Designing and generating user-centered explanations about solutions of a Workforce Scheduling and Routing Problem

Mathieu Lerouge

université
PARIS-SACLAY

Designing and generating user-centered explanations about solutions of a Workforce Scheduling and Routing Problem

Conception et génération d'explications à propos des solutions d'un problème de planification d'employés mobiles pour les utilisateurs d'un système d'optimisation

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°573 Interfaces : Matériaux, systèmes, usages
Spécialité de doctorat : Informatique
Graduate School : Sciences de l'ingénierie et des systèmes. Référent : voir CentraleSupélec

Thèse préparée dans l'unité de recherche **Mathématiques et Informatique pour la Complexité et les Systèmes** (Université Paris-Saclay, CentraleSupélec), sous la direction de **Vincent MOUSSEAU**, Professeur, Université Paris-Saclay, CentraleSupélec (MICS), de **Wassila OUERDANE**, Maître de conférence HDR, Université Paris-Saclay, CentraleSupélec (MICS), et **Céline GICQUEL**, Maître de conférence HDR, Université Paris Saclay (LISN).

Thèse soutenue à Paris-Saclay, le 27 novembre 2023, par

Mathieu LEROUGE

Composition du Jury

Membres du jury avec voix délibérative

Odile BELLENGUEZ Professeur, IMT Atlantique	Présidente
François CLAUTIAUX Professeur, Université de Bordeaux	Rapporteur & Examinateur
Alexis TSOUKIAS Directeur de recherche CNRS, Université Paris Dauphine	Rapporteur & Examinateur
Claudia ARCHETTI Professeur, ESSEC Business School	Examinatrice

THESE DE DOCTORAT
NNT : 2023UPAST174

université PARIS-SACLAY | ÉCOLE DOCTORALE Interfaces: matériaux, systèmes, usages

Titre: Conception et génération d'explications à propos des solutions d'un problème de planification d'employés mobiles pour les utilisateurs d'un système d'optimisation

Mots clés: Optimisation Combinatoire, Intelligence Artificielle, Problèmes de planification d'employés mobiles, Explications orientées utilisateur, Explications contrastives, scénarios et contrefactuelles, Système d'aide à la décision.

Résumé: Les systèmes d'aide à la décision basés sur l'optimisation combinatoire trouvent des applications dans divers domaines professionnels. Cependant, les décideurs qui utilisent ces systèmes ne comprennent souvent pas les concepts mathématiques et les principes algorithmiques qui les sous-tendent. Ce manque de compréhension peut entraîner du scepticisme et une réticence à accepter les solutions générées par le système, érodant ainsi la confiance placée dans le système. Cette thèse traite cette problématique dans le cas du problème de planification d'employés mobiles, en anglais *Workforce Scheduling and Routing Problem* (*WSRP*), un problème d'optimisation combinatoire couplant de l'allocation de ressources humaines et du routage. Tout d'abord, nous proposons un cadre qui modélise le processus d'explication de solutions pour les utilisateurs d'un système de résolution de WSRP, permettant d'aborder une large gamme de sujets. Les utilisateurs initient le processus en faisant des observations sur une solution et en formulant des questions liées à ces observations grâce à des modèles de texte prédéfinis. Ces questions peuvent être de type contrastif, scénario ou contrefactuel. D'un point de vue mathématique, elles reviennent essentiellement à se demander s'il existe une solution faisable et meilleure dans un voisinage de la solution courante. Selon les types de questions, cela conduit à la formulation d'un ou de plusieurs problèmes de décision et de programmes mathématiques. Ensuite, nous développons une méthode pour générer des textes d'explication de différents types, avec un vocabulaire de haut niveau adapté aux utilisateurs. Notre méthode repose sur des algorithmes efficaces calculant du contenu explicatif afin de remplir des modèles de textes d'explication. Des expériences numériques montrent que ces algorithmes ont des temps d'exécution globalement compatibles avec une utilisation en temps quasi-réel des explications par les utilisateurs. Enfin, nous présentons un design de système structurant les interactions entre nos techniques de génération d'explications et les utilisateurs qui reçoivent les textes d'explication. Ce système sert de base à un prototype d'interface graphique visant à démontrer l'applicabilité pratique et les potentiels bénéfices de notre approche dans son ensemble.

Title: Designing and generating user-centered explanations about solutions of a Workforce Scheduling and Routing Problem

Keywords: Combinatorial Optimization, Artificial Intelligence, Working Scheduling and Routing Problem, User-centered explanations, Contrastive, scenario and counterfactual explanations, Decision support system

Abstract: Decision support systems based on combinatorial optimization find application in various professional domains. However, decision-makers who use these systems often lack understanding of their underlying mathematical concepts and algorithmic principles. This knowledge gap can lead to skepticism and reluctance in accepting system-generated solutions, thereby eroding trust in the system. This thesis addresses this issue in the case of the Workforce Scheduling and Routing Problems (WSRP), a combinatorial optimization problem involving human resource allocation and routing decisions. First, we propose a framework that models the process for explaining solutions to the end-users of a WSRP-solving system while allowing to address a wide range of topics. End-users initiate the process by making observations about a solution and formulating questions related to these observations using predefined template texts. These questions may be of contrastive, scenario or counterfactual type. From a mathematical point of view, they basically amount to asking whether there exists a feasible and better solution in a given neighborhood of the current solution. Depending on the question types, this leads to the formulation of one or several decision problems and mathematical programs. Then, we develop a method for generating explanation texts of different types, with a high-level vocabulary adapted to the end-users. Our method relies on efficient algorithms for computing and extracting the relevant explanatory information and populates explanation template texts. Numerical experiments show that these algorithms have execution times that are mostly compatible with near-real-time use of explanations by end-users. Finally, we introduce a system design for structuring the interactions between our explanation-generation techniques and the end-users who receive the explanation texts. This system serves as a basis for a graphical-user-interface prototype which aims at demonstrating the practical applicability and potential benefits of our approach.

À Éliane et Gisèle

Remerciements

Le lundi 27 novembre 2023, dans l'intimité de l'amphithéâtre e.068, doté de ses modestes quatre rangs de sièges, j'ai eu le privilège de soutenir ma thèse de doctorat, autrement dit, d'exposer trois ans de travail, dans un temps imparti de 45 minutes, face aux regards critiques d'experts, disposés à prolonger ce moment de vulnérabilité par une longue série de questions... Alors que ce moment aurait normalement dû déclencher mon habituelle angoisse des présentations publiques, je l'ai étonnamment vécu avec tranquillité. Certaines personnes ont interprété mon calme comme l'expression de ma supposée confiance, due à ma potentielle maîtrise approfondie de mon sujet. Pour ma part, je préfère attribuer ma sérénité à la bienveillance et au soutien que me porte mon auditoire, et que me portent plus largement toutes les personnes représentées par mon auditoire, à savoir, ma direction de thèse, mes collègues, mes amis et ma famille. Cet environnement social constitue pour moi une richesse profonde et une source de fierté considérable. Ainsi, je souhaite exprimer ma gratitude à leur égard dans les paragraphes qui suivent.

Céline, Vincent, Wassila, merci beaucoup pour votre confiance, votre disponibilité et votre soutien pendant ces trois années de collaboration. En formant un trio complémentaire et cohérent, tant sur le plan technique, humain et managérial, vous avez su me guider dans mon travail de recherche, depuis mes premières lectures d'articles jusqu'à ma défense de thèse. Céline, je retiendrai de cet oral de présentation de thèse, le regard attentif et lumineux que tu m'as porté tout au long de mon discours, qui je crois reflète assez bien ta bienveillance et ta douceur, constantes, au cours de ces trois années. Vincent, je garderai en mémoire ton attitude souriante et détendue en ce 27 novembre, toi qui sais toujours apporter de la légèreté au sérieux, dont le potentiel angoissant en est atténué par la même occasion. Wassila, je me souviendrai entre autres de tes discrets hochements de tête et de tes légers plissements d'yeux pendant le moment des questions, les premiers signifiant ton accord avec mes propos et les seconds me suggérant que des précisions sont possibles, le tout pour m'inciter à toujours donner le meilleur de moi-même - une attitude que tu as tenue ici pendant la soutenance, mais plus généralement pendant toute la thèse. Merci encore à vous trois pour ces trois années de travail et de plaisir !

Je tiens ensuite à exprimer ma gratitude envers les membres de mon jury, Alexis, François, Claudia et Odile. Merci à vous quatre pour l'intérêt que vous avez porté pour mon travail, pour vos questions, pour les discussions que nous avons eu le jour de ma soutenance, ainsi que pour vos mots de félicitations à l'issue des délibérations. En particulier, Alexis et François, merci pour vos rapports encourageants qui m'ont rassurés en amont de la soutenance. Fidèle à moi-même François, j'ai décidé de faire des remerciements un peu verbeux !

Je tiens ensuite à remercier tous mes collègues de DecisionBrain. Daniel et Filippo, merci pour m'avoir accueilli au sein de l'entreprise, malgré mon statut extérieur - que je m'amuse souvent à qualifier de "parasite". Avec bientôt le projet de post-doc à Bologne, il est désormais certain que notre collaboration ne va pas s'arrêter à cette thèse. Merci également à vous deux ainsi qu'à Désirée pour l'intérêt que vous avez porté pour mon travail de recherche, illustré notamment par les nombreuses réunions au début de ma thèse, lorsque j'avais besoin de me définir des pistes à explorer et alors que je n'avais pas forcément beaucoup de matière à présenter. Anne-Laurence et Mehdi, merci pour toutes ces heures d'enseignement partagées, à discuter du contenu à transmettre à nos élèves mais également de la meilleure pédagogie à adopter pour y parvenir. Bruno, merci de m'avoir accompagné à ma toute première conférence, où je devais exposer mes réflexions face à de nombreux industriels expérimentés, rendant le moment un peu impressionnant. Sébastien, merci pour tous tes conseils avisés sur la rédaction de mon manuscrit et sur la préparation de ma soutenance - ainsi que pour tous tes petits desserts ! Roberta, chère happiness manager, merci pour toutes tes propositions d'activités, ta joie et ta sincérité. Alexa, merci pour avoir contribué volontairement ou involontairement à me sentir à l'aise au bureau en étant ma première voisine de bureau dans l'open space, avec qui je partageais des musiques, des recettes, etc. - y gracias por las manzanas con tajin! Elisa, merci pour ton écoute, nos partages de podcasts et nos conversations en anglais, français et italien - qui tendent de plus en plus à être dans les deux dernières langues, certo! Giulia, grazie per i compiti e grazie per la salsa, ci vediamo a Bologna! Antoine et Marie, merci pour ces descentes de piste de ski et ces parties de babyfoot - auxquelles je suis toujours aussi mauvais ! Nevra, thank you for all your energy, your curiosity and our passionate talks about paintings. Merci, merci à tous. Il n'y a aucun doute que bénéficier d'un tel environnement professionnel m'a apporté une grande richesse et une grande stabilité dans mon quotidien.

J'aimerais ensuite dédier les prochaines lignes à remercier mes amies et amis.

Il y a d'abord celles et ceux que je connais depuis des années - voire même véritablement des décennies pour certaines personnes ! Chiddi, merci pour toutes ces années, pour nos innombrables conversations, au téléphone, au restaurant ou dans la voiture, à 3h du matin, stationnée depuis déjà plusieurs heures. Merci pour ton écoute, ton ouverture d'esprit, ta curiosité. Merci de me faire grandir au travers de nos conversations, les podcasts, les vidéos, les recettes, tous les contenus que l'on se partage. Lucas, merci également pour toutes ces années, nos soirées, nos vacances, nos week-ends à la campagne, nos repas et apéros maison avec Juju, nos fêtes de Nouvel An - surtout celui aux Sables ! Merci pour ton humour, tes bêtises et ta lourdeur - tiens, ça me rappelle quelqu'un ça ! Tanguy, merci pour ta curiosité, pour

avoir pris le temps de t'informer sur mon travail et puis pour avoir fait le déplacement jusqu'à CentraleSupélec pour ma soutenance. Merci pour nos séances d'escalade, passées et futures - ainsi qu'à nos séances de surf, passées et futures ?... Flavio, mon broc, merci pour ta sincérité, ta générosité, ton sens de l'accueil. Merci également d'être venu assister à ma soutenance. Bientôt l'ouverture de la maison d'hôtes ? Camille, mon éternel binôme d'anniversaire, merci pour tes fous rires, ta tendresse, tes maladresses. Et j'espère que l'avenir sera fait de plus d'Uzès, avec les petits Michels !

Il y a également celles et ceux que j'ai rencontrés dans les études supérieures, en particulier à l'École des Ponts. Séverine et Yanis, merci d'être venus assister à ma soutenance pour représenter ce petit groupe et merci d'avoir bravé la galère des transports en commun pour venir jusqu'à CentraleSupélec ! Darius et Victor, merci pour la curiosité que vous avez porté envers mon travail au cours de ces années de thèse et pour avoir assisté à ma soutenance en ligne. Nerea, partenaire de sorties cinéma et visites de musée, merci pour ton écoute et ton soutien, en particulier dans les périodes où j'en ai eu besoin - y claro, gracias a ti y tus padres por el curso de preparación de tortilla en Bilbao! Merci à Jeanne, mon ancienne colocataire rue des Prairies qui m'a vu démarrer cette thèse, à Bastien, notre professeur de maths préféré, Baptiste, notre futur docteur, Pierre notre déjà-docteur. Kaio, obrigado por ter sido curioso o suficiente para assistir à minha defesa, online do Brasil e, acima de tudo, obrigado por me receber em sua casa!

Et puis, il y a celles et ceux qui sont entrés plus récemment dans ma vie. Hanna, Karine et Michele, merci pour cette année de colocation, pour nos multiples conversations dans le salon, notre jungle à la lumière tamisée. Merci pour nos soirées en petits et grands comités, pour nos apéros, nos brunchs, nos cafés. Merci pour votre présence et votre soutien quotidien. Vous êtes l'une de mes plus belles expériences de colocation à ce jour et je compte bien en profiter pendant ces quelques prochains mois. Manu, ça a été un plaisir de faire ces trois années de thèse, de recherche, d'enseignement, de conférences en ta présence. Merci pour tous tes présents, bracelets, tee-shirt, statuette. Et bientôt à toi d'obtenir le titre de docteur ! Anaïs, merci pour ton intelligence, ton empathie, ta douceur, ton miel, et tes grains de maïs un peu partout dans tes plats !

Pour terminer, j'aimerais remercier ma famille, entre autres ma mère, mon père, ma soeur et mon frère. Maman, papa, merci d'être venus assister à ma soutenance, et désolé de ne pas m'être étendu le jour de ma soutenance dans les remerciements à votre égard, parce que quand j'ai croisé vos yeux brillants d'humidité, je me suis dit que je n'allais pas parvenir à retenir mes émotions bien longtemps. Merci pour votre confiance, votre soutien, notamment dans mon parcours scolaire. Finalement, je ne serai pas architecte, mais docteur en informatique ! Par contre, je n'ai toujours pas écarté l'idée d'une reconversion dans le milieu de la cuisine dans le futur... Débo, merci d'avoir assisté à ma soutenance en ligne et d'avoir fait ma promotion auprès de tes collègues. Je sais que tu aurais aimé être présente sur place mais je te savais présente en ligne, c'est le plus important. Et merci à toi et à Adri de faire que la famille Lerouge s'agrandisse, ce qui promet de nombreuses heureuses fêtes de famille !

Contents

List of Figures

List of Tables

List of Algorithms

Notations

Workforce Scheduling and Routing Problem

$\mathcal{I}$	Instance
$\mathcal{E}$	Set of employees
n	Number of employees
ske_i	Skill level of employee i
$[lbe_i,\ ube_i]$	Working time window of employee i
$\mathcal{T}$	Set of tasks
m	Number of tasks
dt_j	Duration of task j
skt_j	Skill level of task j
$[lbt_j,\ ubt_j]$	Availability time window of task j
$\mathcal{A}_i$	Set of activities of employee i
d_i	Departure of employee i from their home (activity)
r_i	Return of employee i to their home (activity)
tr_{jk}	Travel time between activities j and k
$\mathcal{S}$	Solution (*i.e.* family of plannings)
$(\mathcal{R}_i, \mathcal{C}_i)$	Planning (*i.e.* a route-schedule pair) of employee i
$\mathcal{R}_i$	Route of employee i
$\mathcal{C}_i$	Schedule of employee i
st_j	Start time of activity j (within an employee schedule)
$twt(\mathcal{S})$	Total working time of solution $\mathcal{S}$
$trt(\mathcal{S})$	Total travel time of solution $\mathcal{S}$
$mm(\mathcal{I})$	Main model related to instance $\mathcal{I}$
$\mathcal{X}$	Main model solution (*i.e.* assignment of the decision variables involved in the main model)
T_j	Start time decision variable (integer) fixing the start time of task j
U_{ijk}	Path decision variable (binary) fixing whether employee i goes from activity j to activity k or not
φ	Bijection mapping main model solutions into solutions

Solution transformations and neighborhoods

tf	Transformation
$\mathcal{N}(tf, \mathcal{S})$	Neighborhood induced by the transformation tf applied to the solution $\mathcal{S}$
b_j	Backward Time Slack (BTS) of activity j (as part of a given planning)
f_j	Forward Time Slack (FTS) of activity j (as part of a given planning)
st_j^b	Backward Earliest start Time (BET) of task j (when inserting j in a given planning)
st_j^f	Forward Latest start Time (FLT) of task j (when inserting j in a given planning)

Modeling explanations

o	Observation (about a given solution)
$\mathcal{I}'$	Other instance (usually a relaxation of an initial instance)
$\mathcal{J}'$	Set of other instances (usually set of relaxations of an initial instance)
q	Question (about a given solution)
$q_c(o, \mathcal{S}, \mathcal{I})$	Contrastive question: "Why o is observed in $\mathcal{S}$, solution of $\mathcal{I}$?"
$q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$	Scenario question: "o is observed in $\mathcal{S}$, solution of $\mathcal{I}$, but what if $\mathcal{I}$ is changed into $\mathcal{I}'$?"
$q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$	Counterfactual question: "o is observed in $\mathcal{S}$, solution of $\mathcal{I}$, but how to observe $\neg o$ considering that $\mathcal{I}$ can be changed into $\mathcal{I}' \in \mathcal{J}'$?"
$\mathcal{N}(o, \mathcal{S})$	Neighborhood induced by the transformation related to the observation o applied to the solution $\mathcal{S}$
$dp(o, \mathcal{S}, \mathcal{I})$	Decision problem related to observation o about $\mathcal{S}$, solution of instance $\mathcal{I}$
$dpq(q)$	Decision-problem interpreted question associated with question q
$fm(o, \mathcal{S}, \mathcal{I})$	Foil model related to observation o about $\mathcal{S}$, solution of instance $\mathcal{I}$
$fmq(q)$	Foil-model interpreted question associated with question q
$x(q)$	Explanation of question q
$ux(q)$	Explanation text of question q

Generating explanation texts

$\mathcal{I}^\star$	Support relaxation
$\mathcal{S}^\star$	Support solution
$(\mathcal{I}^\star, \mathcal{S}^\star)$	Support relaxation-solution pair
$ctm(o, \mathcal{S}, \mathcal{I})$	Contrastive transformation model solved for computing a support solution
$T^{lb}_{j^*}$	BET decision variable (integer) related to task j^*
$T^{ub}_{j^*}$	FLT decision variable (integer) related to task j^*
$htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$	Counterfactual transformation model solved for computing a support relaxation-solution pair
ΔDt_j	Altering decision variable (integer) for reducing the duration dt_j of task j
ΔLBT_j	Altering decision variable (integer) for decreasing the availability lower-bound lbt_j of task j
ΔUBT_j	Altering decision variable (integer) for increasing the availability upper-bound ubt_j of task j
XDt_j	Decision variable (binary) enabling the reduction of the duration dt_j of task j
$XLBT_j$	Decision variable (binary) enabling the decrease of the availability lower-bound lbt_j of task j
$XUBT_j$	Decision variable (binary) enabling the increase of the availability upper-bound ubt_j of task j
ΔT_{max}	Decision variable (integer) measuring the largest alteration
ΔLBE_i	Altering decision variable (integer) for decreasing the availability lower-bound lbt_j of task j
ΔUBE_i	Altering decision variable (integer) for increasing the availability upper-bound ubt_j of task j
$XLBE_i$	Decision variable (binary) enabling the decrease of the working time window lbe_i of employee i
$XUBE_i$	Decision variable (binary) enabling the increase of the working time window ube_i of employee i

Chapter 1 Introduction

1.1 Context and motivations

Nowadays, decision support systems based on Combinatorial Optimization (CO) find application in various professional domains. However, most often, the decision-makers who use these optimization-based systems do not have the necessary background to fully understand their mathematical concepts and algorithmic principles; and even if they do, they may be surprised by some aspects of the decisions proposed by the systems, leading to doubts about the relevance of these decisions. In both cases, the lack of understanding can erode the decision-makers' trust in the optimization-based systems, causing reluctance to implement the suggested decisions.

