, the conceptual framework described in this paper, guiding the progression from an end-user question to an explanation, is problem-independent. Even though this paper focuses on a WSRP use case, the key concepts involved in this framework, such as neighborhood-related questions, foil models, positive and negative explanations, etc., are not specific to the WSRP and could be used for other problems. Finally, our explanation computation techniques do not depend on the algorithm used to solve instances of the studied CO problem. They only require a solution to the instance, without needing to know how it was computed.
- Contrastive reasoning, described in this paper, and counterfactual, presented in Lerouge et al. (2023), are two possible ways to provide explanations. Then, a comprehensive explanation framework could put together both types. Specifically, a counterfactual explanation could serve as recourse to a negative contrastive one, to highlight how the instance could be altered to obtain a feasible and better neighboring solution. This sequence of contrastive followed by counterfactual explanations was first introduced in the Ph.D. thesis by Lerouge (2023a). We seek to expand on this idea in future work.

Acknowledgments

This work has been supported by the project PSPC AIDA: 2019-PSPC-09 funded by BPI-France. We thank Daniel Godard, Filippo Focacci, and Desirée Rigonat from DecisionBrain for their time and their helpful feedback throughout this work.

References

AIA, 2021. Regulation (EU) 2021/0106 of the European Parliament and of the Council of 14 April 2021 on a European approach for artificial intelligence and amending certain Union legislative acts. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52021PC0206

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F., 2020. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58, 82–115.

Bogaerts, B., Gamba, E., Guns, T., 2021. A framework for step-wise explaining how to solve constraint satisfaction problems. *Artificial Intelligence* 300, 103550.

Bredström, D., Rönnqvist, M., 2008. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research* 191, 1, 19–31.

Cambazard, H., Jussien, N., 2006. Identifying and exploiting problem structures using explanation-based constraint programming. *Constraints* 11, 295–313.

Cashmore, M., Collins, A., Krarup, B., Krivic, S., Magazzeni, D., Smith, D., 2019. Towards explainable AI planning as a service. Paper presented at the International Conference on Automated Planning and Scheduling Second Workshop on Explainable Planning.

Castillo-Salazar, J.A., Landa-Silva, D., Qu, R., 2016. Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research* 239, 39–67.

Chakraborti, T., Sreedharan, S., Kambhampati, S., 2020. The emerging landscape of explainable AI planning and decision making. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. IJCAI Organization, Darmstadt, Germany, pp. 4803–4811.

Chandrasekaran, B., Tanner, M., Josephson, J., 1989. Explaining control strategies in problem solving. *IEEE Expert* 4, 9–15.

Chen, X., Thomas, B.W., Hewitt, M., 2016. The technician routing problem with experience-based service times. *Omega* 61, 49–61.

Čyras, K., Karamlou, A., Lee, M., Letsios, D., Misener, R., Toni, F., 2020. AI-assisted schedule explainer for nurse rostering. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 2101–2103.

Čyras, K., Letsios, D., Misener, R., Toni, F., 2019. Argumentation for explainable scheduling. In *Proceedings of the Thirty-Third Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*. AAAI Press, Washington D.C., pp. 2752–2759.

DARPA, 2016. Explainable artificial intelligence (XAI) program. https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf

Doshi-Velez, F., Kim, B., 2017. Towards a rigorous science of interpretable machine learning. arXiv. https://doi.org/10.48550/arxiv.1702.08608.

Dung, P.M., 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77, 2, 321–357.

Forrest, J., Sripada, S., Pang, W., Coghill, G., 2018. Towards making NLG a voice for interpretable machine learning. In *Proceedings of the 11th International Conference on Natural Language Generation*. Association for Computational Linguistics, Tilburg University, The Netherlands, pp. 177–182.

GDPR, 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj

Ginsberg, M., 1993. Dynamic backtracking. *Journal of Artificial Intelligence Research* 1, 25–46.

Goel, A., Meisel, F., 2013. Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research* 231, 1, 210–228.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D., 2018. A survey of methods for explaining black box models. *ACM Computing Surveys* 51, 1–42.

Gunning, D., Aha, D., 2019. DARPA's explainable artificial intelligence (XAI) program. *AI Magazine* 40, 44–58.

Junker, U., 2004. Quickxplain: Preferred explanations and relaxations for over-constrained problems. In *Proceedings of the Ninetieth Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*. AAAI Press, Washington, DC, pp. 167–172.