In this thesis, we focus on the case of organizations using decision support systems to enhance resource management by solving a CO problem called the *Workforce Scheduling and Routing Problem* (*WSRP*). The WSRP involves assigning geographically dispersed tasks to mobile employees and building a route-schedule pair for each employee. Such a pair specifies the time at which the employee begins and ends their working day, the tasks they have to perform, at what times and in what order. The solution of a WSRP should comply with various constraints, including constraints regarding employee working hours, task availability hours, skill requirements to perform tasks, *etc.* Being a generalization of vehicle routing problems and scheduling problems which are NP-hard problems, the WSRP is also an NP-hard problem [VDG19]. In the literature, it arises in many and various contexts such as home care services [MSV19] or heating, ventilation, and air-conditioning home services [CTH16]. A comprehensive literature review on the WSRP can be found in [CSLSQ16].

Focusing on the case of organizations solving the WSRP has been originally motivated by the needs of our industrial partner DecisionBrain[1]. DecisionBrain is a French company which develops and sells optimization software products, primarily in the fields of workforce management and production planning. The optimization problems that they address thus involve routing, scheduling and lot-sizing problems. Among others, they develop for their customers a decision support system for solving variants of the WSRP. Since the WSRP is an NP-hard problem, decision-makers using DecisionBrain's system may feel overwhelmed by the high combinatorial aspect of the problem. They sometimes have difficulties in appreciating why generated solutions are good ones and contact DecisionBrain to ask questions about these solutions.

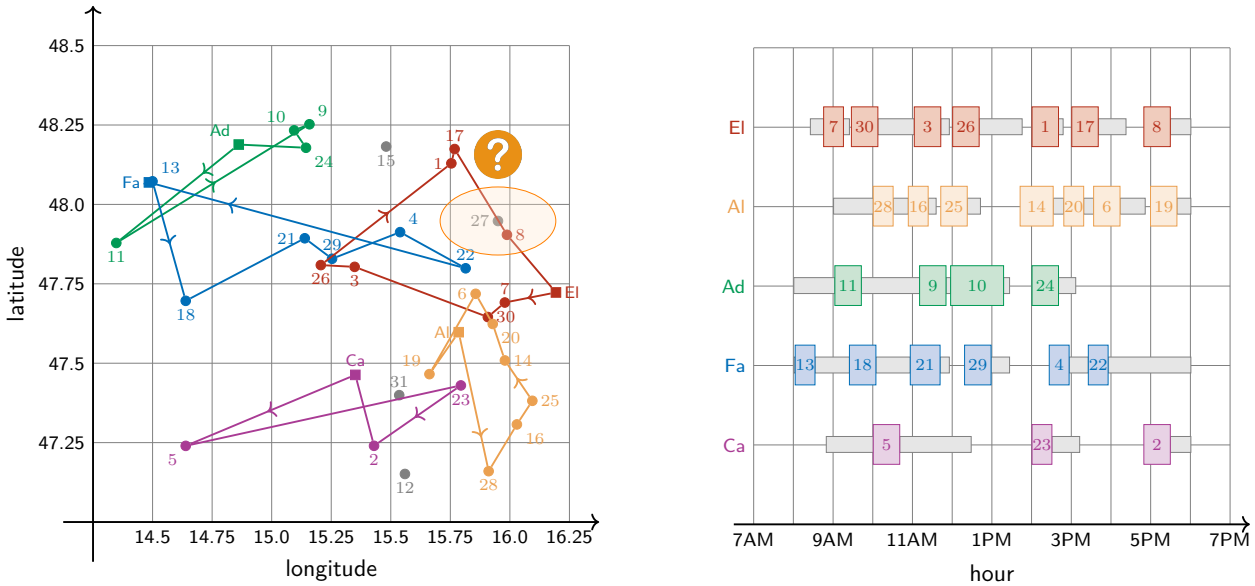


Figure 1.1: Intriguing fact observed (highlighted by the orange ellipse) in a WSRP solution whose routes are depicted on the left and schedules on the right.

Figure 1.1 depicts a typical situation where decision-markers are provided with a solution obtained thanks to a WSRP-solving system and wonder about some facts observed in the solution. The figure primarily represents a WSRP solution. Each employee is related to a color, *e.g.* 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 non-performed tasks, while squares correspond

[1] https://decisionbrain.com/

to employee starting locations, which we call home for convenience. The Gantt chart on the right depicts schedules: high and colored rectangles represent tasks; small and 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. While looking at this figure, the decision-makers may be intrigued by a fact observed in the solution. Namely, in the graph of routes on the left, an orange ellipse spotlights that Ellen is not performing task 27 even though it is on her route between task 17 and task 8. There may be various causes behind this fact: *e.g. i)* Ellen might not be skilled enough to perform task 27, *ii)* she might not be able to be at task 27 early enough to perform it while it is available, or *iii)* she might actually be able to perform task 27 but the heuristic WSRP-solving system missed it. Therefore, interested in knowing the causes behind this fact, Decision-makers may wonder: "Why is Ellen not performing task 27 in addition to her tasks?" On one hand, answering such questions is time-consuming for system designers, like DecisionBrain, as it supposes firstly to get familiar with the WSRP instance and solution data, and secondly to find explanatory content to provide to the decision-makers. On the other hand, without answering these questions, decision-makers may gradually loose trust in the WSRP-solving system and be less inclined to use it. One way to tackle these issues is to design an approach for automatically generating explanations for decision-makers using WSRP-solving systems.

Developing techniques for generating explanations falls under the wide field of Explainable Artificial Intelligence (XAI) [GA19] and relates to what the XAI community calls explainability. In Machine Learning (ML), explainability is defined as follows [BADRDS$^+$20]: "Given a certain audience, explainability refers to the details and reasons a model gives to make its functioning clear or easy to understand." Over the last ten years, explainability has been attracting substantial attention in the Artificial Intelligence (AI) community, especially in the ML one [BADRDS$^+$20]. This strong interest has been driven by initiatives such as the XAI research program funded by the Defense Advanced Research Projects Agency[2] in the United States and by the recent introduction of the General Data Protection Regulations [GDP16] in Europe. The latter 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 share common goals, including algorithmic transparency, user trust and bias mitigation [MZR21]. Besides, they often rely on the same fundamentals concepts rooted in social sciences and philosophy such as the notions of contrastive question [Lip90] or counterfactual explanations [Lew73]. Works on explanation from different fields may thus inspire each other. However, the differences between these fields (in terms of problems, use cases, models, inputs, algorithms, *etc.*), pose challenges when attempting to directly transpose an explanation method developed for one field to another. For instance, methods generating explanations as saliency heatmaps (a.k.a. sensitivity maps) in Deep Learning [ZF14, SVZ14] 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, it is necessary to design explanation methods tailored specifically for CO contexts.

Yet, to the best of our knowledge, there are only a few works dealing with explanations for problems that could be formulated in CO terms, see [LKS18, ČLMT19, KSB21]. Ludwig *et al.* [LKS18] study a scheduling problem and seek to provide explanations about the sequence and timing of the tasks. However, the provided explanations tightly depend on the heuristic approach used for solving the problem, which supposes that the people receiving the explanations know about the heuristic and agree to solve their instances with this sub-optimal algorithm. Čyras *et al.* [ČLMT19] also consider a scheduling problem but propose an explanation method based on abstract argumentation [Dun95]. But this method seems to be applicable only to a specific class of integer linear programs: the program needs to be essentially based on binary variables and clique constraints on these variables, which is quite limiting. Korikov *et al.* [KSB21] focus on another specific class of integer linear programs. They restrain 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. This again limits the application scope of the proposed method. Besides, the three works aim at providing explanations about one to three different topics in the solution: *e.g.* [LKS18] only deals with explanations about why a given task is scheduled at a given time in the solution. To sum up, all these works rely on assumptions which strongly limit their applicability and make it challenging to employ the proposed explanation methods for other CO problems such as the WSRP, hence the need to develop new methods.

Research issues. However, designing a new approach for automatically generating explanations for decision-makers using WSRP-solving systems raises several challenging open research issues.

- **What are explanations?** How can they be mathematically modeled in a CO context, and especially in a WSRP context? Can we develop models that handle various explanation types and a large diversity of topics?
- **How can explanations be computed?** Is it possible to devise polynomial-time algorithms for explanation computation? Do we need to resort to solving mathematical programs? Are there computation techniques that are efficient enough to be compatible with near-real-time use of explanations by decision-makers?

[2] https://www.darpa.mil/program/explainable-artificial-intelligence

- **How can explanations be communicated?** Can explanations be presented in text form? Can we express explanation texts using a vocabulary that is adapted to decision-makers?

1.2 Proposals and contributions

This PhD thesis is a first attempt at addressing the research issues mentioned in the previous subsection: a framework, algorithms and a system are designed, aiming at modeling, computing, generating and communicating explanations about solutions of a WSRP use case to the end-users of a system solving this CO problem.

Proposals We first devise an original framework aiming at modeling the explanation process in a CO context, specifically applied to our WSRP use case. End-users initiate this process by making an observation about a solution and then request an explanation by formulating a question related to this observation. Observations are assumed to focus on alternative facts that end-users would have expected to see in the solution and are assumed to be inherently linked to solution neighborhoods - a concept drawn from the Local Search literature and widely employed in algorithms for solving various CO problems. The spectrum of topics covered by these observations can be quite large as we can potentially leverage any solution neighborhood to define an observation topic. To allow the expression of observations, we provide the end-users with banks of predefined template texts. From a given observation, three types of questions can be formulated resulting in three types of explanations: either contrastive, scenario or counterfactual. Contrastive explanations clarify why one fact occurred in a solution in contrast to an alternative fact. Scenario explanations describe how user-defined changes in the instance parameters affect a fact observed in the solution. Counterfactual explanations aim at identifying how data could be changed in the input instance to obtain a user-defined alternative solution. Just like observations, questions are expressed thanks to template texts.

From a CO point of view, these questions can be interpreted as the formulation of one or several decision problems. Basically, these decision problems amount to asking whether there exist a feasible and better solution of the problem within the neighborhood induced by the end-user question. Regardless of the question type, we show that answering these decision problems comes down to determining whether a mathematical program, we call the foil model, is feasible. We thus propose a mathematical definition of explanations based on the feasibility of this foil model.

We then conceive a method for producing explanations of various types (contrastive, scenario or counterfactual) taking the form of concise texts written in high-level vocabulary adapted to end-users. Given a question about an alternative fact that end-users expected to see in a solution, our method consists essentially in identifying another solution that we term "support solution". This support solution essentially corresponds to the "best" solution neighboring the end-users' solution that satisfies the alternative fact. It is rigorously defined as the best feasible solution or the "nearest-to-feasibility" solution within the neighborhood inherently related to the question. Subsequently, from the content and properties of the support solution, explanatory information can be extracted. This information is then used to fill pieces of template texts which we arrange together to form an explanation text. In order to compute the support solution, we develop either polynomial-time algorithms or algorithms that resort to solving Integer Linear Programming (ILP) models. Numerical experiments show that our algorithms have execution times that are mostly compatible with near-real-time use of explanations by end-users.

We finally propose an original system design for structuring the interactions between end-users and the method that we develop for generating explanation texts. Such a system enables end-users to raise questions about a solution of a WSRP instance and receive explanation texts in return. It also enables them to explore the set of feasible solutions as well as to alter the parameters of the instance, if it helps end-users getting a desired alternative fact. This system, along with the method and algorithms for generating explanation texts, are integrated into a Graphic User Interface (GUI) prototype that we have implemented as a way to demonstrate the potential and interest of our proposals as a whole.

Main contributions The main contributions of this thesis are the following ones.

- **A comprehensive framework modeling the explanation process in a CO context.** We conceive an original framework that models the explanation process in a CO context, specifically applied to our WSRP use case. This framework leverages the concept of neighborhood to provide explanations on a wide range of topics. Furthermore, our framework is flexible enough to handle contrastive, scenario and counterfactual explanations depending on the preferences of the end-users. This contrasts with most previously published works which focused on explaining a very limited number of aspects of the solution of a CO problem and could provide a single type of explanation. Finally, it interprets each textual question asked by the end-users in CO terms, reformulating it as a decision problem (or a set of decision problems). This enables us to propose a formal definition of an explanation based on the feasibility of a mathematical program called the foil model. To the best of knowledge, this is the first time such a formal definition is provided in the context of a CO problem.

- **An efficient generation of user-centered explanation texts.** We propose a novel method for generating explanation texts of different types, with a vocabulary adapted to end-users, in the context of our WSRP use case. Our method relies on efficient algorithms for computing a support solution, extracts from it relevant information and populates explanation template texts.
- **An interactive system design.** We introduce an original system design for structuring the interactions between our explanation-generation techniques and the end-users who receive the explanation texts. This system serves as the basis for a GUI prototype demonstrating the practical applicability and potential benefits of our approach.

Our explanation-modeling framework was initially introduced in [LGMO22] while focusing only on contrastive explanations. Moreover, our methods for computing and generating user-centered explanation texts were presented in [LGMO22, LGMO23] for the case of contrastive and counterfactual explanations. The material presented in this manuscript thus broadens these works by introducing a modeling framework and explanation generation methods able to deal with contrastive, scenario, and counterfactual explanations.

1.3 Manuscript structure

The remainder of this manuscript is organized as follows.

Chapter 2 reviews related literature about explanations in various fields from social sciences to CO. We first present a few theoretical elements and principles about explanations from the perspective of social sciences. We then discuss works dealing with explanations in AI, in OR and more specifically in CO. This enable us to clarify where this thesis stands relatively to the explanation-related literature.

Chapter 3 specifies the Workforce Scheduling and Routing Problem use case for which we aim at developing a method to explain its results. We detail the assumptions and the characteristics of this use case. We then provide an ILP formulation of it. Finally, we introduce various notions related to the WSRP. We focus in particular on solution neighborhoods and their related solution transformations as these notions will play a key role in our explanation-modeling framework.

Chapter 4 presents our framework for modeling the explanation process. We first describe what end-user observations are about and how questions can be formulated based on such observations. We then translate these questions expressed in common language into mathematical terms involving mathematical programs and end up providing a definition of explanations relying on the feasibility of such programs.

Chapter 5 deals with our methods and algorithms for computing and generating explanation texts. We describe them for each of the three types of explanations: contrastive, scenario and counterfactual explanations. Numerical experiments are performed in order to evaluate the time efficiency of our generation techniques on large-scale instances and solutions.

Chapter 6 describes our system for structuring the interactions between the end-users and the methods that we develop for generating explanation texts. We first detail its general design, how it We then present the graphic user interface integrating this system that we have implemented as a proof of concept.

Chapter 7 finally sums up the contributions of this manuscript and suggests perspectives.

Chapter 2 Literature related to explanations

2.1 Introduction

Before being a subject of research in the computer science fields such as Artificial Intelligence (AI), Operations Research (OR) or Combinatorial Optimization (CO), "explanation" is above all a common language term and a concept familiar to all of us. But what is exactly an explanation? What contributes to the quality of an explanation? What are the underlying processes of explanations? In the light of these questions, we believe it is appropriate to clarify what is commonly meant by the term "explanation", especially if we aim in this thesis at designing effective explanation methods. Thus, before delving into the computer science world, we propose to take a brief glance at what social sciences have learned, modeled and concluded about the way humans produce, formulate and perceive explanations.

This chapter provides an overview of the existing literature on explanations across various fields, spanning from social sciences to CO. This overview then helps us to position the thesis with respect to this existing literature.

The remainder of the chapter is organized as follows. Section 2.2 presents some insights about explanations from the perspective of social sciences and then sets out some guidelines to help us in the development of methods for explaining results in a CO context. Section 2.3 deals with the literature about explanations in AI and identifies key characteristics of explanations in this domain. Section 2.4 reviews works related to explanations in the specific fields of OR including CO. Section 2.5 concludes this chapter and positions this thesis with respect to the existing literature on explanations.

2.2 A few insights about explanations from social sciences

This section provides some insights about explanations from the perspective of social sciences. We first present a definition of an explanation and then detail some of its essential characteristics. We finally outline a few conclusions to guide us towards our goal in this thesis, namely developing methods for explaining solutions of a given CO problem, a Workforce Scheduling and Routing Problem (WSRP) use case, to the end-users of a system solving this problem.

Definition of an explanation. As described by Miller in [Mil19], the notion of *explanation* refers to a product but also to processes involving an explainer and an explainee.

1. From the perspective of the explanation giver, an explanation is first and foremost a *diagnosis of causality*, in other words, a process of determining the causes behind the fact about which the explanation-receiver is wondering and investigating.
2. From the perspective of the explanation receiver, an explanation is rather the *product* of this diagnosis that the giver communicates to them.
3. Finally, explanations are also a *process of knowledge transfer* between the giver and receiver.

In this thesis, we employ the term "explanation" to refer essentially to the product. Whenever our focus shifts towards the processes of causality diagnosis and knowledge transfer, we name them as such, and not as explanation.

In relation with the above definition of explanations, we propose in the following paragraphs to describe some essential characteristics of explanations from which we will draw some guidelines for the development of our explanation methods in the next subsection.

Explanations are part of a conversation. As a process of knowledge transfer, explanations occur within a *conversation i.e.* a social interaction involving speakers making several contributions.

As a consequence, explanations must follow the rules that guarantee the quality of a conversation. According to Grice in [Gri75], speakers involved in a conversation must adhere to a *cooperative principle*, *i.e.* they must make contributions that are appropriate to the conversation, its context and the other speakers. To meet this principle, Grice lists four maxims about the speakers' contributions.

- *Quality*: "Make sure that the information is of high quality. Do not say things that you believe to be false; do not say things for which you do not have sufficient evidence."
- *Quantity*: "Provide the right quantity of information. Make your contribution as informative as is required; do not make it more informative than is required."
- *Relation*: "Only provide information that is related to the conversation. Be relevant."
- *Manner*: "Be perspicuous. Avoid obscurity of expression; avoid ambiguity; be brief (avoid unnecessary prolixity); be orderly."

Relatively to our goal of designing explanation methods for end-users in a CO context, providing explanations that satisfy the maxim of quality seems quite natural and even necessary. However, the maxims of quantity, relation and manner may demand more careful consideration to be met. Indeed, it may be challenging to *i)* understand the issues about which end-users expect explanations (relation), *ii)* find the right compromise between a long exhaustive explanation text, detailing point by point very specific arguments, and a short explanation text, involving abstract argument (quantity), and *iii)* use an adapted vocabulary (manner).

Explanations are selective. As diagnosis of causality and products of this diagnosis, explanations are *selective*: people rarely expect an explanation that consists of a complete causality diagnostic [Mil19].

This selectivity characteristic has also to do with Grice's conversation maxims of quantity and manner: "Provide the right quantity of information" and "be brief".

Explanations are mostly contrastive. As products and process of knowledge transfer, explanations are often *contrastive* [Lip90, Hil90]: they justify something in contrast to something else. In other words, explanations often answer questions, also said contrastive, that have the following form "Why this event rather than that other event?". In this contrastive question, "this event" is called the *fact*, while "that other event" is called the *foil*.

Building upon the theoretical elements that we introduced above, we can now formulate some guidelines to direct us in our aim of designing methods for explaining outcomes of a CO problem to the end-users of a system solving such a problem.

Guidelines for the development of our explanation methods. Suppose that end-users are wondering about a fact observed in a solution and would like to obtain explanations about it. Below are some guidelines that we, explanation designers, can follow while developing our methods.

(G1) Given a fact observed in a solution, there may be many causes contributing to it so that presenting all of these causes to the end-users may be impractical. In such situations, in accordance with the selectivity characteristic of explanations, as well as the conversation maxims of quantity and relation, we must identify and communicate to end-users a limited number of causes so that it remains concise and graspable for end-users. In addition, in order for end-users to understand our explanations, we must pay attention not to use too technical concepts which could be obscure for them. In other words, we must produce explanations that are *adapted* to the end-users in terms of size and vocabulary.

(G2) Since research in social sciences shows that people usually give and receive explanations that are *contrastive*, we must consider such an explanation type, *i.e.* explaining the fact observed by end-users not *per se* but relatively to some other outcome (foil) that did not occur.

(G3) Since explanations are a part of a *conversation i.e.* an *interaction*, we seek to enable end-users to interact with our explanation methods and request, if needed, not just one but several consecutive explanations, as long as they feel the need to better understand the fact they observed in the solution.

After clarifying the meaning of the common term "explanation" from the perspective of social sciences and after establishing some guidelines for the design of explanation methods, our focus can now turn towards examining explanations as a research subject in the domains of AI and OR.

2.3 Explanations in Artificial Intelligence

As mentioned in Section 1.1, in the past decade, there has been a rising trend of contributions in explainable AI (XAI) [BADRDS$^+$20]. XAI methods from these contributions can be described according to several key characteristics, as shown in Figure 2.1: *i)* the target audience, *ii)* the scope, *iii)* the type, *iv)* the trigger, and *v)* the form of the explanations produced by the methods. In the following paragraphs, we discuss these characteristics in a cross-disciplinary way, by referring to works about explanations from various AI fields, namely Expert Systems (ES), AI Planning (AIP), Machine Learning (ML) and Constraint Satisfaction Problems (CSP).

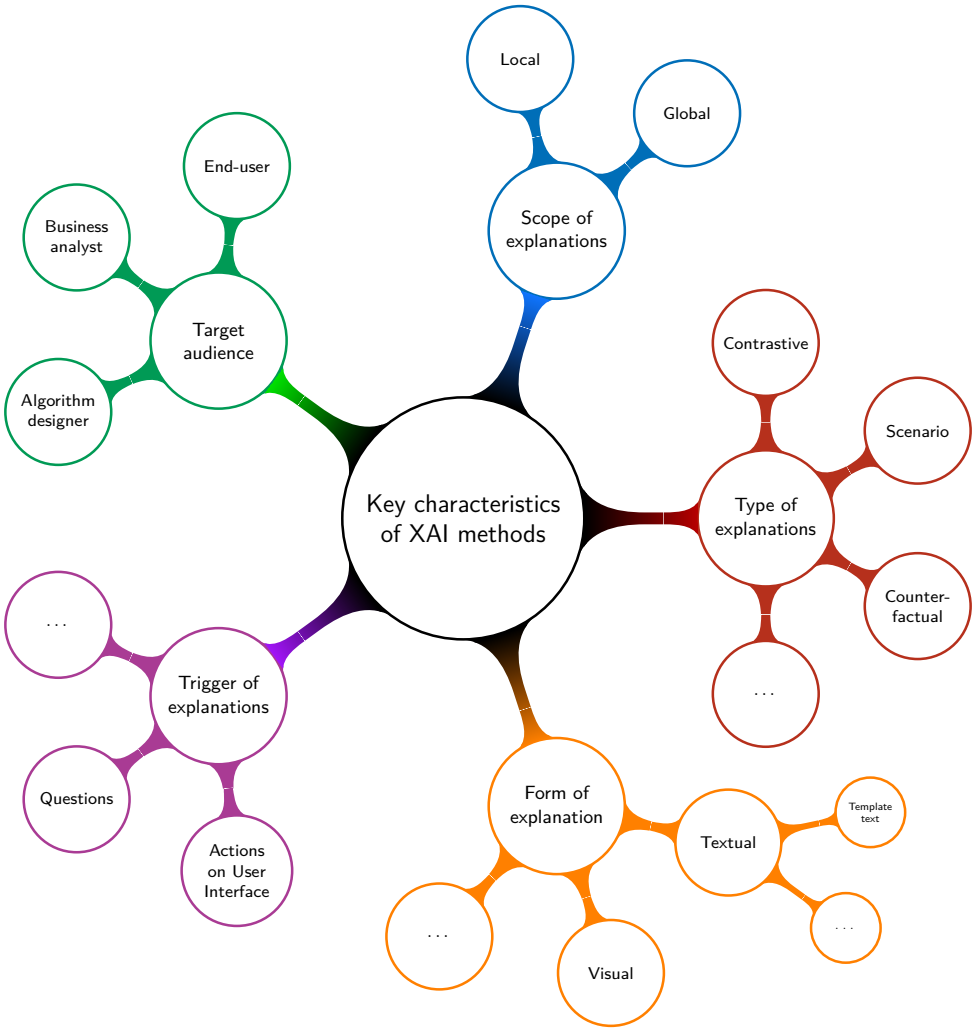


Figure 2.1: Key characteristics of XAI methods

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 [BADRDS+20, MZR21]. Whatever the considered AI field, the literature generally identifies three target audiences: in ES [WT92], 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 [CSK20], 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 ML [MZR21], 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 analog to the end-user in ES and AIP or the AI novice in ML. In DecisionBrain's context (Section 1.1), such an audience corresponds, within their clients' organizations, to the *planners*, *i.e.* people in charge of designing human resources' plannings thanks to 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 according to their *scope*, also known as *focus* [WT92] or *interpretation scale* [MZR21]. The XAI community generally identifies two scopes of explanations, namely local and global scopes. Originally, in the context of Expert Systems (ES), Wick and Thompson [WT92] partition explanations into two groups. The first is made of *process-related explanations*, which involve information on the way 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 *e.g.* [DVK17, GMR$^+$18, LGM20, MZR21], authors rather refer to the first group as *global explanations* and to the second one as *local explanations*. As mentioned by Mohseni *et al.* in [MZR21], local explanations better suit AI novices than global ones as they are less overwhelming.

Since the target audience in our work is an end-user, we design local explanations: our explanations focus on solutions and aim at describing causes justifying some intriguing topics that can be 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 types of explanations based on characteristics of their corresponding questions, or vice versa. Especially, works like [MZR21] name types of explanations 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 type of explanation may be suitable for answering questions corresponding to different interrogative forms as noted in [LGM20]. However, it is a convenient and intuitive way to name, compare and delineate different types of explanations. For this reason, we use it in what follows to name types of explanations.

Types of explanations. As previously mentioned, there are many interrogative forms for the questions to be answered, and consequently many types of explanations. Among them, two types are widely used in the XAI literature: "why-not" explanations, also known as *contrastive* explanations, and "how-to" explanations, also known as *counterfactual* explanations. We are also interested in "what-if" explanations which occur in fewer XAI works. We call this last type of explanations: *scenario* explanations.

- **Contrastive / "why-not" explanations.** Lipton [Lip90] 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 making 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 in [Mil19], some authors refer to the foil as the *counterfactual case* which makes sense as it is literally a counter-fact *i.e.* 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 next bullet point). For this reason, in this work, we adopt the term foil.
  Explanations answering contrastive questions are usually called *contrastive explanations*. In [Mil19, Mil21], Miller 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 [SSK18, CCK$^+$19, Lin20] or in OR [ČLMT19, KSB21].
- **Counterfactual / "how-to" explanations.** Consider an input and its corresponding output obtained thanks to an AI system. A *counterfactual* explanation determines and presents a hypothetical alternative input that would have resulted in a different output such as a user-specified output [WMR18, MZR21]. 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?") - 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 [KSB21].
- **Scenario / "what-if" explanations.** Consider again an input of an AI system. A what-if explanation, or *scenario* explanation, involves a demonstration of how changes in the input affect the output returned by the AI system [Lin20, MZR21]. According to the etymological meaning of the term "counterfactual", a scenario explanation is also a counterfactual explanation, since it considers a hypothetical alternative input data, in other words a counter-input data, and then looks at the consequences on the output data. However, the distinction we draw between a counterfactual and a scenario explanation lies in the responsibility for identifying the hypothetical alternative input. In the former, it falls upon the explanation giver to determine the hypothetical alternative input, while in the latter it is left to the explanation receiver to define it.

In this thesis, we design original methods for modeling and generating contrastive, scenario or counterfactual explanations. Considering contrastive explanations as part of the types of explanations that we aim to produce is in line with guideline (G2), which promotes the contrastive type of explanations (see Section 2.2).

Trigger of explanations. Many works in eXplainable AI (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 *e.g.* [KSB21]. Other works take into account interactions with the target audience and specify ways to trigger explanations. For instance, [LKS18, ČKL+20] 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 topics 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 [SS87, CTJ89, CCK+19, Lin20].

In our work, we choose to trigger explanations with end-user questions, which allows to generate explanations on many aspects of the solution. This is a first step towards guideline (G3) which relates to the conversational and interactive nature of explanations (see Section 2.2).

Form of explanations. There are various ways to present the explanations to the audience [MZR21]. 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 *e.g.* [ZF14, SVZ14] use a saliency heatmap to emphasize important features in the input image; in Artificial Intelligence Planning (AIP), in [KKM+21], Krarup *et al.* 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 [SF96, LKS18]. Another approach is to use Natural Language Generation techniques to automate the verbalization of explanations [FSPC18, POP21].

This thesis deals with textual explanations that are expressed thanks to template texts and does not consider visual explanations.

In this section, we discussed several key ingredients related to the design of an explanation method in the broad field of AI. In the next one, we study how the notion of explanation has been addressed in the specific field of Operations Research.

2.4 Explanations in Operations Research

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

Explanations in Constraint Satisfaction Problems. There are several works on explanations in the field of CSP. In most of them, *e.g.* [dK86, Gin93, Jun04, CJ06], the term explanation (or equivalently nogood, 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, *e.g.* [Jun04], look for the minimal conflict sets; others, *e.g.* [dK86, Gin93, CJ06], 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 non-expert end-users, hence few works consider expressing explanations in a way that is adapted to this audience. Among them, [SF96, BGG21] focus for instance on explaining how to solve, step by step, the given CSP instance thanks to verbal or visual elements while [JO01] seek to explain infeasibility by naming conflict sets of constraints. In any case, in these works, there is a single topic to explain to the audience: the feasibility / infeasibility of the instance.

However, in our Workforce Scheduling and Routing Problem (WSRP) context, the end-user is assumed to have on hand a feasible solution and to look for explanations about other topics, *e.g.* "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 Combinatorial Optimization contexts. Beside these works dealing with explanations in CSP, there are some other OR-related works on user-centered explanations *e.g.* [LKS18, ČLMT19, KSB21].

Ludwig *et al.* [LKS18] focus on a specific heuristic scheduling system that is based on a greedy algorithm and present a facility that is able to provide to an end-user of this system a verbal explanation to 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 in describing part of the system execution on an instance, which supposes that the end-user understands and agrees that the scheduling problem is solved according to the heuristic approach of the system and not another one. In our context, we would like the explanations to be independent from the algorithmic approach used to solve the WSRP. Most often, the functioning of a WSRP-solving algorithm is not greedy so that detailing its steps may be too technical and overwhelming for end-users. Moreover, such a solving algorithm is likely to be updated over time, which would make our explanation techniques obsolete.

Čyras *et al.* [ČLMT19] 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. However, these argumentation frameworks relies on the implicit assumption that the problem can be formulated as a mathematical program involving exclusively binary variables. 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 3.2.2.

Korikov *et al.* [KSB21] 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 3.2.2.

In summary, these existing works have specific application limitations, making it challenging to employ the explanation methods they developed for other CO problems such as the WSRP. Thus, there is a need to propose new explanation approaches adapted to our WSRP context, and potentially applicable to other CO problems.

2.5 Conclusion

In this chapter, we reviewed the concept of explanation across various fields spanning from social sciences to CO. Such a review was essential for our upcoming aim of designing efficient methods explaining solutions of a CO problem, a WSRP use case, to the end-users of systems solving such a problem.

We initially acknowledged that the notion of explanation extends beyond the field of computer sciences, since "explanation" is first and foremost a term from everyday language. This led us to study a few insights about explanations learned from works in social sciences. We recognized the dual nature of explanations as both a process involving a giver and receiver and as a product of that process. Noticing that explanations are part of a conversation, we emphasized the need for explanation designers to adhere to the cooperative principle by following the maxims of quality, quantity, relation and manner as regards the content of our explanations. We acknowledged the inherent selectivity and prevalent contrastive nature of explanations. This helped us in defining a set of guidelines for our work.

We then transitioned our review of the concept of explanations into the general field of AI. We examined key characteristics of explanation methods in AI, including the target audience, scope, types, triggers, and forms of explanations, which helped us to position our work relatively to this literature. In addition, we focused on works related to explanation in OR, in order to analyze how explanations have been addressed in this domain, especially in CO, and revealed the need for methods that go beyond the limitations of the current approaches.

In conclusion, the positioning of this thesis is the following one. The main target audience of our explanations consists of the end-users of an CO system that solves our WSRP use case. Our explanations are local by focusing on a given solution of a WSRP instance. They can be of three types: either contrastive, scenario or counterfactual. They take the form of texts built thanks to templates. They are triggered by questions, also formulated thanks to template texts, which allows us to consider various topics to explain on a given WSRP solution. This positioning is summarized in Figure 2.2, which is based on Figure 2.1 where characteristics that are out of the scope of this thesis are faded.

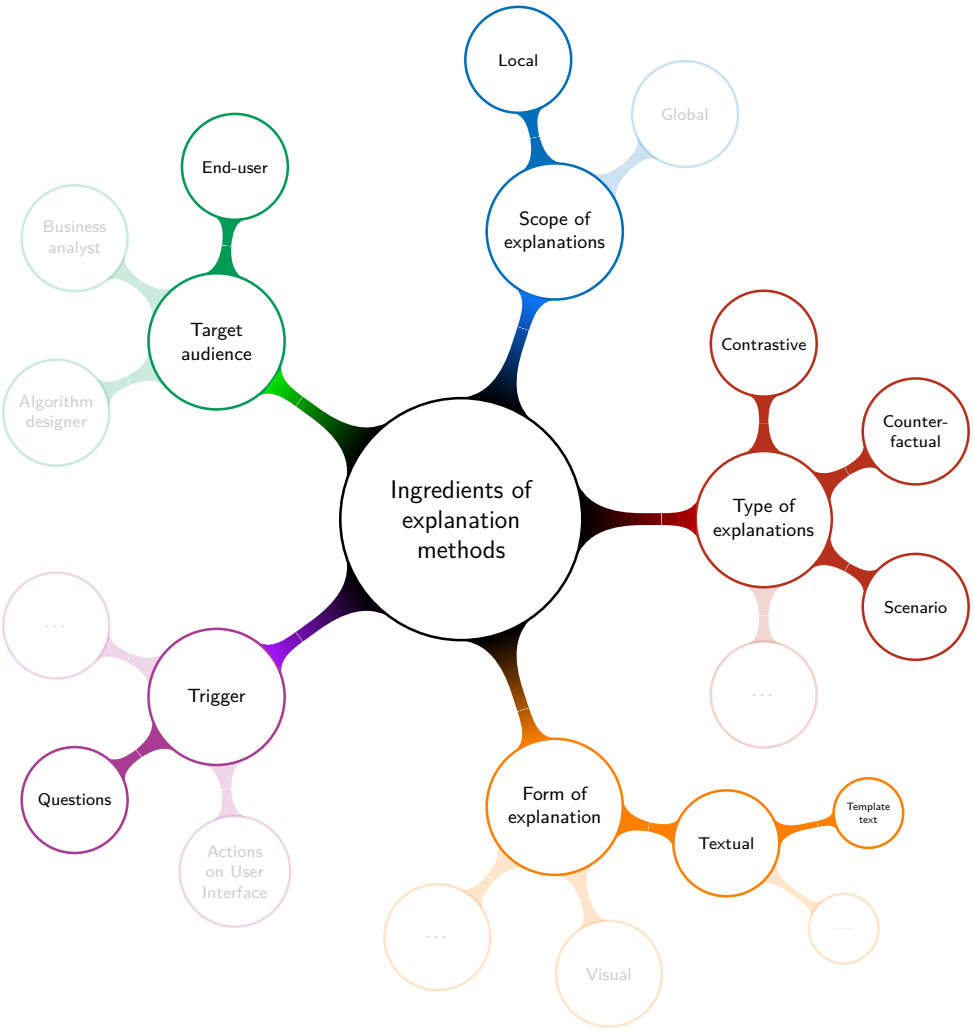


Figure 2.2: Positioning of the thesis in relation to key characteristics of XAI methods

Chapter 3 Background on the Workforce Scheduling and Routing Problem (WSRP)

3.1 Introduction

This chapter deals with the *Workforce Scheduling and Routing Problem* (*WSRP*) for which we aim at developing an approach for explaining its solutions to the end-users of a WSRP-solving system. The WSRP can be stated as follows. Given a set of mobile employees and a set of geographically dispersed tasks, the problem consists in building and assigning to each employee a pair of route and schedule which defines the tasks that 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.

In Section 1.1 we outlined that the WSRP arises in various contexts, leading to multiple variants of the problem. For a comprehensive literature review about the WSRP, we refer the reader to *e.g.* [CSLSQ16]. Consequently, in this chapter, we delineate the specific characteristics of our WSRP use case. This includes specifying the contents of instances and solutions, the objective functions to optimize and the constraints to satisfy. We also introduce a mathematical formulation of this use case in the form of a bi-objective Integer Linear Programming (ILP) model which we refer to as *main model*.

As mentioned in Section 1.2, an essential feature of our general explanation approach is the use of solution *neighborhoods*. In our explanation-modeling framework described in Chapter 4, neighborhoods are the bedrock of the process leading from end-user observations and questions about solutions to explanations. In our algorithms designed for computing and generating explanations presented in Chapter 5, neighborhoods are explored and analyzed in order to find explanatory information to provide to the end-users. This concept of neighborhoods is drawn from the Local Search (LS) literature where a neighborhood of a given solution is the set of all potential other solutions obtained by applying a transformation on this solution. For instance, in our WSRP context, a transformation can be inserting a task within an employee planning, exchanging a task performed by an employee with another task not performed by this employee or reordering tasks within an employee planning, among others. Thus, in this chapter, in addition to specifying our WSRP use case, we expound on various neighborhoods of WSRP solutions that will serve later in Chapters 4 and 5 for both modeling and computing explanations. We introduce various categories of transformations as well as general notions such as *route-equality*, *Backward Earliest start Time* (*BET*), *Forward Latest start Time* (*FLT*) and *feasibility gap*, which will be used throughout this manuscript.

The remainder of this chapter is organized as follows. Subsection 3.2.1 specifies our WSRP use case by detailing its general characteristics and providing a bi-objective ILP formulation of it. Section 3.3 deals with solution neighborhoods.

3.2 Definition of our WSRP use case

3.2.1 General characteristics

In this subsection, we specify general characteristics of our WSRP use case. We first detail the contents of instances and solutions and then define two criteria use for comparing solutions.

Instance. In our use case, we consider a scheduling horizon of one day *i.e.* 1440 minutes. Times are expressed in minutes from 12:00AM (*e.g.* 8:00AM $\equiv$ 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 $ske_i \in \mathbb{N}$, a home location and a working time window $[lbe_i, ube_i] \subseteq [0, 1440]$. Each task $j \in \mathcal{T}$ is characterized by a required skill level $skt_j \in \mathbb{N}$, a location, a performing duration $dt_j \in \mathbb{N}$ and an availability time window $[lbt_j, ubt_j] \subseteq [0, 1440]$.

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 d_i and r_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 $\mathcal{A}_i = \mathcal{T} \cup \{d_i, r_i\}$.

Finally, assuming that all employees travel at the same speed, we note $tr_{jk} \in \mathbb{N}$ 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 pair $(\mathcal{R}_i, \mathcal{C}_i)$. The *route* $\mathcal{R}_i$ is a sequence of activities of $\mathcal{A}_i$ that starts with d_i and ends with r_i. It defines the tour done by i. The *schedule* $\mathcal{C}_i$ is a sequence of *start times* $st_j \in \mathbb{N}$ that defines when i starts performing each activity j in $\mathcal{R}_i$. In other words, let $p \in \mathbb{N}$ be the number of tasks performed by i, then $\mathcal{R}_i = (d_i,\ j_1,\ j_2,\ \ldots,\ j_p,\ r_i)$ and $\mathcal{C}_i = (st_{d_i},\ st_{j_1},\ st_{j_2},\ \ldots,\ st_{j_p},\ st_{r_i})$ - where we do not indicate that $j_1,\ j_2,\ \ldots,\ j_p$ depend on i for readability purpose. Such a route-schedule pair is what we call a *planning*.

Figure 3.1 below is based on Figure 1.1 from Section 1.1. It represents a solution of a WSRP instance. We recall that each employee is related to a color. The graph on the left represents routes: colored dots correspond to tasks performed by employees and gray ones to non-performed tasks, while squares correspond to employee starting locations, which we call home for convenience. The Gantt chart on the right depicts schedules: high and colored rectangles represent tasks; small and 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.