Jussien, N., Ouis, S., 2001. User-friendly explanations for constraint programming. Paper presented at the Proceedings of the Eleventh Workshop on Logic Programming Environments.

de Kleer, J., 1986. Problem solving with the ATMS. *Artificial Intelligence* 28, 197–224.

Korikov, A., Shleyfman, A., Beck, C., 2021. Counterfactual explanations for optimization-based decisions in the context of the GDPR. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. IJCAI Organization, Darmstadt, Germany, pp. 4097–4103.

Krarup, B., Krivic, S., Magazzeni, D., Long, D., Cashmore, M., Smith, D.E., 2022. Contrastive Explanations of Plans through Model Restrictions. *Journal of Artificial Intelligence Research* 72, 533–612.

Lerouge, M., 2023a. Designing and generating user-centered explanations about solutions of a workforce scheduling and routing problem. Ph.D. thesis, Université Paris-Saclay.

Lerouge, M., 2023b. WSRP-data. https://github.com/MathieuLerouge/WSRP-data.

Lerouge, M., Gicquel, C., Mousseau, V., Ouerdane, W., 2023. Counterfactual explanations for workforce scheduling and routing problems. In *Proceedings of the 12th International Conference on Operations Research and Enterprise Systems (ICORES)*. SciTePress, Set bal, Portugal, pp. 50–61.

Lewis, D., 1973. *Counterfactuals*. Cambridge, MA: Blackwell.

Liao, Q.V., Gruen, D., Miller, S., 2020. Questioning the AI: Informing design practices for explainable AI user experiences. In *Proceedings of the 2020 Conference on Human Factors in Computing Systems*, Association for Computing Machinery, New York, pp. 1–15.

Lindsay, A., 2020. Using generic subproblems for understanding and answering queries in XAIP. In *Proceedings of the 2020 International Conference on Automated Planning and Scheduling Workshop on Knowledge Engineering for Planning and Scheduling*, Nancy, France, October 26–20, 2020.

Lipton, P., 1990. Contrastive explanation. *Royal Institute of Philosophy Supplement* 27, 247–266.

Ludwig, J., Kalton, A., Stottler, R., 2018. Explaining complex scheduling decisions. In *Proceedings of the 2018 Association of Computing Machinery Conference on Intelligent User Interfaces*, Vol. 2068, ACM Press, New York.

Miller, T., 2019. Explanation in artificial intelligence: insights from the social sciences. *Artificial Intelligence* 267, 1–38.

Miller, T., 2021. Contrastive explanation: a structural-model approach. *The Knowledge Engineering Review* 36, e14.

Mohseni, S., Zarei, N., Ragan, E.D., 2021. A multidisciplinary survey and framework for design and evaluation of Explainable AI systems. *ACM Transactions on Interactive Intelligent Systems* 11, 3–4.

Mosquera, F., Smet, P., Vanden Berghe, G., 2019. Flexible home care scheduling. *Omega* 83, 80–95.

Poli, J.P., Ouerdane, W., Pierrard, R., 2021. Generation of textual explanations in XAI: the case of semantic annotation. In *Proceedings of the 2021 Institute of Electrical and Electronics Engineers International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, Piscataway, NJ, pp. 1–6.

Simonyan, K., Vedaldi, A., Zisserman, A., 2014. Deep inside convolutional networks: visualising image classification models and saliency maps. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, Banff, AB, Canada, April 14–16, 2014. abs/1312.6034.

Sqalli, M., Freuder, E., 1996. Inference-based constraint satisfaction supports explanation. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*. AAAI Press, Washington, DC, pp. 318–325.

Sreedharan, S., Srivastava, S., Kambhampati, S., 2018. Hierarchical expertise level modeling for user specific contrastive explanations. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI Organization, Darmstadt, Germany, pp. 4829–4836.

Stepin, I., Alonso, J.M., Catala, A., Pereira-Fariña, M., 2021. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *IEEE Access* 9, 11974–12001.

Swartout, W., Smoliar, S., 1987. On making expert systems more like experts. *Expert Systems* 4, 3, 196–208.

Volte, G., Desdouits, C., Giroudeau, R., 2019. The Workforce routing and scheduling problem: solving real-world instances. In *Proceedings of the Ninth International Network Optimization Conference*. OpenProceedings, Konstanz, Germany, pp. 60–65.

Wachter, S., Mittelstadt, B., Russell, C., 2018. Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harvard Journal of Law & Technology* 31, 841–887.