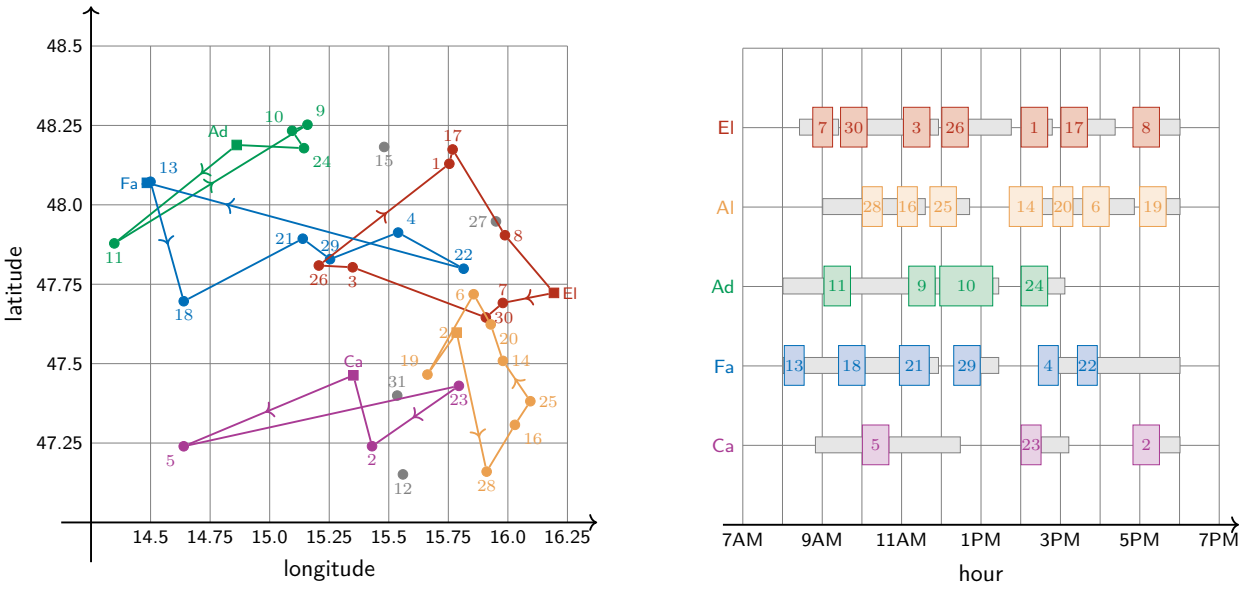


Figure 3.1: Routes (left) and schedules (right) of a solution of a WSRP instance.

Throughout this document, we will often use the solution represented in Figure 3.1 as a small illustrative example. The instance associated with this solution involves 5 employees and 31 tasks, whose data are detailed in Table A.1 of Appendix A.1. The routes and schedules of this solution are described in Table A.2 in Appendix A.2. For instance, the planning $(\mathcal{R}_1, \mathcal{C}_1)$ of employee 1, named Ellen, is given by $\mathcal{R}_1 = (d_1,\ 7,\ 30,\ 3,\ 26,\ 1,\ 17,\ 8,\ r_1)$ and $\mathcal{C}_1 = (504,\ 525,\ 567,\ 662,\ 720,\ 840,\ 900,\ 1009,\ 1080)$ and the planning $(\mathcal{R}_4, \mathcal{C}_4)$ of employee 4, named Fabian, is given by $\mathcal{R}_4 = (d_4,\ 13,\ 18,\ 21,\ 29,\ 4,\ 22,\ r_4)$ and $\mathcal{C}_4 = (480,\ 482,\ 564,\ 656,\ 738,\ 866,\ 925,\ 1080)$.

Total working and travel times. All solutions are not equally good. 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 $twt(\mathcal{S})$ and $trt(\mathcal{S})$, and computed as follows:

$$twt(\mathcal{S}) = \sum_{i \in \mathcal{E}} \sum_{j \in \mathcal{T} \cap \mathcal{R}_i} dt_j \qquad \text{and} \qquad trt(\mathcal{S}) = \sum_{i \in \mathcal{E}} \sum_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} tr_{jk}.$$

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 $(twt(\mathcal{S}), -trt(\mathcal{S})) \geq (twt(\mathcal{S}'), -trt(\mathcal{S}'))$ with a lexicographic order.

Now that we have defined the content of both instances and solutions for our WSRP use case, as well as a bi-criterion on the solutions, we can develop in the next section a mathematical programming model of this WSRP.

3.2.2 Integer Linear Programming (ILP) model

In this subsection, we provide a mathematical formulation of our WSRP use case in the form of a bi-objective Integer Linear Programming (ILP) model, which we call *main model*. Given an instance $\mathcal{I}$, the main model related to $\mathcal{I}$ is noted $mm(\mathcal{I})$ and is presented in Model 1. Its decision variables, objective function, constraints are detailed below.

$$lex\ max \quad \Big(\sum_{i\in\mathcal{E}} \sum_{j\in\mathcal{T}} \sum_{k\in\mathcal{A}_i,\ k\neq d_i,j} U_{ijk}\ dt_j, \quad -\sum_{i\in\mathcal{E}} \sum_{j\in\mathcal{A}_i,\ j\neq r_i} \sum_{k\in\mathcal{A}_i,\ k\neq d_i,j} U_{ijk}\ tr_{jk} \Big) \tag{M1.1}$$

$s.t.$

$$\sum_{k\in\mathcal{A}_i,\ k\neq d_i} U_{i,d_i,k} = 1 \qquad \forall\ i\in\mathcal{E} \tag{M1.2}$$

$$\sum_{j\in\mathcal{A}_i,\ j\neq r_i} U_{i,j,r_i} = 1 \qquad \forall\ i\in\mathcal{E} \tag{M1.3}$$

$$\sum_{j\in\mathcal{A}_i,\ j\neq k,r_i} U_{ijk} = \sum_{j'\in\mathcal{A}_i,\ j'\neq d_i,k} U_{ikj'} \qquad \forall\ i\in\mathcal{E},\ \forall\ k\in\mathcal{T} \tag{M1.4}$$

$$\sum_{k\in\mathcal{A}_i,\ k\neq d_i,j} U_{ijk} \leq \mathbb{1}_{\{skt_j\leq ske_i\}} \qquad \forall\ i\in\mathcal{E},\ \forall\ j\in\mathcal{T} \tag{M1.5}$$

$$\sum_{i\in\mathcal{E}} \sum_{k\in\mathcal{A}_i,\ k\neq d_i,j} U_{ijk} \leq 1 \qquad \forall\ j\in\mathcal{T} \tag{M1.6}$$

$$\sum_{i\in\mathcal{E}} \sum_{k\in\mathcal{A}_i,\ k\neq d_i,j} U_{ijk}\ lbt_j \leq T_j \qquad \forall\ j\in\mathcal{T} \tag{M1.7}$$

$$T_j \leq \sum_{i\in\mathcal{E}} \sum_{k\in\mathcal{A}_i,\ k\neq d_i,j} U_{ijk}\ (ubt_j - dt_j) \qquad \forall\ j\in\mathcal{T} \tag{M1.8}$$

$$\sum_{i\in\mathcal{E}} U_{i,d_i,k}(lbe_i + tr_{d_ik}) \leq T_k \qquad \forall\ k\in\mathcal{T} \tag{M1.9}$$

$$T_j + dt_j + \sum_{i\in\mathcal{E}} U_{ijk}\ tr_{jk} \leq T_k + \Big(1 - \sum_{i\in\mathcal{E}} U_{ijk}\Big) ubt_j \qquad \forall\ (j,k)\in\mathcal{T}^2,\ j\neq k \tag{M1.10}$$

$$T_j + dt_j \leq \sum_{i\in\mathcal{E}} U_{i,j,r_i}(ube_i - tr_{jr_i}) + \Big(1 - \sum_{i\in\mathcal{E}} U_{i,j,r_i}\Big) ubt_j \qquad \forall\ j\in\mathcal{T} \tag{M1.11}$$

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

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

Model 1: Main model $mm(\mathcal{I})$ related to a WSRP instance $\mathcal{I}$.

Decision variables. Two sets of decision variables are used in $mm(\mathcal{I})$. The first set is related to the temporal dimension of the WSRP, we call them *start time decision variables*. For each task $j\in\mathcal{T}$, the start time decision variable T_j is an integer decision variable defining 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, we call them *path decision variables*. For each employee $i\in\mathcal{E}$ and each pair of activities $(j,k)\in(\mathcal{A}_i\setminus\{r_i\})\times(\mathcal{A}_i\setminus\{d_i,j\})$, the path decision variable U_{ijk} is a binary decision variable which is equal to 1 if i performs activity j and then moves to activity k, and to 0 otherwise. Note that, thanks to such binary variables, we can build expressions to track whether task j is performed by employee i and whether task j is performed by any employee, respectively as follows:

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

These expressions are involved several times in the objective function and the constraints of $mm(\mathcal{I})$.

Bi-objective function. In $mm(\mathcal{I})$, a bi-objective function (M1.1) is maximized according to a lexicographic order: the first objective corresponds the total working time and the second one to the opposite of the total travel time.

Constraints. The constraints of $mm(\mathcal{I})$ are described below by groups.

- *Flow constraints* (M1.2) to (M1.4) ensure that each employee starts their working day from their home location, goes from one activity to the next, without splitting into multiple directions, and ends their day at their home location. Note that these constraints alone do not prevent subtours, *i.e.* loops connecting only tasks, assigned to employees as part of their routes. However, sequencing constraints defined below will prohibit such sub-tours.
- *Skill constraints* (M1.5) ensure that an employee $i \in \mathcal{E}$ can be assigned to a task $j \in \mathcal{T}$ only if their skill level ske_i is higher than the minimum skill level skt_j required for performing j.
- *Occurrence constraints* (M1.6) guarantee that each task $j \in \mathcal{T}$ is performed at most once, *i.e.* occurs at most once within all employee routes.
- *Availability constraints* (M1.7) and (M1.8) ensure that, if a task $j \in \mathcal{T}$ is performed, then it must be started and ended within its availability time-window $[lbt_j,\ ubt_j]$.
- *Working hours and sequencing constraints* (M1.9) to (M1.11) 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 $[lbe_i,\ ube_i]$ and i must have enough time to travel from j to k, after ending j and before starting k. The constraints (M1.9), (M1.10) and (M1.11) correspond respectively to the following cases: $j = d_i$ and $k \in \mathcal{T}$; $j \in \mathcal{T}$ and $k \in \mathcal{T} \setminus \{j\}$; $j \in \mathcal{T}$ and $k = r_i$.

Throughout this document, we will refer to the group of constraints formed by availability, working hours and sequencing constraints, *i.e.* constraints from (M1.7) to (M1.11), as *time constraints*.

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

Two solution characterizations. In Subsection 3.2.1, we define a solution $\mathcal{S}$ of an instance $\mathcal{I}$ as a *family of employee 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 $mm(\mathcal{I})$, *i.e.* the *assignment* of a value to each decision variable involved in the model. The first characterization rather corresponds to an end-user way of representing a solution, while the second to an optimization-system way.

In most Operations Research papers, such two characterizations are implicitly equated. However, in our context, it is important to clearly distinguish between them. 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 $mm(\mathcal{I})$ decision variables and note it $\mathcal{X}$, while we save the term *solution* for naming a family of employee plannings, noted $\mathcal{S}$.

Bijection between solution characterizations. Consider an instance $\mathcal{I}$ and its corresponding $mm(\mathcal{I})$. Provided that an ILP-solution $\mathcal{X}$ satisfies flow constraints (M1.2) to (M1.4) 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 path decision variables $(U_{ijk})_{ijk}$ and the schedules $(\mathcal{C}_i)_{i\in\mathcal{E}}$ from the values of the temporal variables $(T_j)_{j\in\mathcal{T}}$. Conversely, by following the inverse reasoning, one can build an ILP-solution $\mathcal{X} = \varphi^{-1}(\mathcal{S})$ from a solution $\mathcal{S}$. For more details about φ, see Algorithms C.1 and C.2 in Appendix C.

We illustrate this mapping with the small illustrative example of solution $\mathcal{S}$ represented in Figure 3.1. By applying φ^{-1} on $\mathcal{S}$, we obtain an ILP-solution $\mathcal{X} = \varphi^{-1}(\mathcal{S})$. The full description of $\mathcal{X}$ is given in Table B.1 in Appendix B. We give below the part related to employee 1.

- $T_7 = 525,\ T_{30} = 567,\ T_{26} = 720,\ T_1 = 840,\ T_{17} = 900,\ T_8 = 1009$;
- $U_{1,d_1,7} = U_{1,7,30} = U_{1,30,3} = U_{1,3,26} = U_{1,26,1} = U_{1,1,17} = U_{1,17,8} = U_{1,8,r_1} = 1$ and $U_{1jk} = 0$ for all other pairs of activities $(j,k) \in (\mathcal{A}_1 \setminus \{r_1\}) \times (\mathcal{A}_1 \setminus \{d_1, j\})$.

Feasible solution and feasible planning. We described above the constraints that an ILP-solution must satisfy to be feasible. Thanks to 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 feasible solution $\mathcal{S}$.

In this subsection, given an instance $\mathcal{I}$, we presented the main model $mm(\mathcal{I})$, a bi-objective ILP model, which precisely defined the constraints governing the feasibility of the solutions of $\mathcal{I}$. In the next section, we described various transformations that can be applied to solutions of $\mathcal{I}$ defining various neighborhoods.

3.3 Solution transformations and neighborhoods

Throughout this section, we note $\mathcal{S}$ a feasible solution of a WSRP instance $\mathcal{I}$. As mentioned in Section 3.1, solution *neighborhoods* along with their corresponding *transformations* play a central role in our explanation-modeling framework and explanation-generation methods. This is why we devote a section to detailing various transformations and neighborhoods as well as organizing them into categories. In other words, the purpose of this section is to lay the groundwork for Chapters 4 and 5 which deal with explanation modeling and generation. Before delving into the details of any specific transformation or neighborhood, let us start by providing comprehensive definitions of for each of these two concepts. We also introduce the notion of *route-equality* which is used to define categories of transformations and neighborhoods.

Transformation and neighborhood. A *transformation* tf is an application that applies a series of changes to the routes and/or schedules of $\mathcal{S}$ resulting in a set of other solutions, which we call *neighborhood* and note $\mathcal{N}(tf, \mathcal{S})$. We call *neighbors* or *neighboring solutions* the solutions of $\mathcal{N}(tf, \mathcal{S})$. Besides, we say that a transformation is feasible if there is at least one feasible neighboring solution in $\mathcal{N}(tf, \mathcal{S})$.

Route-equal plannings and solutions. Associated with a route $\mathcal{R}_i$, various schedules can be chosen, which differ from each other in the values given to the starting times of the activities in $\mathcal{R}_i$. 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.

From the above definitions, we can now delineate categories of transformations based on the size of their neighborhoods. Indeed, the size of a neighborhood $\mathcal{N}(tf, \mathcal{S})$ varies depending on its associated transformation tf. For example, given a solution, the neighborhood associated with inserting a task within an employee planning between a *given* pair of consecutive activities is smaller than the one associated with inserting the same task within the same employee planning between *any* pair of consecutive activities. In this work, we measure the size of a neighborhood $\mathcal{N}(tf, \mathcal{S})$ by counting the number of disjoint subsets of $\mathcal{N}(tf, \mathcal{S})$, containing solutions that are all route-equal to each other, and then by comparing this number to instance-related size parameters, namely the numbers of employees (denoted as n) and tasks (denoted as m). We define below three categories of transformations and neighborhoods. These categories will play an important role in the work to be presented in Chapters 4 and 5. Namely, our approach for computing and generating explanation texts will be based on solution transformations, and the techniques used to do so will depend on the category of the involved transformations.

Categories of transformations and neighborhoods. We consider three categories as follows.

- **Constant-size.** We say that a solution transformation tf is *constant-size* and that the neighborhood $\mathcal{N}(tf, \mathcal{S})$ has a *constant-size structure* if all the neighboring solutions of $\mathcal{N}(tf, \mathcal{S})$ are route-equal to each other.
- **Polynomial-size.** We say that a solution transformation tf is *polynomial-size* and that a neighborhood $\mathcal{N}(tf, \mathcal{S})$ has a *polynomial-size structure* if the number of subsets of $\mathcal{N}(tf, \mathcal{S})$ containing solutions that are all route-equal to each other is polynomial (non-constant) in n, number of employees in $\mathcal{E}$, and m, number of tasks in $\mathcal{T}$.
- **Exponential-size.** We say that a solution transformation tf is *exponential-size* and that a neighborhood $\mathcal{N}(tf, \mathcal{S})$ has a *exponential-size structure* if the number of subsets of $\mathcal{N}(tf, \mathcal{S})$ containing solutions that are all route-equal to each other is exponential in n and m.

Note that we plan to use these transformations not as part of an LS algorithm solving a WSRP but as part of a system generating explanations on a given solution of a WSRP. This is why we consider not only constant-size and polynomial-size transformations, but also exponential-size transformations which we would probably not consider in the deign of a LS algorithm due to the size of their corresponding neighborhoods.

In the remainder of this section, for each of these three categories, we will introduce several "practical" transformations which will be later involved in Chapters 4 and 5. However, before detailing them, we will first consider a few *elementary transformations*. As we will see, elementary transformations are actually constant-size, but we isolate them and refer to them as "elementary" because we will mostly use them to define the practical transformations. Furthermore, a crucial aspect for the numerical tractability of our explanation-generation algorithms lies in our ability to efficiently check whether a neighborhood contains at least one feasible solution *i.e.* whether a transformation is feasible. Equally important is the ability to efficiently compute either the best feasible neighboring solution if the transformation is feasible and to identify the reasons of the infeasibility otherwise. Therefore, this section goes beyond providing detailed descriptions of the transformations and neighborhoods within each category. It also delves into efficiency considerations related to the assessment of transformation feasibility and the computation of neighboring solutions.

Thus, this section starts with a preliminary subsection (Subsection 3.3.1) presenting the elementary transformations as well as some technical notions. Subsequent subsections deal with more complex transformations, namely constant-size (Subsection 3.3.2), polynomial-size (Subsection 3.3.3) and exponential-size (Subsection 3.3.4) transformations, that are built upon these elementary transformations. Within each subsection, we also outline efficiency considerations.

3.3.1 Preliminaries

3.3.1.1 Elementary transformations

In this part, we introduce three *elementary transformations*, namely *shifting*, *removing* and *inserting* transformations, which will be used in Subsections 3.3.2 and 3.3.3 to build interesting constant-size and polynomial-size transformations. These elementary transformations are also depicted in Figure 3.2. While the top row of this figure recalls the example solution represented in Figure 3.1 in Subsection 3.2.1 with a focus on Ellen's and Fabian's planning, the three following rows show neighboring solutions obtained by applying one of these elementary transformations.

Elementary shifting transformation. Let $i \in \mathcal{E}$ be an employee. Let tf be the elementary *shifting* transformation. tf consists in shifting start times of the schedule $\mathcal{C}_i$ of i, *i.e.* the neighborhood $\mathcal{N}(tf, \mathcal{S})$ comprises every solution $\mathcal{S}'$ obtained from $\mathcal{S}$ by changing one or several start times in the schedule $\mathcal{C}_i$.

For example, consider the solution $\mathcal{S}$ of the illustrative example represented in Figure 3.1. By shifting start times of Fabian's planning $(\mathcal{R}_4, \mathcal{C}_4)$, a new planning $(\mathcal{R}'_4, \mathcal{C}'_4)$ can be obtained such that $\mathcal{R}'_4 = \mathcal{R}_4 = (d_4,\ 13,\ 18,\ 21,\ 29,\ 4,\ 22,\ r_4)$ and $\mathcal{C}'_4 = (504,\ 506,\ 586,\ 678,\ 738,\ 820,\ 879,\ 1034)$.

Elementary removing transformation. Let $i \in \mathcal{E}$ be an employee and $j \in \mathcal{T}$ be a task performed by employee i in $\mathcal{S}$. Let tf be the elementary *removing* transformation. tf consists in removing j from the planning $(\mathcal{R}_i, \mathcal{C}_i)$ of i, *i.e.* the neighborhood $\mathcal{N}(tf, \mathcal{S})$ comprises every solution $\mathcal{S}'$ obtained from $\mathcal{S}$ by:

i) removing j and st_j respectively from $\mathcal{R}_i$ and $\mathcal{C}_i$ so as to obtain a new route $\mathcal{R}_{i'}$ and a new schedule $\mathcal{C}'_i$;
ii) shifting or keeping unchanged the start times in $\mathcal{C}'_i$;
iii) keeping unchanged the planning of every employee $i' \in \mathcal{E}$ other than i.

For example, consider again the solution $\mathcal{S}$ of the illustrative example. By removing task 18 from Fabian's planning $(\mathcal{R}_4, \mathcal{C}_4)$, a new planning $(\mathcal{R}'_4, \mathcal{C}'_4)$ can be obtained such that $\mathcal{R}'_4 = (d_4,\ 13,\ 21,\ 29,\ 4,\ 22,\ r_4)$ and $\mathcal{C}'_4 = (540,\ 542,\ 678,\ 738,\ 820,\ 879,\ 1034)$.

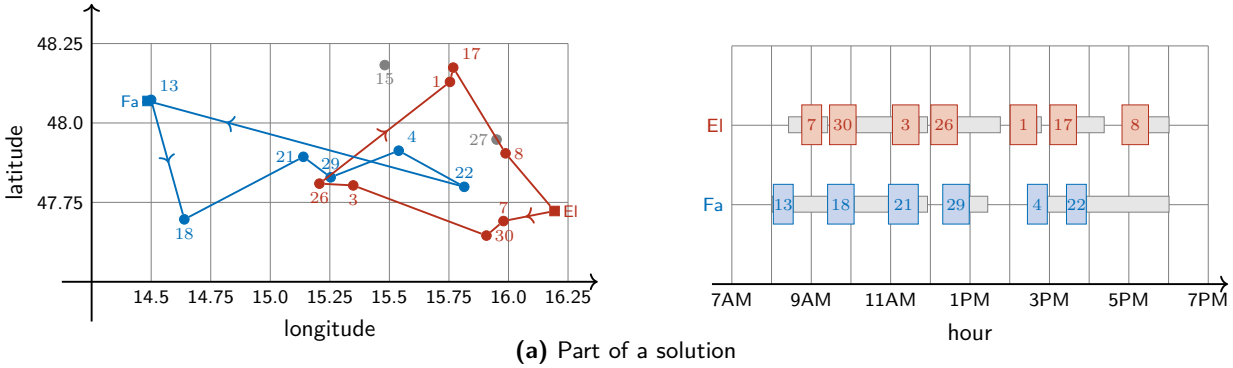
Elementary inserting transformation. Let $i \in \mathcal{E}$ be an employee, $j \in \mathcal{T}$ be a task not performed by any employee in $\mathcal{S}$, and $(k_1, k_2) \in \mathcal{A}_i^2$ be two activities performed consecutively by employee i. Let tf be the elementary *insertion* transformation. tf consists in inserting j in the planning $(\mathcal{R}_i, \mathcal{C}_i)$ of i, between activities k_1 and k_2, *i.e.* the neighborhood $\mathcal{N}(tf, \mathcal{S})$ comprises every solution $\mathcal{S}'$ obtained from $\mathcal{S}$ by:

i) inserting j in $\mathcal{R}_i$ between k_1 and k_2 to obtain $\mathcal{R}'_i$;
ii) inserting st_j in $\mathcal{C}_i$, between st_{k_1} and st_{k_2} to obtain $\mathcal{C}'_i$ and choosing new values for all the start times in $\mathcal{C}'_i$;
iii) keeping unchanged the planning of every employee $i' \in \mathcal{E}$ other than i.

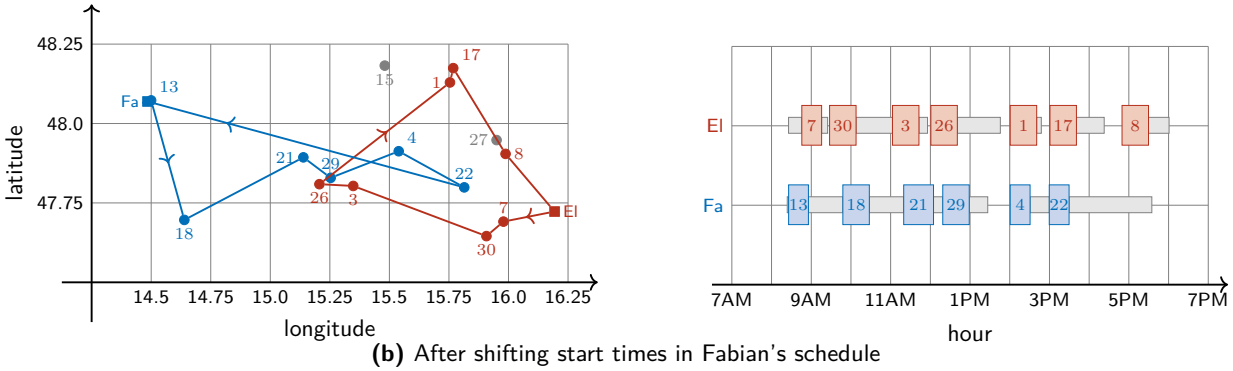
For example, consider again the solution $\mathcal{S}$ of the illustrative example. By inserting task 15 in Ellen's planning $(\mathcal{R}_1, \mathcal{C}_1)$ between task 26 and task 1, a new planning $(\mathcal{R}'_1, \mathcal{C}'_1)$ can be obtained such that $\mathcal{R}'_1 = (d_1,\ 7,\ 30,\ 3,\ 26,\ 15,\ 1,\ 17,\ 8,\ r_1)$ and $\mathcal{C}'_1 = (504,\ 525,\ 567,\ 662,\ 720,\ 816,\ 860,\ 907,\ 1009,\ 1080)$.

Before ending this part introducing the three elementary transformations, let us make some general comments about the feasibility and the effect on the bi-objective function of these transformations. We recall that $\mathcal{S}$ a feasible solution of a WSRP instance $\mathcal{I}$.

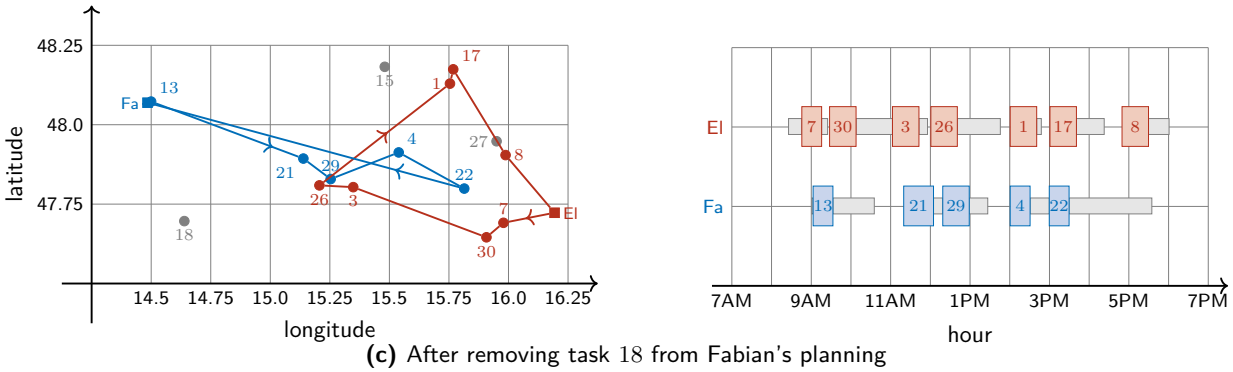
- **Transformation feasibility.**
  - Regarding the elementary removing transformation, consider the neighboring solution $\mathcal{S}'$ obtained by skipping step ii). Since $\mathcal{S}$ is feasible and since steps i) and iii) do not make the solution violate skill, occurrence nor time constraints, (M1.5) to (M1.11) in Model 1 (see Subsection 3.2.2), $\mathcal{S}'$ is also feasible. Therefore, there is always at least one feasible neighboring solution *i.e.* the elementary *removing* transformation is always *feasible*.
  - However, shifting arbitrarily the start times of activities in an employee schedule of $\mathcal{S}$ as it is done in a shifting elementary transformation may provoke violation of time constraints, (M1.7) to (M1.11) in Model 1. Similarly, inserting a task in an employee's route as it is done in an inserting transformation at step i) may provoke violation of skill constraints, (M1.5) in Model 1. Therefore, the elementary *shifting* and *inserting* transformations may be *feasible or not*.
- **Effect on the bi-objective function.** Let tf be an elementary shifting, removing or inserting transformation and suppose that there exists $\mathcal{S}'$ a feasible neighboring solution of $\mathcal{N}(tf, \mathcal{S})$.
  - If tf is a *shifting* transformation, since the routes of $\mathcal{S}$ and $\mathcal{S}'$ are exactly the same, the total working and total travel times of $\mathcal{S}$ equal the ones of $\mathcal{S}'$, *i.e.* $twt(\mathcal{S}') = twt(\mathcal{S})$ and $trt(\mathcal{S}') = trt(\mathcal{S})$, then $\mathcal{S}'$ is *as good as* $\mathcal{S}$.
  - If tf is a *removing* transformation, since all the tasks performed in $\mathcal{S}$ but task j are performed in $\mathcal{S}'$, the total working time of $\mathcal{S}'$ is smaller than the one of $\mathcal{S}$, *i.e.* $twt(\mathcal{S}') < twt(\mathcal{S})$, then $\mathcal{S}'$ is *worst than* $\mathcal{S}$.
  - If tf is a *inserting* transformation, since all the tasks performed in $\mathcal{S}$ plus task j are performed in $\mathcal{S}'$, the total working time of $\mathcal{S}'$ is greater than the one of $\mathcal{S}$, *i.e.* $twt(\mathcal{S}') > twt(\mathcal{S})$, then $\mathcal{S}'$ is *better than* $\mathcal{S}$.



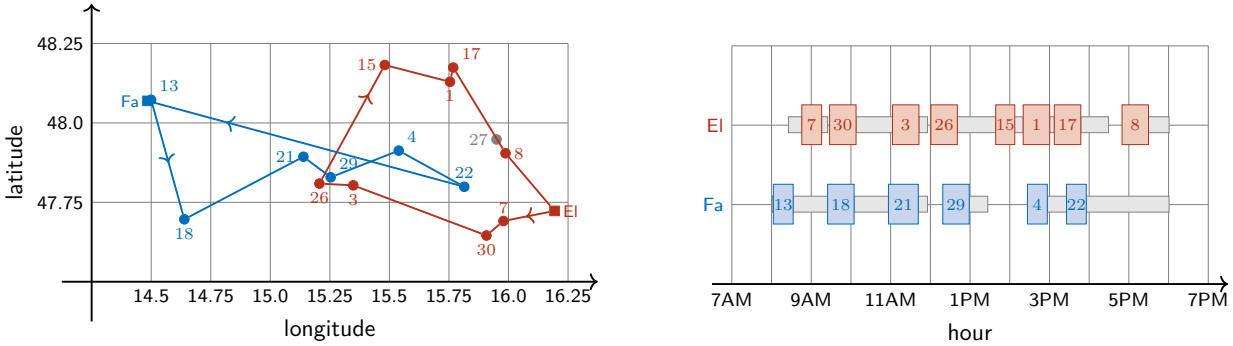
(a) Part of a solution



(b) After shifting start times in Fabian's schedule



(c) After removing task 18 from Fabian's planning



(d) After inserting task 15 in Ellen's planning between task 26 and task 1

Figure 3.2: Neighboring solutions obtained by applying elementary transformations on a solution. The top row represents a part of the solution $\mathcal{S}$, presented in Figure 3.1, focusing on Ellen's and Fabian's plannings. Underneath, the second to fourth rows represent neighboring solutions (with the same focus on Ellen's and Fabian's planning) obtained from $\mathcal{S}$ respectively by applying shifting, removing and inserting elementary transformations.

3.3.1.2 Efficient assessment of elementary transformation feasibility

In this part, we focus on efficient ways to assess the feasibility of elementary transformations and to build feasible neighborhood solutions. Let us first focus specifically on the case of the insertion elementary transformation. We will make general comments about all the elementary transformations at the end of this part.

Using the notations involved in the definition of the elementary insertion transformation, suppose that one seeks to insert task j in the planning $(\mathcal{R}_i, \mathcal{C}_i)$ of employee i, between activity k_1 and activity k_2. A first approach for checking whether this transformation is feasible is to actually insert j in $(\mathcal{R}_i, \mathcal{C}_i)$ and then try to build a new feasible schedule $\mathcal{C}'_i$ for employee i according to an earliest start time policy. We gradually set the start times of $\mathcal{C}'_i$ by following the order of activities in $\mathcal{R}_i$. Given an activity of $\mathcal{R}_i$, we set its start time at the earliest possible time taking into account that *i)* it cannot be earlier than the lower bound of the availability time window of this activity, and *ii)* a travel time is needed between the end of the previous activity and the beginning of this activity. Such an approach is computed in linear time, namely in $\mathcal{O}(p)$ with p the number of activities in $\mathcal{R}_i$ and more loosely in $\mathcal{O}(m)$ with m the number of tasks in $\mathcal{T}$.

However, we can actually devise an approach running in constant time. Within the literature about Local Search (LS) algorithms for solving the Vehicle Routing Problem with Time Windows (VRPTW), Savelsberg in [Sav92] introduces the notion of *Forward Time Slack* (*FTS*) which allows the author to design a constant time computation for checking the feasibility of an insertion. In the following paragraphs, we transpose the notion of FTS from the VRPTW to the WSRP and also define its backward version, namely *Backward Time Slack* (*BTS*); we also define the notions of *Backward Earliest start Time* (*BET*) and *Forward Latest start Time* (*FLT*). We will see then how BET and FLT can be used for checking the feasibility of an elementary inserting transformation in constant time.

Backward and forward time slacks. Given a feasible planning $(\mathcal{R}_i, \mathcal{C}_i)$ and a pair (j, st_j) in $\mathcal{R}_i \times \mathcal{C}_i$, the *Backward Time Slack* (*BTS*), noted b_j, and the *Forward Time Slack* (*FTS*), noted f_j, are defined as the largest amounts of time by which the starting time st_j can be shifted respectively backward and forward without causing the planning $(\mathcal{R}_i, \mathcal{C}_i)$ to violate availability, working hours or sequencing constraints. The BTS of each activity in $(\mathcal{R}_i, \mathcal{C}_i)$ can be computed recursively, in a backward order, as follows:

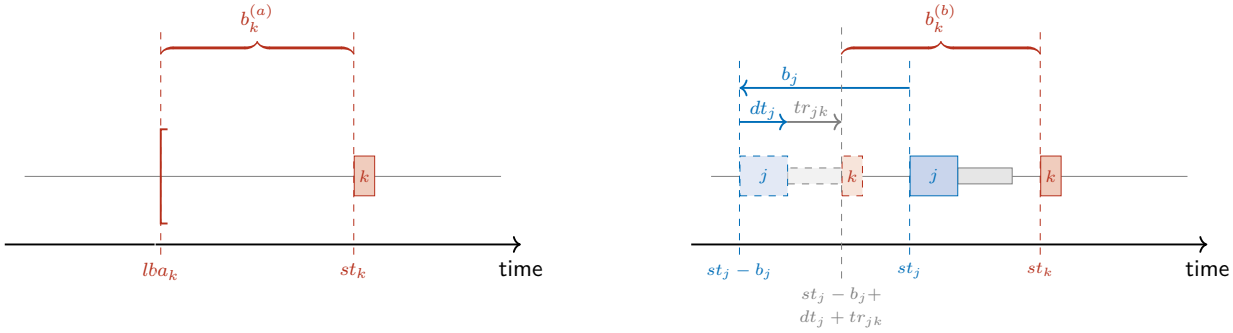
- $b_{d_i} = st_{d_i} - lbe_i$,
- for (j,k) consecutive activities of $\mathcal{R}_i$, $b_k = st_k - max\big(lba_k,\ st_j - b_j + dt_j + tr_{jk}\big)$;

and the FTS of each activity can be computed in a forward order, as follows:

- $f_{r_i} = ube_i - st_{r_i}$,
- for (j,k) consecutive activities of $\mathcal{R}_i$, $f_j = min\big(uba_j,\ st_k + f_k - tr_{jk}\big) - (st_j + dt_j)$;

with $lb_{r_i} = lbe_i$, $ub_{d_i} = ube_i$ and $dt_{d_i} = 0$.

The above formula computing the BTS b_k, given a pair (j,k) of consecutive activities, is depicted in Figure 3.3.



(a) If current start time st_k of activity k has to be changed for an earlier time, then this change is limited by the lower bound of its availability time window $[lba_k,\ uba_k]$, so st_k can not be shifted by more than $b_k^{(a)}$.

(b) If current start time st_k of activity k has to be changed for an earlier time, then this change is limited by the performance of the previous activity j which can not be shifted by more than b_j, so st_k can not be shifted by more than $b_k^{(b)}$.

Figure 3.3: Computation of a Backward Time Slack (BTS). The two figures represents the two limiting cases to consider when computing the BTS b_k of activity k.

As an example, we compute the BTS and FTS for all the performed activities in solution $\mathcal{S}$ represented in Figure 3.1. BTS and FTS values are given in Table A.3 in Appendix A.3. We give below the values associated with Ellen's planning $(\mathcal{R}_1, \mathcal{C}_1)$.

$$\text{BTS: } (b_{d_1} = 5,\ \ b_7 = 0,\ \ b_{30} = 3,\ \ b_3 = 2,\ \ b_{26} = 0,\ \ b_1 = 15,\ b_{17} = 28,\ b_8 = 55,\ b_{r_1} = 55);$$
$$\text{FTS: } (f_{d_1} = 76,\ f_7 = 56,\ f_{30} = 53,\ f_3 = 53,\ f_{26} = 48,\ f_1 = 33,\ f_{17} = 20,\ f_8 = 0,\ \ f_{r_1} = 0).$$

From now on, we assume that a feasible solution $\mathcal{S}$ is systematically associated with corresponding BTS and FTS values. Building upon the notions of BTS and FTS, we can define the ones of Backward Earliest start Time and Forward Latest start Time.

Backward earliest start time and forward latest start time. Consider a feasible planning $(\mathcal{R}_i, \mathcal{C}_i)$ of an employee i, two consecutive activities (k_1, k_2) in $\mathcal{R}_i$ and a task j not in $\mathcal{R}_i$ such that employee i is skilled enough to perform it. We define the notions of *Backward Earliest start Time* (*BET*) noted st_j^b (resp. *Forward Latest start Time* (*FLT*) noted st_j^f) of j as follows. Suppose that one wants to insert j in $\mathcal{R}_i$ between k_1 and k_2. st_j^b is the earliest time (resp. st_j^f 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 which satisfy time constraints (*i.e.* availability, working hours and sequencing constraints). st_j^b and st_j^f are computed as follows:

$$st_j^b = max\,(st_{k_1} - b_{k_1} + dt_{k_1} + tr_{k_1 j},\ lbt_j) \qquad \text{and} \qquad st_j^f = min\,(st_{k_2} + f_{k_2} - tr_{jk_2},\ ubt_j) - dt_j.$$

Using the BET and FLT, we can now propose a constant-time computation for checking whether an elementary insertion transformation is feasible.

Constant-time feasibility check for the elementary inserting transformation. Following on the above-definition of the BET and FLT, suppose that employee i is skilled enough to perform task j. If BTS and FTS of the planning $(\mathcal{R}_i, \mathcal{C}_i)$ are known, then checking the feasibility of inserting j between k_1 and k_2 in the planning of i can be computed in constant time as follows: the insertion is feasible if and only if $st_j^b \leq st_j^f$, given that the BET st_j^b and FLT st_j^f are computed on constant time thanks to the above-formula.

Now that we have this constant-time computation for checking the feasibility of the elementary insertion transformation, we provide below some general comments about computation efficiency related to all elementary transformations.

Efficiency considerations related to elementary transformations. We recall that $\mathcal{S}$ is a feasible solution of an instance $\mathcal{I}$, that m is the number tasks in $\mathcal{T}$.

- **Computing and maintaining consistent BTS and FTS in linear time.** Given $\mathcal{S}$, for each employee planning, computing its corresponding BTS and FTS by applying the recursive formula involved in their definition corresponds to a *linear-time* computation, in $\mathcal{O}(p)$ with p the number of activities in the employee planning, or more loosely in $\mathcal{O}(m)$ with m the number tasks in $\mathcal{T}$. Besides, suppose that one changes the start time of a single activity j from st_j to st'_j in an employee planning. Due to the recursive nature of the BTS and FTS, a *linear-time* computation is also needed to update the BTS and FTS and maintain them consistent with the new employee planning.
- **Checking transformation feasibility in constant time.** While calculating BTS and FTS, as well as maintaining their consistency, requires linear-time computations, the positive counterpart is that checking the feasibility of an elementary transformation can be performed in constant time.
  - Checking whether an *insertion* transformation is feasible can be performed in *constant time* thanks to the above check comparing the BET and FLT.
  - Since the *removing* transformation is always feasible, as mentioned earlier, no feasibility check is needed. In other words, this corresponds to a *constant-time* computation.
  - Regarding the *shifting* transformation, suppose that one wants to change the start time of a single activity j from st_j to st'_j. Checking whether this transformation is feasible can be performed by comparing the difference of start time $st'_j - st_j$ with the values of BTS b_j and FTS f_j: if $st'_j > st_j$ and $st'_j - st_j < f_j$ or if $st_j > st'_j$ and $st_j - st'_j < b_j$, then the transformation is feasible. This corresponds to a *constant-time* computation.
- **Building a neighboring solution in linear time.** Suppose that one wants to build a neighboring solution by applying an elementary transformation whose feasibility has already been checked and confirmed. Regardless of the transformation, at the end of the neighboring solution construction, BTS and FTS must be kept consistent with the new planning which requires to apply the recursive formula. This causes the construction of a neighboring solution related to any of the three elementary transformations to be computed in linear time, $\mathcal{O}(p)$ with p the number of activities in the considered employee planning, or more loosely in $\mathcal{O}(m)$ with m the number of tasks in $\mathcal{T}$.

3.3.1.3 Additional notions related to elementary inserting transformations

Before ending this preliminary subsection, we propose to introduce two additional notions related to the elementary insertion: the feasibility gap as well as the backward and forward critical activities. These notions will be especially exploited in Chapter 5 in order to compute explanations and to justify in explanation texts why solutions are infeasible.

In a previous paragraph, we saw that the mathematical difference between the BET and the FLT informs us about the feasibility of an elementary insertion. We propose to give a name to the positive part of this difference: the *feasibility gap*.