Wick, M., Thompson, W., 1992. Reconstructive expert system explanation. *Artificial Intelligence* 54, 33–70.

Zeiler, M., Fergus, R., 2014. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*. Springer, Berlin, pp. 818–833.

Appendix A: Solution

Table A1
Description of the solution $\mathcal{S} = ((\mathcal{R}_i, \mathcal{C}_i))_{i\in\mathcal{E}}$ represented in Fig. 1. $\mathcal{S}$ is a feasible solution of the WSRP instance given in Table B1. Each employee $i \in \mathcal{E}$ is associated with a planning $(\mathcal{R}_i, \mathcal{C}_i)$ made of, first, a route $\mathcal{R}_i$, which is a sequence of activities starting with the departure b_i of i (from their home location), followed with the tasks $j \in \mathcal{T}$ that i performs, and ending with the return e_i of i (to their home location); second, a schedule $\mathcal{C}_i$, which is a sequence of dates at which the corresponding activities of $\mathcal{R}_i$ start to be performed by i. (See Subsection 3.1 for the definition of a solution to a WSRP instance.)

Employee (i)	Planning $(\mathcal{R}_i, \mathcal{C}_i)$								
1	$\mathcal{R}_1 = (b_1,$	7,	30,	3,	26,	1,	17,	8,	$e_1)$
	$\mathcal{C}_1 = (485,$	525,	567,	662,	720,	840,	900,	1009,	1080)
	$\equiv$ (08:05 a.m.,	08:45 a.m.,	09:27 a.m.,	11:02 a.m.,	12:00 p.m.,	02:00 p.m.,	03:00 p.m.,	04:49 p.m.,	06:00 p.m.)
2	$\mathcal{R}_2 = (b_2,$	28,	16,	25,	14,	20,	6,	19,	$e_2)$
	$\mathcal{C}_2 = (540,$	600,	653,	702,	822,	888,	933,	1019,	1080)
	$\equiv$ (09:00 a.m.,	10:00 a.m.,	10:53 a.m.,	11:42 a.m.,	01:42 p.m.,	02:48 p.m.,	03:33 p.m.,	04:59 p.m.,	6:00 p.m.)
3	$\mathcal{R}_3 = (b_3,$	11,	9,	10,	24,	$e_3)$			
	$\mathcal{C}_3 = (480,$	542,	670,	717,	840,	906)			
	$\equiv$ (08:00 a.m.,	09:02 a.m.,	11:10 a.m.,	11:57 a.m.,	02:00 p.m.,	03:06 p.m.)			
4	$\mathcal{R}_4 = (b_4,$	13,	18,	21,	29,	14,	22,	$e_4)$	
	$\mathcal{C}_4 = (480,$	482,	564,	656,	738,	866,	925,	1080)	
	$\equiv$ (08:00 a.m.,	08:02 a.m.,	09:24 a.m.,	10:56 a.m.,	12:18 p.m.,	02:26 p.m.,	03:25 p.m.,	06:00 p.m.)	
5	$\mathcal{R}_5 = (b_5,$	5,	23,	2,	$e_5)$				
	$\mathcal{C}_5 = (529,$	600,	840,	1009,	1080)				
	$\equiv$ (08:49 a.m.,	10:00 a.m.,	02:00 p.m.,	04:49 p.m.,	06:00 p.m.)				

Appendix B: Instance

Table B1
Description of the instance on which the solution represented in Fig. 1. The first table describes the data about the set of employees $\mathcal{E}$. Each employee $i \in \mathcal{E}$ is characterized by a skill level s_i, a departure and return location, and a time-window $[\underline{w}_i,\ \overline{w}_i]$. The second table describes the data about the tasks. Each task j is described by a skill level r_j, a location, a duration d_j, and a time-window $[\underline{a}_j,\ \overline{a}_j]$. It is assumed that all the employees have the same traveling speed of 50 km/h. Considering that the earth radius is 6731 km, the traveling time, in minutes, between two locations $(lat_1, long_1)$ and $(lat_2, long_2)$ is computed as follows: $6731 \times \arccos(\sin(lat_1) \times \sin(lat_2) + \cos(lat_1) \times \cos(lat_2) \times \cos(long_2 - long_1))/50 \times 60$. (See Subsection 3.1 for the definition of a WSRP instance.)