Feasibility gap related to an elementary insertion transformation. Consider a feasible planning $(\mathcal{R}_i, \mathcal{C}_i)$ of an employee i, two consecutive activities (k_1, k_2) in $\mathcal{R}_i$ and a task j not in $\mathcal{R}_i$ such that employee i is skilled-enough to perform it. We call *feasibility gap* the quantity $max(st_j^b - st_j^f, 0)$. This quantity indicates whether the elementary insertion j between k_1 and k_2 in the planning of i is feasible, and if not it the measures the amount of time constraint violation:

- if $max(st_j^b - st_j^f, 0) = 0$, *i.e.* $st_j^b \leq st_j^f$, then the insertion is feasible;
- if $max(st_j^b - st_j^f, 0) > 0$, *i.e.* $st_j^b > st_j^f$, then the insertion is infeasible, and activities k_1 and k_2 would need to be started respectively earlier and later by a total amount of $st_j^b - st_j^f$ to make the insertion feasible.

One of the main interests of the feasibility gap is that it allows us to *i)* quantify the infeasibility of a neighboring solution obtained after an insertion and *ii)* to compare the infeasibilities of two neighboring solutions obtained after different insertions. Indeed, suppose that we build two different neighboring solutions, each one obtained by applying a single insertion to the same original solution. If both neighboring solutions are infeasible, *i.e.* their corresponding feasibility gap is positive, then we can view the neighboring solution with the smallest feasibility gap as the "least infeasible" solution out of these two solutions.

The feasibility gap will be particularly useful in Chapter 5 where we will often look for the least infeasible solution in a given neighborhood.

In the case where the elementary insertion of a given task in a planning is infeasible, it may be interested to identify which activities of the planning prevent this insertion to be feasible. Suppose that one wants to apply to a solution the elementary insertion of task j in an employee planning between activity k_1 and activity k_2. With the aim of making this insertion feasible, one may *i)* shift backward, as much as possible, all the activities of the planning performed before k_1 (included), *ii)* shift forward, as much as possible, all the activities performed after k_2 (included), and *iii)* try to fit task j in the temporal space created between k_1 and k_2. In case this insertion is infeasible, it may be useful to identify which activities in the planning, before k_1 (included) and after k_2 (included), prevent the temporal space created between k_1 and k_2 to be large enough to feasibly fit task j. We refer to these activities as *backward* and *forward critical activities*.

Backward and forward critical activities. Following on the above-definition of the feasibility gap, suppose that the insertion task j in the planning of employee i between k_1 and k_2 is infeasible, *i.e.* $st_j^b > st_j^f$. We define the *backward critical activity* (resp. *forward critical activity*) as the first activity, in a descending order (resp. ascending order), starting from j then k_1 and previous activities (resp. starting from j then k_2 and all following activities), whose lower bound (resp. upper bound) of availability time-window prevents it, and all the tasks between it and j, to be feasibly performed earlier (resp. later). The backward critical activity can be computed as follows:

- if $st_j^b = lb_j$, then it is j;
- else, starting from k_1 and following a descending order in $\mathcal{R}_i$, it is the first activity k such that $st_k - b_k = lba_k$;

and the forward critical activity as follows:

- if $st_j^f + dt_j = ub_j$, then it is j;
- else, starting from k_2 and following an ascending order in $\mathcal{R}_i$, it is the first activity k such that $st_k + f_k + dt_j = uba_k$.

The notion of backward critical activity is illustrated in Figure 3.4. With the intention of feasibly inserting task 27 in Ellen's planning between tasks 17 and 8, activities before task 17 (included) are shifted backward as much as possible. Still, task 26 cannot be shifted further backward due to the lower bound lbt_{26} of its availability time window, represented with the left red square bracket. Since the sequence of activities from task 26 to task 27 is interrupted, the end time of task 27, drawn as a dashed vertical line, cannot occur earlier. However, the upper bound ubt_{27} of its availability time window is much earlier, as shown with the second red bracket. In other words, in order to satisfy its corresponding availability constraints, task 27 would need to be performed earlier, as symbolized with the pair of dashed arrows. In this situation, task 26 is the backward critical activity.

To conclude on this preliminary subsection, we detailed three elementary transformations, namely shifting, removing and insertion transformations. We also delved into efficiency considerations about assessing their feasibility and computing neighboring solutions. In the upcoming subsections, we continue this effort with more complex transformations, starting with constant-size transformations.

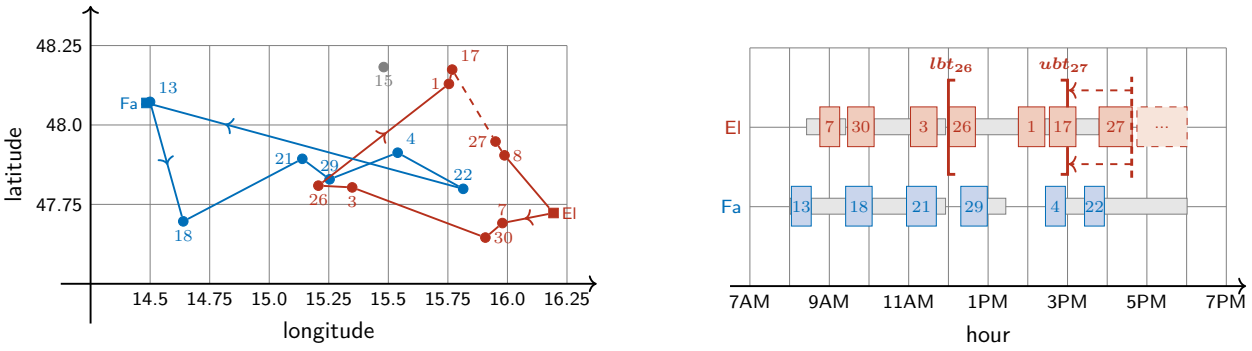


Figure 3.4: Backward critical activity, task 26, preventing task 27 to be performed earlier when inserted task 27 in Ellen's planning between activities 17 and 8.

3.3.2 Constant-size transformations

We recall that a solution transformation tf is said *constant-size* if all the neighboring solutions of $\mathcal{N}(tf, \mathcal{S})$ are route-equal to each other. As mentioned earlier, elementary transformations presented in the previous subsection are actually constant-size, but we isolated them and referred to them as elementary to exploit them in order to define "practical" transformations for future Chapters 4 and 5. In this subsection, we actually define three new constant-size transformations, by combining elementary transformations, namely, *reassignment*, *exchange* and *reordering* constant-size transformations. In the paragraphs below, we detail these transformations, before making some general comments about them. Figure 3.5 shows routes of neighboring solutions obtained by applying examples of each of these constant-size transformations to the same original solution.

Reassigning a task from an employee planning to another. Let $(i, i') \in \mathcal{E}^2$ be two different employees, $j \in \mathcal{T}$ be a task performed by employee i' in $\mathcal{S}$, and $(k_1, k_2) \in \mathcal{A}_i^2$ be two activities performed consecutively by employee i. Let tf be the constant-size *reassignment* transformation. tf consists in reassigning j from the planning $(\mathcal{R}_{i'}, \mathcal{C}_{i'})$ of i' to the planning $(\mathcal{R}_i, \mathcal{C}_i)$ of i, between activities k_1 and k_2. Then, $\mathcal{N}(tf, \mathcal{S})$ is made of every solution $\mathcal{S}'$ obtained from $\mathcal{S}$ by:

i) applying the elementary transformation of removing j from $(\mathcal{R}_{i'}, \mathcal{C}_{i'})$ (without shifting start times in schedules);
ii) applying the elementary transformation of inserting j in $(\mathcal{R}_i, \mathcal{C}_i)$ between k_1 and k_2.

Exchanging a task with another one in an employee planning. Let $i \in \mathcal{E}$ be an employee, $j \in \mathcal{T}$ be a task not performed by any employee in $\mathcal{S}$ and $j' \in \mathcal{T}$ be a task performed by employee i. Let $(k_1, k_2) \in \mathcal{A}_i^2$ be the two activities performed by employee i respectively before and after j'. Let tf be the constant-size *exchange* transformation. tf consists in exchanging j' with j in the planning $(\mathcal{R}_i, \mathcal{C}_i)$ of i. Then, $\mathcal{N}(tf, \mathcal{S})$ is made of every solution $\mathcal{S}'$ obtained from $\mathcal{S}$ by:

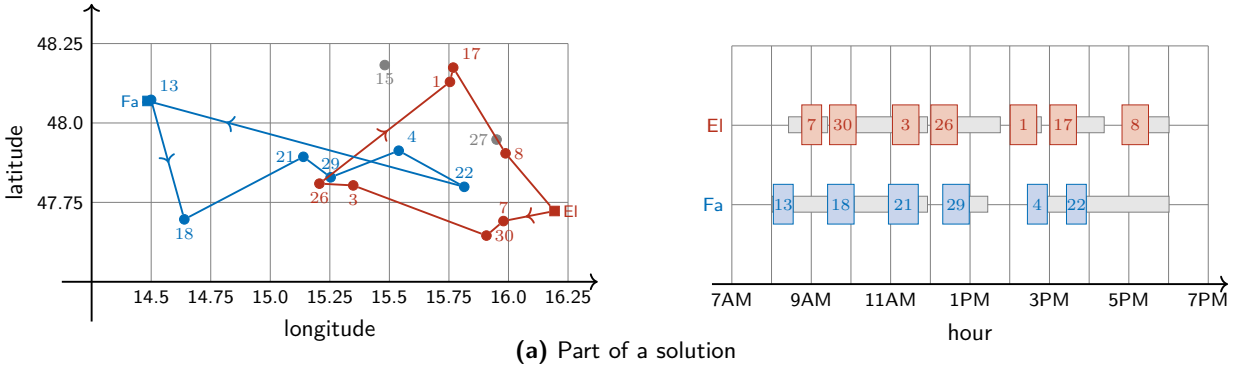
i) applying the elementary transformation of removing j' from $(\mathcal{R}_i, \mathcal{C}_i)$ (without shifting start times in schedules);
ii) applying the elementary transformation of inserting j in $(\mathcal{R}_i, \mathcal{C}_i)$ between k_1 and k_2.

Reordering a task in an employee planning. Let $i \in \mathcal{E}$ be an employee, $j \in \mathcal{T}$ be a task performed by employee i in $\mathcal{S}$ and $(k_1, k_2) \in \mathcal{A}_i^2$ be two activities, different from j, performed consecutively by employee i. Let tf be the constant-size *reordering* transformation. tf consists in reordering j in the planning $(\mathcal{R}_i, \mathcal{C}_i)$ of i, by moving it between (k_1, k_2). Then, $\mathcal{N}(tf, \mathcal{S})$ is made of every solution $\mathcal{S}'$ obtained from $\mathcal{S}$ by:

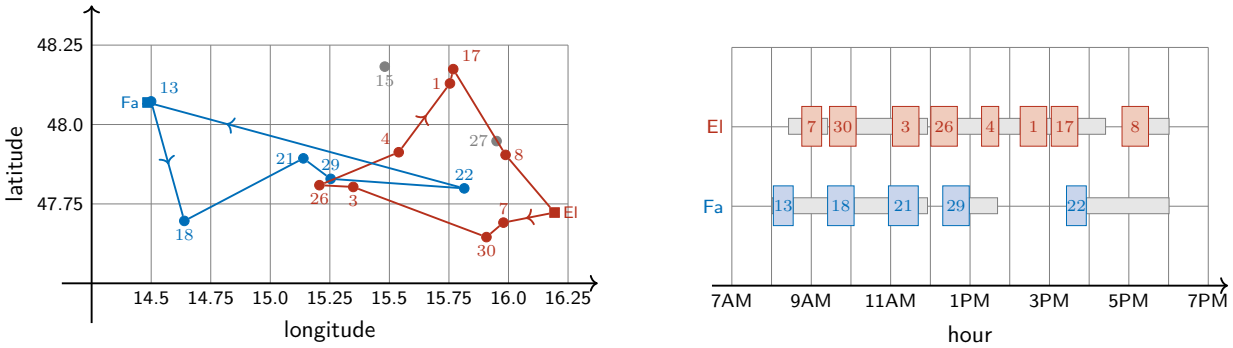
i) applying the elementary transformation of removing j from $(\mathcal{R}_i, \mathcal{C}_i)$ (without shifting start times in schedules);
ii) applying the elementary transformation of inserting it back between k_1 and k_2.

Let us make a few comments about these constant-size transformations. Let tf be a reassignment, exchange or reordering constant-size transformation.

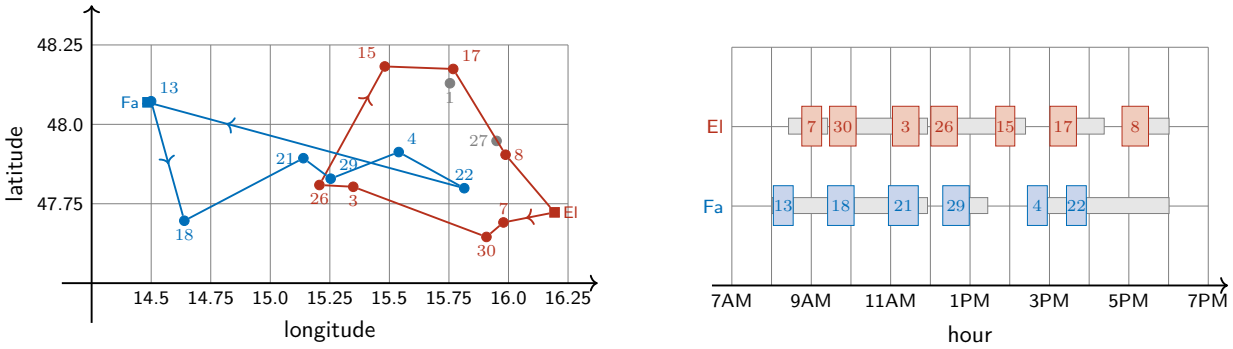
- **Checking transformation feasibility in constant or linear time.**
  - If tf is a *reassignment* transformation, since removing a task (without shifting start times), as it is done in step i), preserves the feasibility and since the planning from which task j is removed (planning of employee i') is different from the one in which the task is inserted (planning of employee i), we can simply check the feasibility of the insertion of j in the planning of i (without removing j from the planning of i'). Therefore, the feasibility of tf can be checked in *constant time* by using the constant-time feasibility check of elementary insertion.



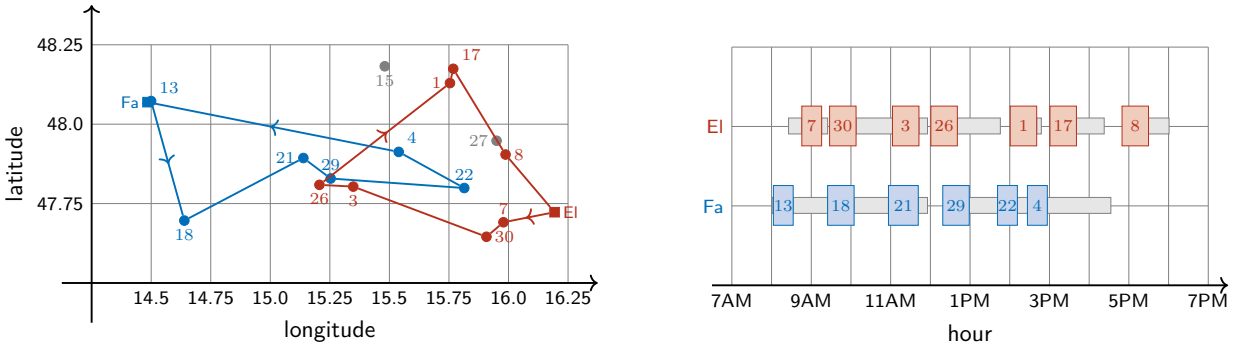
(a) Part of a solution



(b) After reassigning task 4 to Ellen between task 26 and task 1



(c) After exchanging task 1 with task 15 in Ellen's planning



(d) After reordering task 4 in Fabian's planning to be performed after task 22

Figure 3.5: Neighboring solutions obtained by applying constant-size transformations on a solution. In the top row is represented a part of the solution $\mathcal{S}$, presented in Figure 3.1, focusing on Ellen's and Fabian's plannings. Underneath, the second to fourth rows represent neighboring solutions (with the same focus on Ellen's and Fabian's plannings) obtained from $\mathcal{S}$ respectively by applying reassigning, exchanging and reordering constant-size transformations.

- If tf is an *exchange* transformation, we can actually check whether replacing task j' by task j is feasible without having to remove task j' from the planning of employee i, which computation would be linear in $\mathcal{O}(m)$. Indeed, let k_1 and k_2 be the activities respectively before and after j' in the planning of i. The BTS b_{k_1} and FTS f_{k_2} of k_1 and k_2 remain the same whether j' is in the planning of i or removed from it. Therefore, we can use b_{k_1} and f_{k_2} to compute the BET st_j^b and FLT st_j^f related to the insertion of j between k_1 and k_2 in the planning of i without removing j'. Then, the feasibility of tf can be checked in *constant time* by using the constant-time feasibility check of elementary insertion using BET st_j^b and FLT st_j^f
- If tf is a *reordering* transformation, we must first remove task j from the planning of employee i, which is performed in linear time $\mathcal{O}(m)$ (where m is the number of tasks in $\mathcal{T}$), and then check the feasibility of inserting j back in the same planning, which can be performed in constant time. Therefore, the feasibility of tf can be checked in *linear time* in $\mathcal{O}(m)$.

- **Building a neighboring solution - in linear time.** Whether tf is a reassignment, exchange or reordering constant-size transformation, tf consists in i) in removing a task and ii) inserting a task. Both transformations can be computed in linear time in $\mathcal{O}(m)$. Therefore building a given neighboring solution can be performed in *linear time* in $\mathcal{O}(m)$.
- **Building a neighboring solution - consistent insertion.** Let us emphasize that building a neighboring solution implies applying a single elementary insertion. This *consistent presence of a single insertion* as part of process of building a neighboring solution will play an important role in the computation of explanation in Chapter 5.

In this subsection, we detailed three new constant-size transformations, namely reassignment, exchange and reordering transformations. Along with the elementary insertion transformation, which is also constant-size, they form the four "practical" constant-size transformations which will be exploited in our methods for modeling and generating explanations. In the next subsection, we propose to extend these four transformations into polynomial-size transformations.

3.3.3 Polynomial-size transformations

We recall that a solution transformation tf is said *polynomial-size* if the number of subsets of $\mathcal{N}(tf, \mathcal{S})$ containing solutions that are all route-equal to each other is polynomial (non-constant) in n, number of employees in $\mathcal{E}$, and m, number of tasks in $\mathcal{T}$. Below, we outline how each of the four constant-size transformations - insertion, reassignment, exchange and reordering - can be extended in polynomial-size transformations.

Practical polynomial-size transformations. There are various ways to define a polynomial-size transformation from each of the four constant-size transformations: insertion, reassignment, exchange and reordering transformations. For each of these transformations, we propose below several practical polynomial-size extensions. Differences between the original constant-size transformation and its polynomial-size counterparts are emphasized in italics.

- **Insertion:**
  a. inserting a given non-performed task between *any* pair of consecutive activities of a given employee planning;
  b. inserting *any* non-performed task between *any* pair of consecutive activities of a given employee planning;
  c. inserting a given non-performed task between *any* pair of consecutive activities of *any* employee planning.

  Figure 3.6 shows the routes of some neighboring solutions obtained by applying a polynomial-size insertion transformation (extension a.) to the same original solution.
- **Reassignment:**
  a. reassigning a given task, performed by an employee, between *any* pair of consecutive activities in the planning of a given other employee;
  b. reassigning a given task, performed by an employee, between *any* pair of consecutive activities in the planning of *any* other employee.
- **Exchange:**
  a. exchanging *any* task performed by a given employee with a given non-performed task;
  b. exchanging *any* task performed by a given employee with *any* non-performed task;
  c. exchanging *any* task performed by *any* employee with a given non-performed task.
- **Reordering:**
  a. reordering a given task of a given employee planing by moving it between *any* pair of consecutive activities performed *before* this task in the planning;
  b. reordering a given task of a given employee planing by moving it between *any* pair of consecutive activities performed *after* this task in the planning;
  c. reordering a given task of a given employee planing by moving it between *any* pair of consecutive activities of the planning.

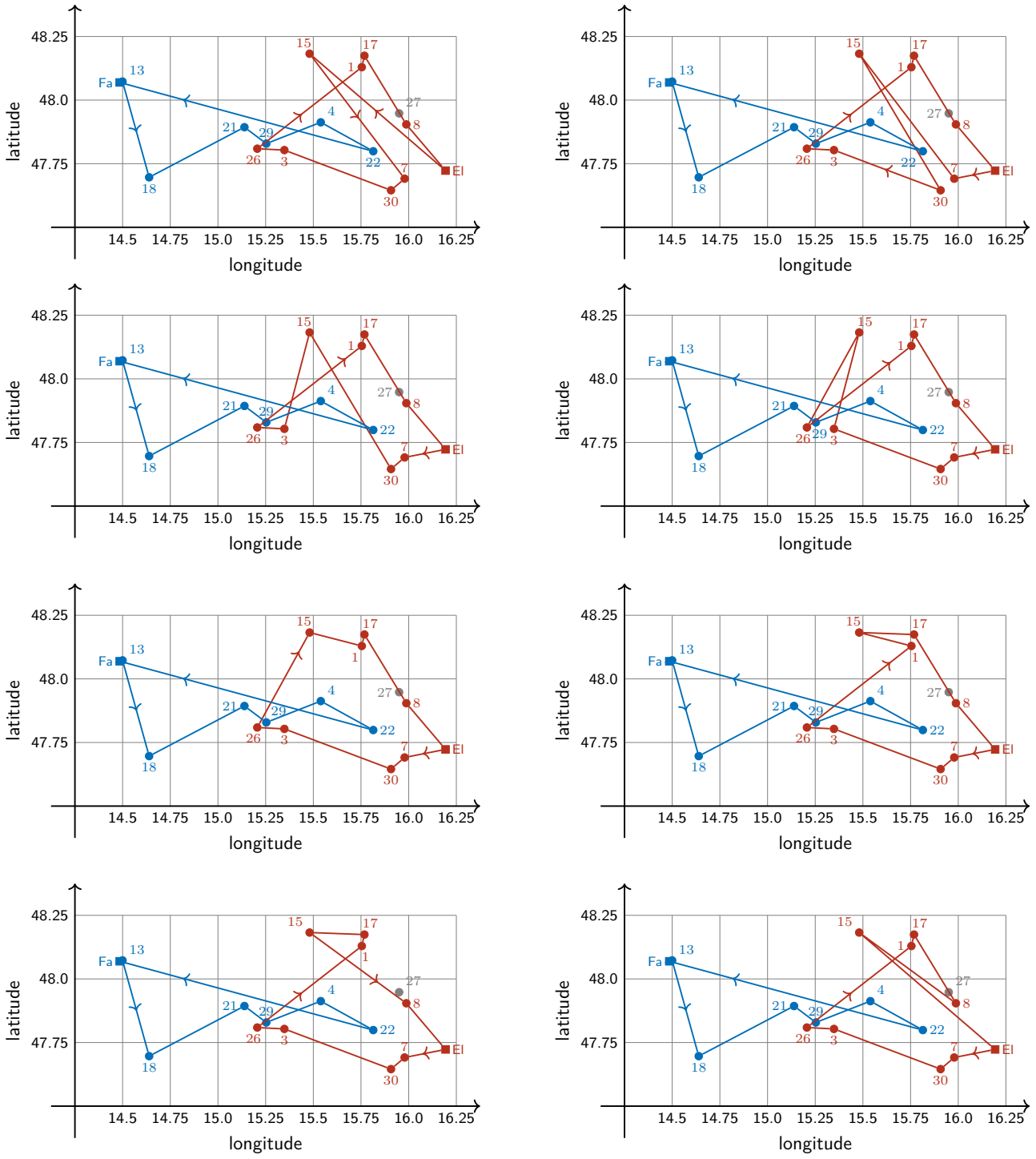


Figure 3.6: Neighboring solutions obtained by applying a polynomial-size insertion on a solution. Consider $\mathcal{S}$ the solution represented in Figure 3.1 and focus on Ellen's and Fabian's plannings. The figures represent the routes of neighboring solutions obtained from $\mathcal{S}$ by applying a polynomial-size insertion: inserting task (15) between every pair of consecutive activities in Ellen's planning.

A few comments similar to those made for constant-size transformations can be made about these polynomial-size transformations.

- **Checking transformation feasibility in polynomial time.** Mathematically, the neighborhood associated with a polynomial-size transformation can be seen as the union of the neighborhoods associated with the corresponding constant-size transformation applied over a linear set of activities or employees (*e.g.* the neighborhood associated with the polynomial-size insertion transformation "a", which implies inserting a task in an employee planning, can be obtained as the union of the neighborhoods associated with the constant-size insertion transformation applied over the set of pairs of consecutive activities of this employee planning, which is linear in m number of tasks). Building upon that mathematical union, for every polynomial-size transformation, we can devise a polynomial-time feasibility check, in m number of tasks, and n, number of employees. For a given polynomial-size transformation, we repeat the feasibility check of the corresponding constant-size transformation over the linear set of activities or employees. If any of these check is feasible, then the polynomial-size transformation is feasible.
- **Building a neighboring solution - in linear time.** While the feasibility checks of polynomial-size transformations require more computations than the ones of constant-size transformations, the process for building a given neighboring solution remains unchanged. Building a neighboring solution can then be performed in *linear time* in $\mathcal{O}(m)$.
- **Building a neighboring solution - consistent insertion.** There is again a *consistent presence of a single insertion* as part of process of building a neighboring solution related to a polynomial-size transformation.

In this subsection, we presented various polynomial-size transformations that can be seen as applying constant-size transformations over sets employees or tasks instead of a unique employee or a unique task. Checking whether such a polynomial-size transformation is feasible implies higher algorithmic complexity computations than in the case of constant-size transformations. However, these computations remained polynomial in n, number of employees, and m, number of tasks. In the next subsection, we study exponential-size transformations for which the equivalent computations are no longer polynomial.

3.3.4 Exponential-size transformations

We recall that a solution transformation tf is said *exponential-size* if the number of subsets of $\mathcal{N}(tf, \mathcal{S})$ containing solutions that are all route-equal to each other is exponential in n, number of employees in $\mathcal{E}$, and m, number of tasks in $\mathcal{T}$. Below, we outline how each of the four constant-size transformations - insertion, reassignment, exchange and reordering transformations - can be extended into an exponential-size transformation.

Practical exponential-size transformations. Given a constant-size transformation, among the insertion, reassignment, exchange and reordering ones, we propose to extend it into an exponential-size transformation tf as follows. $\mathcal{N}(tf, \mathcal{S})$ is made of every solution $\mathcal{S}'$ obtained from $\mathcal{S}$ by:

i) applying the constant-size transformation on $\mathcal{S}$ - as part of this transformation, an elementary task insertion is applied on the planning of an employee $i \in \mathcal{E}$;
ii) permuting the tasks (except the departure and return activities) in the route of i and setting values of start times for the schedule of i.

Unlike the definitions of constant-size and polynomial-size transformations, this definition is more theoretical than practical: the two steps above do not correspond to the way we will build neighboring solutions in practice, as described below.

We make a few comments about these exponential-size transformations.

- **Checking transformation feasibility using ILP solving.** By definition, the neighborhood associated with an exponential-size transformation can be obtained by considering all the permutations of the tasks involved in an employee planning - and setting corresponding schedules. Mathematically, it corresponds to a union of route-equal solution groups where the size of the union is the number of permutations of a set of tasks. Contrary to the polynomial-size transformation case, we cannot build upon this mathematical union to devise a feasibility check for exponential-size transformations as iterating over a set of permutations is usually computationally intractable. We propose instead to resort to ILP modeling and solving in order to implicitly explore the set of neighboring solutions with the aim of determining whether the transformation is feasible. This will be detailed in Chapter 5.
- **Building a neighboring solution - using ILP solving.** The same ILP-based algorithm used to check the feasibility of an exponential-size transformation will be used to build a neighboring solution.

- **Building a neighboring solution - consistent insertion.** Let us emphasize again that building a neighboring solution implies applying a single elementary insertion. Admittedly, this insertion is then combined with a permutation which is also critical and may cause the infeasibility of the solution. However, in practice the insertion and the permutation involved in the exponential-size transformation will be handled simultaneously through ILP solving as it will be detailed in Chapter 5. Thus, we can again conclude the *consistent presence of a single insertion* as part of process of building a neighboring solution.

To conclude, in this subsection, we outlined various exponential-size transformations that can be seen as the application of a constant-size transformation followed by a permutation of the activities of the modified employee planning. Since LS solving algorithms rely on repeating various solution transformations in order to optimize gradually the solution, they do not usually involve such exponential-size transformations whose feasibility check and neighboring solution construction require too much computation time. On the contrary, such transformations can be envisaged within our explanation methods, as they will never be involved more than once in the calculation of an explanation. Chapters 4 and 5 will make the use of all these transformations and neighborhoods in our explanatory methods more tangible for the reader.

3.4 Conclusion

In this chapter, we first delineated the characteristics of the WSRP use case, for which we aimed at developing techniques to explain its solutions. Especially, we modeled this use case as a bi-objective ILP problem that we called *main model*.

Then, we gave a comprehensive presentation of solution *transformations* and *neighborhoods*, as these will play a crucial role in our endeavor to model, compute and generate explanations, described in the following chapters. We classified transformations into three categories depending on the structure of their corresponding neighborhoods, namely *constant-size*, *polynomial-size* and *exponential-size* transformations. We listed various families of solution transformations including transformations about *inserting* a task in an employee planning, *reassigning* a task from an employee planning to another, *exchanging* a task in an employee planning with another task or *reordering* tasks within an employee planning. We also introduced various notions that will be used in the following chapters, including the *route-equality* of solutions, the *Backward Earliest start Time*, the *Forward Latest start Time* and the *feasibility gap* associated with an insertion as well as the *critical activities* when such an insertion is infeasible.

To facilitate future reference, Table 3.1 sums up all the practical transformations introduced in this chapter. Each row corresponds to a specific transformation. It provides a description of this transformation as well as its category. It recalls the computational complexities associated with checking the feasibility of the transformation and building a neighboring solution. It also recalls the main operations to apply for building a neighboring solution, emphasizing the *consistent presence of a task insertion in an employee planning* among these operations. To identify each transformation listed in this table, we introduce labels. The *label* of a transformation is a tuple "(X,Y,Z)" such that:

- "X" describes the family of the considered solution transformation; it may be equal to "Ins", "Ass", "Ex" or "Ord" which are short-names standing respectively for "insertion", "reassignment", "exchange" and "reordering".
- "Y" gives the category of solution transformation defined by the structure of the corresponding neighborhood; it is either "C", "P" or "E" which stand for "constant-size", "polynomial-size" and "exponential-size".
- "Z" is a letter (*e.g.* "a", "b" or "c") referring to a variant of the transformation of family "X" and category "Y"; if "X" & "Y" defines a unique transformation, then there is no letter for "Z" and the label is simply "(X, Y)".

Given the central role of transformations in our explanation approach, other tables in subsequent chapters, such as the list of observations Table 4.1 or the list of questions Table 4.2, will relate to Table 3.1.

Label	Description of the transformation	Category	Feasibility	Building a neigh. solution	
(Ins,C)	Inserting a given non-performed task between a given pair of consecutive activities in a given employee planning	Constant -size	$\mathcal{O}(1)$	Elementary *insertion*	$\mathcal{O}(m)$
(Ins,P,a)	Inserting a given non-performed task between any pair of consecutive activities in a given employee planning	Polynomial -size	$\mathcal{O}(m)$	Elementary *insertion*	$\mathcal{O}(m)$
(Ins,P,b)	Inserting any non-performed task between any pair of consecutive activities in a given employee planning	Polynomial -size	$\mathcal{O}(m^2)$	Elementary *insertion*	$\mathcal{O}(m)$
(Ins,P,c)	Inserting a given non-performed task between any pair of consecutive activities in any employee planning	Polynomial -size	$\mathcal{O}(nm)$	Elementary *insertion*	$\mathcal{O}(m)$
(Ins,E)	Inserting any non-performed task in a given employee planning while permuting the order of the tasks in this planning	Exponential -size	ILP-based	*Insertion* w. permutation	ILP-based
(Ass,C)	Reassigning a given task performed by an employee between a given pair of consecutive activities in the planning of a given other employee	Constant -size	$\mathcal{O}(1)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ass,P,a)	Reassigning a given task performed by an employee between any pair of consecutive activities in the planning of a given other employee	Polynomial -size	$\mathcal{O}(m)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ass,P,b)	Reassigning a given task performed by an employee between any pair of consecutive activities in the planning of any other employee	Polynomial -size	$\mathcal{O}(nm)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ass,E)	Reassigning a given task performed by an employee in the planning of a given other employee while permuting the order of the tasks in this planning	Exponential -size	ILP-based	Elementary removal, *insertion* w. permutation	ILP-based
(Ex,C)	Exchanging a given task performed by a given employee with a given non-performed task	Constant -size	$\mathcal{O}(1)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ex,P,a)	Exchanging any task performed by a given employee with a given non-performed task	Polynomial -size	$\mathcal{O}(m)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ex,P,b)	Exchanging any task performed by a given employee with any non-performed task	Polynomial -size	$\mathcal{O}(m^2)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ex,P,c)	Exchanging any task performed by any employee with a given non-performed task	Polynomial -size	$\mathcal{O}(nm)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ex,E)	Exchanging any task performed by a given employee with a given non-performed task while permuting the order of the tasks	Exponential -size	ILP-based	Elementary removal, *insertion* w. permutation	ILP-based
(Ord,C,a)	Reordering a given task in a given employee planning by moving the task between a given pair of consecutive activities performed after it	Constant -size	$\mathcal{O}(m)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ord,C,b)	Reordering a given task in a given employee planning by moving the task between a given pair of consecutive activities performed before it	Constant -size	$\mathcal{O}(m)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ord,P,a)	Reordering a given task in a given employee planning by moving the task between any pair of consecutive activities performed after it	Polynomial -size	$\mathcal{O}(m)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ord,P,b)	Reordering a given task in a given employee planning by moving the task between any pair of consecutive activities performed before it	Polynomial -size	$\mathcal{O}(m)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ord,P,c)	Reordering a given task in a given employee planning by moving the task between any pair of consecutive activities	Polynomial -size	$\mathcal{O}(m)$	Elementary removal, elementary *insertion*	$\mathcal{O}(m)$
(Ord,E)	Reordering the tasks of a given employee planning	Exponential -size	ILP-based	Elementary removal, *insertion* w. permutation	ILP-based

Table 3.1: List of transformations. Each transformation is characterized by a label and a description. We write in italics some words in the description to emphasize the differences between the transformations about the same topics. The category (constant-size, polynomial-size or exponential-size) of the transformation is given. For constant-size and polynomial-size transformations, we also provide the algorithmic complexity of assessing the feasibility of the transformation as well as the one of building any neighboring solution. We recall that m is the number of tasks and n the number of employees in the instance.

Chapter 4 New framework for modeling explanations

4.1 Introduction

In Chapter 3, we specified the Combinatorial Optimization (CO) problem we focus on in this thesis: we defined a use case of the *Workforce Scheduling and Routing Problem* (*WSRP*) (see Subsection 3.2.1) and formulated it as a bi-objective Integer Linear Program (ILP) model, which we called *main model* (see Model 1 in Subsection 3.2.2). We also described various *transformations* that can be applied on a solution in order to build new ones, including insertion, reassignment, exchange and reordering transformations. We called *neighborhood* the set of solutions produced by such a transformation, and we organized transformations in categories, namely *constant-size*, *polynomial-size* and *exponential-size* transformations, depending on the structure of their neighborhoods (see Section 3.3).

In this chapter, we present our framework modeling the explanation of WSRP solutions, which exploits the ILP formulation of our WSRP use case as well as the solution transformations. Throughout this chapter, we consider that end-users have $\mathcal{S}$, a feasible solution of an instance $\mathcal{I}$, that they have typically obtained thanks to a WSRP-solving system. In order to illustrate the content developed in this chapter, let us bring back the example introduced in Chapter 1 presenting a typical situation where end-users wonder about an intriguing fact in their solution. Figure 4.1 below recalls Figure 1.1 representing this situation.

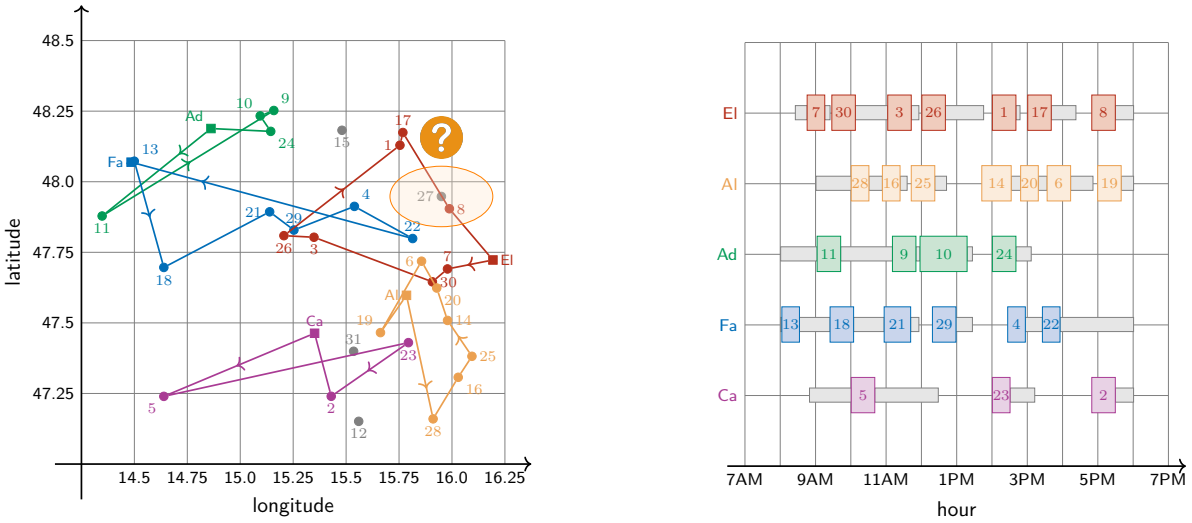


Figure 4.1: Intriguing observation about the WSRP solution represented in Figure 3.1: "Ellen is not performing task 27 in addition to the activities of her planning" (highlighted by the orange ellipse). Recall of Figure 1.1.

Figure 4.1, in the graph of routes on the left, an orange ellipse highlights an observation that end-users could make about the solution and that may be intriguing for them: despite the positioning of task 27, which is on the route of Ellen between task 17 and task 8, "Ellen is not performing task 27 in addition to the activities of her planning".

Then, a natural question that end-users could ask regarding this observation is "Why is Ellen not performing task 27 in addition to the activities of her planning?". To answer it, we can rely on a specific kind of explanation, named a *contrastive explanation*, where the aim is to clarify why one fact occurred in contrast to another (see Sections 2.2 and 2.3).

Actually, noticing that Ellen's schedule is quite full and almost uninterrupted, end-users may anticipate that it is impossible to add task 27, a 50-minute long task, to Ellen's planning, so they could ask instead "Ellen is not performing task 27 in addition to the activities of her planning, but what if task 27 lasts 40 minutes instead of 50 minutes?". Such a question is answered by a scenario explanation, that is to say by describing how changes in the input, *i.e.* in the instance parameters such as the duration of task 27, affect a fact occurring in the output, *i.e.* the solution (see Section 2.3).

Alternatively, instead of understanding why Ellen does not perform task 27 or checking if suggested alterations of the instance parameters help making Ellen perform the task, end-users could wish to force task 27 to be included in her activities and raise the question: "How to make Ellen perform task 27 in addition to the activities of her planning, considering that the availability time windows of each task in Ellen's planning may be widen by at most 10 minutes?". In this case, the answer is a *counterfactual explanation*, which aims to identify which parameters could be changed in the instance to obtain a feasible solution featuring a user-defined alternative fact, namely here a solution such that Ellen performs task 27 in addition to the tasks of her planning (see Section 2.3).

This example illustrates the types of questions end-users could ask about a given observation regarding a solution, to understand this observation, or to move towards a desired, non-obtained and possibly unobtainable solution. The type of question asked ("Why not?", "What if?" or "How to") conditions the type of explanation we give (contrastive, scenario and counterfactual). Regardless of the type, it should be stressed that the production of explanations is triggered by questions raised by end-users wondering about the relevance of facts observed in their solution.

Thus, we propose in this chapter an original framework for modeling explanations about solutions of WSRP instances. This framework consists in a process starting from an end-user observation about a solution and ending with an explanation text about this observation. Figure 4.2 depicts this process and details it into various steps.

The top part of this figure represents the steps related to end-users: from an end-user perspective, our framework consists in (A) making an observation o about $\mathcal{S}$, feasible solution of instance $\mathcal{I}$, (B) asking a question q based on this observation o, and (F) obtaining an explanation as a text $ux(q)$. Step (B), associated with step (A), allows end-users to raise questions about various topics (related to solution transformation introduced in Section 3.3) and to obtain explanations of different types (contrastive, scenario and counterfactual).