Employee i In $\mathcal{E}$	Name	Skill level s_i in $\mathbb{N}^\star$	Location as (lat., long.) (deg.)	Time window $[\underline{w}_i,\ \overline{w}_i]$ as time range	as integer range
1	Ellen	2	(47.773, 16.193)	[8:00 a.m., 6:00 p.m.]	[480, 1080]
2	Alex	3	(47.598, 15.785)	[9:00 a.m., 6:00 p.m.]	[540, 1080]
3	Adam	2	(48.188, 14.862)	[8:00 a.m., 6:00 p.m.]	[480, 1080]
4	Fabian	1	(48.069, 14.485)	[8:00 a.m., 6:00 p.m.]	[480, 1080]
5	Carlotta	1	(47.463, 15.351)	[8:00 a.m., 6:00 p.m.]	[480, 1080]

Task j in $\mathcal{T}$	Skill level r_j in $\mathbb{N}^\star$	Location As (lat., long.) (deg.)	Duration d_j (min.)	Time window $[\underline{a}_j,\ \overline{a}_j]$ as time range	as integer range
1	1	(48.129, 15.754)	40	[8:00 a.m., 6:00 p.m.]	[480, 1080]
2	1	(47.240, 15.430)	40	[2:00 p.m., 6:00 p.m.]	[840, 1080]
3	1	(47.803, 15.348)	40	[11:00 a.m., 6:00 p.m.]	[660, 1080]
4	1	(47.913, 15.539)	30	[2:00 p.m., 6:00 p.m.]	[840, 1080]
5	1	(47.240, 14.639)	40	[8:00 a.m., 12:00 p.m.]	[480, 720]
6	3	(47.719, 15.856)	40	[8:00 a.m., 6:00 p.m.]	[480, 1080]
7	2	(47.691, 15.979)	30	[8:45 a.m., 12:00 p.m.]	[525, 720]
8	1	(47.904, 15.988)	40	[8:00 a.m., 6:00 p.m.]	[480, 1080]
9	1	(48.252, 15.159)	40	[9:00 a.m., 12:00 p.m.]	[540, 720]
10	2	(48.233, 15.094)	80	[9:00 a.m., 6:00 p.m.]	[540, 1080]
11	2	(47.879, 14.348)	40	[8:00 a.m., 6:00 p.m.]	[480, 1080]
12	2	(47.151, 15.559)	40	[8:00 a.m., 6:00 p.m.]	[480, 1080]
13	1	(48.073, 14.500)	30	[8:00 a.m., 6:00 p.m.]	[480, 1080]
14	1	(47.509, 15.979)	50	[8:00 a.m., 6:00 p.m.]	[480, 1080]
15	2	(48.182, 15.480)	25	[8:00 a.m., 2:00 p.m.]	[480, 840]
16	1	(47.307, 16.030)	30	[8:00 a.m., 6:00 p.m.]	[480, 1080]
17	1	(48.174, 15.768)	40	[12:30 p.m., 4:00 p.m.]	[750, 960]
18	1	(47.696, 14.639)	40	[8:00 a.m., 6:00 p.m.]	[480, 1080]
19	2	(47.466, 15.662)	40	[8:00 a.m., 6:00 p.m.]	[480, 1080]
20	2	(47.624, 15.928)	30	[8:00 a.m., 6:00 p.m.]	[480, 1080]
21	1	(47.893, 15.139)	45	[8:00 a.m., 6:00 p.m.]	[480, 1080]
22	1	(47.799, 15.814)	30	[3:00 p.m., 6:00 p.m.]	[900, 1080]
23	1	(47.430, 15.794)	30	[10:00 a.m., 3:00 p.m.]	[600, 900]
24	2	(48.178, 15.144)	40	[2:00 p.m., 3:00 p.m.]	[840, 900]
25	1	(47.382, 16.094)	40	[8:00 a.m., 1:00 p.m.]	[480, 780]
26	2	(47.809, 15.206)	40	[12:00 p.m., 6:00 p.m.]	[720, 1080]
27	2	(47.948, 15.950)	50	[8:00 a.m., 3:00 p.m.]	[480, 900]
28	3	(47.160, 15.991)	30	[8:00 a.m., 6:00 p.m.]	[480, 1080]
29	1	(47.829, 15.253)	40	[8:00 a.m., 1:00 p.m.]	[480, 780]
30	1	(47.646, 15.907)	40	[8:00 a.m., 12:00 p.m.]	[480, 720]
31	2	(47.399, 15.535)	40	[8:00 a.m., 4:00 p.m.]	[480, 960]