The bottom part of the figure, which corresponds to the causality diagnosis process (see Section 2.2), is divided into several consecutive mathematical steps (C), (D) and (E) leading from q to $ux(q)$. Steps (C) and (D) consist in translating q, which is expressed in common language, into mathematical terms. To do so, step (C) makes use of the notion of *neighborhood* $\mathcal{N}(o, \mathcal{S})$ that we introduced in Section 3.3, in order to redefine q into another question $dpq(q)$ using mathematical terms related to decision problems. Remark that the neighborhood, noted $\mathcal{N}(o, \mathcal{S})$, here depends on an observation o instead of a transformation tf. As we will see it in Subsection 4.2.1, observations are assumed to be related to transformations, that is why we allow ourselves this slight abuse of the neighborhood notation. Then, step (D) introduces an ILP model, referred to as *foil model*, denoted $fm(.)$ and relying on the structure of the main model $mm(.)$, with the purpose of rephrasing $fmq(q)$ with terms related to ILP. Finally, step (E) uses feasibility information about $fm(.)$ to produce a mathematical explanation $x(q)$.

To the best of our knowledge, this is the first time that, in a CO context, an explanation framework addresses a large range of topics to explain, handles various types of explanations, provides a mathematical-programming interpretation of questions and explanations, and harnesses the concept of neighborhood to play a central role in the modeling of the causality diagnosis process. This framework was initially introduced in [LGMO22] but it focused exclusively on contrastive explanations.

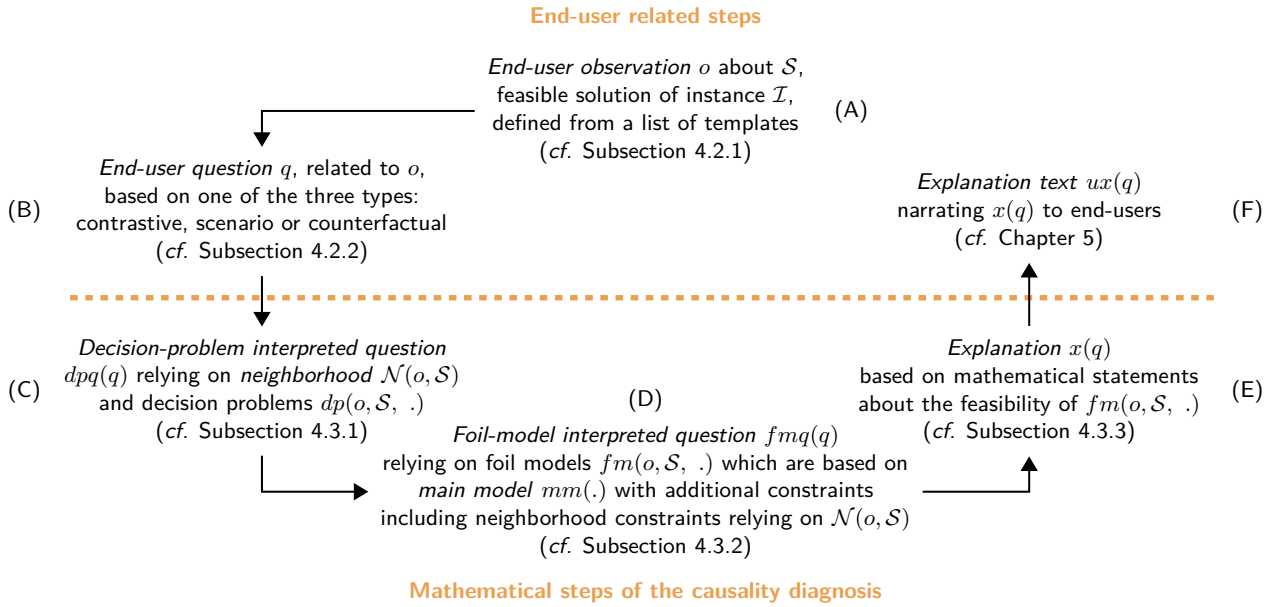


Figure 4.2: Overview of our framework modeling the explanation process, from observations to explanations.

The remainder of this chapter is organized as follows. Section 4.2 deals with end-user related steps (A) and (B). We specify the nature of the end-user observations handled by our framework and describe how questions can be formulated based on these observations. Section 4.3 delves into the causality diagnosis steps from (C) to (E). We present how questions expressed in common language can be interpreted into mathematical terms. This section introduces key concepts such as the one of foil model and ends with a formal definition of explanations. The generation of explanation texts, which correspond to step (F), will be developed in Chapter 5.

4.2 End-user related steps - From observations to questions

4.2.1 End-user observations about a solution

For various reasons, *e.g.* getting a better understanding of their solution $\mathcal{S}$, end-users may want explanations about $\mathcal{S}$. Specifically, they may want explanations about facts observed in $\mathcal{S}$. In this subsection focusing on step (A) of Figure 4.2, we specify what contents we seek to explain with our approach. Especially, we introduce the notion of *transformation-related observations*, in short *observations*, which are the starting point of the explanation process in our framework.

Contents to explain: transformation-related observations about solutions. We detail hereafter some assumptions regarding the contents to explain to end-users in our approach.

- **Scope of explanations.** As mentioned in Section 2.3, explanations may have a local or global scope. In our approach, we choose to deal with local explanations, more precisely, explanations *about solutions* of WSRP instances.
- **Focus on observations.** Rather than to explain a solution $\mathcal{S}$ as a whole (focusing *e.g.* on its quality), we seek to explain facts observed in $\mathcal{S}$, *i.e. observations* (*e.g.* the observation that a given task is not performed by any employee in $\mathcal{S}$). Doing so allows us to draw the attention of end-users to key information and reduce information overload.
- **Transformation-related observations.** In order to explain these observations about $\mathcal{S}$, we anticipate that we will have to compare $\mathcal{S}$ to other solutions. Therefore, we require that these observations can be associated with solution *transformations*, such as the ones we described in Section 3.3 (*e.g.* inserting a given non-performed task in a given employee planning between two given activities).

To sum up, we design an approach for *explaining transformation-related observations about a solution*. Dealing with such explanations can be advantageous for both end-users and designers of explanation approaches. Indeed, such assumptions incite end-users to make their issue explicit by specifying: 1) which observation, in which specific part of $\mathcal{S}$, they want to question; 2) what alternative solutions they would have expected to have instead of $\mathcal{S}$. This helps then designers to restrict their examination and propose relevant explanations to end-users in reasonable time.

Although making these assumptions may seem restrictive, we will see below that our approach is able to address a significant number of observations.

As mentioned in Section 2.5, the primary target audience of our explanations are planners *i.e.* persons in charge of designing employee plannings thanks to a WSRP-solving system. In association with our industrial partner DecisionBrain (see Section 1.1), we enumerated various observations that planners may make about solutions. Many observations relate to the transformations detailed in Section 3.3. We present below the list of these observations.

List of end-user observation templates. Table 4.1 is a non-exhaustive list of observation templates that end-users can use to build observations that our approach is able to deal with. Each observation template is characterized by a *label*, *e.g.* (Ins,C), and a *template text*, *e.g.* "$\langle$employee $i^*\rangle$ is not performing $\langle$task $j^*\rangle$ just after $\langle$activity $k^*\rangle$".

- The label of an observation template coincides with the label of its related transformation. We recall that the transformation label is a tuple "(X,Y,Z)" where "X" indicates the family of the transformation (task insertion, reassignment[1], exchanging or reordering), "Y" its category (constant-size, polynomial-size or exponential-size) and "Z" a variant (see Table 3.1 in Section 3.4).
- Each template text contains one or several symbols $\langle . \rangle$ which indicate *fields* that end-users have to specify, using data from the instance $\mathcal{I}$ (*e.g.* the name of an employee, the id of a task or an activity). We annotated with a superscript * the indices of employees and tasks involved in the fields of the template in order to differentiate them from generic indices of tasks and employees.

To illustrate this paragraph, let us recall the introductory example presented in Section 4.1. The observation that was given as an example, namely "Ellen is not performing task 27 in addition to the activities of her planning", can be built from (Ins,E) observation template "$\langle$Employee $i^*\rangle$ is not performing $\langle$task $j^*\rangle$ in addition to the activities of their planning" by choosing field values "Ellen" and "task 27".

In this subsection, we described step (A) of Figure 4.2: we specified assumptions about the starting point of our approach, namely transformation-related observations about a solution, and we presented a list of templates allowing to formulate such observations in Table 4.1. From now on, we only consider end-user observations that are transformation-related, like the ones that can be built thanks to Table 4.1 templates. In the next subsection, we move to step (B).

[1]Note that none of the observation labels coincide with a label corresponding to an assignment transformation (*i.e.* when "X" is equal to "Ass"). Actually, assignment transformations can be seen as particular cases of insertion transformations where the inserted task is already performed by an employee. Thus, in (Ins,C), (Ins,P,a) and (Ins,E) observation templates, if task j^* is performed by an employee, the observation templates actually relate to (Ass,C), (Ass,P,a) and (Ass,E) transformations. This allows to reduce the number of observation templates.

Label	Observation template text
(Ins,C)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ just after ⟨activity k^*⟩."
(Ins,P,a)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ between any pair of consecutive activities of their planning."
(Ins,P,b)	"⟨Employee i^*⟩ is not performing any non-performed task between two consecutive activities of their planning."
(Ins,P,c)	"No employee is performing ⟨task j^*⟩ between two consecutive activities of their planning."
(Ins,E)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ in addition to the activities of their planning, regardless of the planning order."
(Ex,C)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ rather than ⟨task k^*⟩."
(Ex,P,a)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ rather than any of their tasks."
(Ex,P,b)	"⟨Employee i^*⟩ is not performing any non-performed task rather than one of the activities of their planning."
(Ex,P,c)	"No employee is performing ⟨task j^*⟩ rather than one of the activities of their planning."
(Ex,E)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ rather than any of their tasks, regardless of the planning order."
(Ord,C,a)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ later in their planning, just after ⟨task k^*⟩."
(Ord,C,b)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ earlier in their planning, just before ⟨task k^*⟩."
(Ord,P,a)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ at a later stage of their planning."
(Ord,P,b)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ at an earlier stage of their planning."
(Ord,P,c)	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ at any other stage of their planning."
(Ord,E)	"⟨Employee i^*⟩ is not performing the activities of their planning in a different order."

Table 4.1: List of observation templates related to the list of transformations presented in Table 3.1. Each observation template is characterized by a label and a template text. Without making it explicit, observation templates whose label's second term is "C" or "P" assume that end-users do not consider changing the order of activities in the employee planning. On the opposite, observation templates whose label's second term is "E" assume that end-users consider that the order of activities in the employee planning can be changed.

4.2.2 End-user questions about a solution

Step (B) of Figure 4.2 essentially corresponds to end-users defining questions based on observations that they made about their solution $\mathcal{S}$. As illustrated by the example presented at the beginning of this chapter, from a single observation (*e.g.* "Ellen is not performing task 27 in addition to the activities of her planning"), various types of questions can be defined, including a question asking for a contrastive explanation ("Why is Ellen not performing task 27 in addition to the activities of her planning?") or one asking for a counterfactual explanation ("How to make Ellen perform task 27 in addition to the activities of her planning?"). In our approach, we consider these two types of questions as well as questions asking for scenario explanations. These questions will then be used as ways to request explanations.

Before presenting how end-user observations can actually be used to build questions of various types, we need to introduce the notion of instance relaxation. We recall that the data characterizing an instance, as well as the notations relating to these data, are described in Subsection 3.2.1.

Instance relaxation. We call *relaxation of an instance* $\mathcal{I}$, any instance $\mathcal{I}'$ such that the set of feasible solutions of $\mathcal{I}$ is included in the one of $\mathcal{I}'$. Usually, $\mathcal{I}'$ is obtained from $\mathcal{I}$ by altering the values of some parameters, *e.g.* by reducing the duration dt_j of one or several tasks, by extending the working time windows $[lbe_i,\ ube_i]$ of one or several employees, *etc.* We also say that $\mathcal{I}'$ is obtained by *relaxing* $\mathcal{I}$.

Now, let us introduce the different types of questions.

End-user questions of three different types. From an observation o about $\mathcal{S}$, feasible solution of an instance $\mathcal{I}$, questions of three different types can be built. Let q be one of these questions, then q can be:

- a *contrastive question*, *i.e.* a question asking for a contrastive explanation, in which case,
  $q = q_c(o, \mathcal{S}, \mathcal{I}) =$ "Why o is observed in $\mathcal{S}$, solution of $\mathcal{I}$?";
- a *scenario question*, *i.e.* a question asking for a scenario explanation, in which case,
  $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}') =$ "o is observed in $\mathcal{S}$, solution of $\mathcal{I}$, but what if $\mathcal{I}$ is changed into $\mathcal{I}'$?"
  where the instance $\mathcal{I}'$ is obtained by relaxing $\mathcal{I}$;
- a *counterfactual question*, *i.e.* a question asking for a counterfactual explanation, in which case
  $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}') =$ "How to make $\neg o$ possible considering that $\mathcal{I}$ can be changed into $\mathcal{I}' \in \mathcal{J}'$?"
  where $\neg o$ is the negation of the observation (*e.g.* if o is the observation that "Ellen is not performing task 27 in

addition to her tasks" then $\neg o$ is the opposite "Ellen is performing task 27 in addition to her tasks") and $\mathcal{J}'$ is a set of instances obtained by relaxing $\mathcal{I}$.

As scenario and counterfactual questions involve other instances than $\mathcal{I}$, respectively $\mathcal{I}'$ and $\mathcal{I}' \in \mathcal{J}'$, we will sometimes refer to $\mathcal{I}$ as the *original instance*, by opposition to $\mathcal{I}'$ or $\mathcal{I}' \in \mathcal{J}'$, which we will call *relaxations*.

In the previous section, we presented a list of end-user observation templates in Table 4.1. From this list, we can build another list of end-user question templates as follows.

List of end-user question templates. Table 4.2 presents how each observation template in Table 4.1 can be mapped into three question templates: contrastive, scenario and counterfactual question templates.

Note that, in the above definition of a scenario question, $q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}') =$ "o is observed in $\mathcal{S}$, solution of $\mathcal{I}$, but what if $\mathcal{I}$ is changed into $\mathcal{I}'$?", a relaxation $\mathcal{I}'$ of the original instance $\mathcal{I}$ is supposed to be provided. In scenario question templates, we propose that end-users instead specify the alterations to apply to the parameters of $\mathcal{I}$ to obtain $\mathcal{I}'$, as alterations are applied only to a few parameters in practice. Similarly, in the above definition of a counterfactual question, $q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}') =$ "How to make $\neg o$ possible considering that $\mathcal{I}$ can be changed into $\mathcal{I}' \in \mathcal{J}'$?", a set of instances $\mathcal{J}'$, obtained by relaxing $\mathcal{I}$, is supposed to be provided. In counterfactual question templates, we propose that end-users define this set of instances $\mathcal{J}'$ by specifying the groups of alterations that they allow to apply to the parameters of $\mathcal{I}$.

To illustrate this paragraph, let us return again to the introductory example presented in Section 4.1. The contrastive, scenario and counterfactual questions that were given as examples can be built from the three question templates related to observation template (Ins,E), by choosing field values "Ellen" and "task 27". For the sake of simplicity, the example of counterfactual question did not mention any group of alterations to apply to the instance.

Label	Type	Question template text
(Ins,C)	Contrastive	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ just after ⟨activity k^*⟩?"
	Scenario	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ just after ⟨activity k^*⟩, but what if ⟨changes applied to instance $\mathcal{I}$ parameter values⟩?"
	Counterfactual	"How to make "⟨employee i^*⟩ perform ⟨task j^*⟩ just after ⟨activity k^*⟩, considering ⟨alterations that can be applied to instance $\mathcal{I}$ parameter values⟩?"
⋮	⋮	⋮
(Ins,E)	Contrastive	"Why is ⟨employee i^*⟩ not performing ⟨task j^*⟩ in addition to the activities of their planning?"
	Scenario	"⟨Employee i^*⟩ is not performing ⟨task j^*⟩ in addition to the activities of their planning, but what if ⟨changes applied to instance $\mathcal{I}$ parameter values⟩?"
	Counterfactual	"How to make ⟨employee i^*⟩ perform ⟨task j^*⟩ in addition to the activities of their planning, considering ⟨alterations that can be applied to instance $\mathcal{I}$ parameter values⟩?"
⋮	⋮	⋮
(Ord,E)	Contrastive	"Why is ⟨employee i^*⟩ not performing the activities of their planning in a different order?"
	Scenario	"⟨Employee i^*⟩ is performing the activities of their planning in a certain order, but what if ⟨changes applied to instance $\mathcal{I}$ parameter values⟩?"
	Counterfactual	"How to make ⟨employee i^*⟩ perform the activities of their planning in a different order, considering ⟨alterations that can be applied to instance $\mathcal{I}$ parameter values⟩?"

Table 4.2: List of question templates based on the list of observation templates presented in Table 4.1. Each question template is characterized by a label, a type and a template text.

In this subsection, we described step (B) of the explanation process represented Figure 4.2: we presented how questions of three different types, either contrastive, scenario or counterfactual, can be built from end-users observations about a solution, and we provided a list of templates in order to build such questions. From now on, we consider that end-users ask questions built thanks to the templates of Table 4.1 in order to request explanations about their solution. In the next section, we present how in our approach such questions are then processed through a series of mathematical steps corresponding to the causality diagnosis, from (C) to (E) in Figure 4.2, leading to the definition of explanations.

4.3 Mathematical steps - From questions to explanations

4.3.1 Decision-problem interpreted questions

Assume that end-users have defined a question q about a solution $\mathcal{S}$, thanks to one of the templates of Table 4.2. While $\mathcal{S}$ may be seen by end-users as a group of routes and schedules, represented in a pair of graph and chart like in Figure 3.1, $\mathcal{S}$ is first and foremost a solution of a CO-problem instance. In other words, $\mathcal{S}$ is an object from the field of CO, obtained *a priori* through an optimization process. Therefore, while q is expressed in common language by end-users, it seems opportune to consider q through the prism of CO and look for an explanation answering q using CO-related concepts such as feasibility, optimality, neighborhood, *etc.* That is why we propose that the next step towards determining an explanation, step (C) in Figure 4.2, be to translate q into a question involving mathematical terms from the field of optimization. Performing this translation leads us to introduce a generic decision problem. Thus, we call such rephrasing of q, the decision-problem interpreted question associated with q, noted $dpq(q)$.

Before presenting $dpq(q)$, we first need to introduce the generic notions of neighborhood and decision problem related to an observation, which are involved in the definition of $dpq(q)$. We recall that we introduce the notion of neighborhood of a solution $\mathcal{S}$ related to a transformation tf in Section 3.3 as the subset of solutions, noted $\mathcal{N}(tf, \mathcal{S})$ produced by applying this transformation on $\mathcal{S}$.

Neighborhood related to an observation. Given an observation o about a solution $\mathcal{S}$, we call *neighborhood of $\mathcal{S}$ related to the observation o* the subset of solutions obtained by applying the transformation tf related to o on $\mathcal{S}$. As there is a correspondence between observations and transformations, we make a slight abuse of the neighborhood notation and note $\mathcal{N}(o, \mathcal{S})$ the neighborhood of $\mathcal{S}$ related to the observation o, using o instead of tf in $\mathcal{N}(tf, \mathcal{S})$. We recall that we name the solutions in $\mathcal{N}(o, \mathcal{S})$ *neighbors* or *neighboring solutions* of $\mathcal{S}$.

Let us illustrate again this notion of neighborhood, but here related to an observation. Consider the feasible solution $\mathcal{S}$ represented in Figure 4.1, more specifically Ellen's planning $(\mathcal{R}_1, \mathcal{C}_1)$. An observation about $\mathcal{S}$ based on (Ins,C) template is o_1 = "Ellen is not performing task 27 just after task 17". Then, the neighborhood $\mathcal{N}(o_1, \mathcal{S})$ related to o_1 is the set made of all the solutions $\mathcal{S}' = ((\mathcal{R}'_i, \mathcal{C}'_i))_{i \in \mathcal{E}}$ such that:

- the planning $(\mathcal{R}'_i, \mathcal{C}'_i)$ of any employee $i \in \mathcal{E}$ other than Ellen is the same as $(\mathcal{R}_i, \mathcal{C}_i)$ in $\mathcal{S}$;
- Ellen's planning $(\mathcal{R}'_1, \mathcal{C}'_1)$ is route-equal (see Subsection 3.2.1) to the one obtained from $(\mathcal{R}_1, \mathcal{C}_1)$ by inserting task 27 between tasks 17 and 8 in $\mathcal{R}_1$ and the start time st_{27}, with an arbitrary value, between st_{17} and st_8 in $\mathcal{C}_1$.

This neighborhood $\mathcal{N}(o_1, \mathcal{S})$ is represented in Figure 4.3. Ellen's route now includes task 27 between tasks 17 and 8, as it is indicated with dashed lines. Her schedule is a tuple of start times $(st_{d_1}, st_7, st_{30}, st_3, st_{26}, st_1, st_{17}, st_{27}, st_8, st_{r_1})$, whose values may be arbitrarily chosen. In order to translate it graphically, Ellen's schedule is represented by a large dashed rectangle including all the tasks performed by her.

$\mathcal{N}(o_1, \mathcal{S}) =$

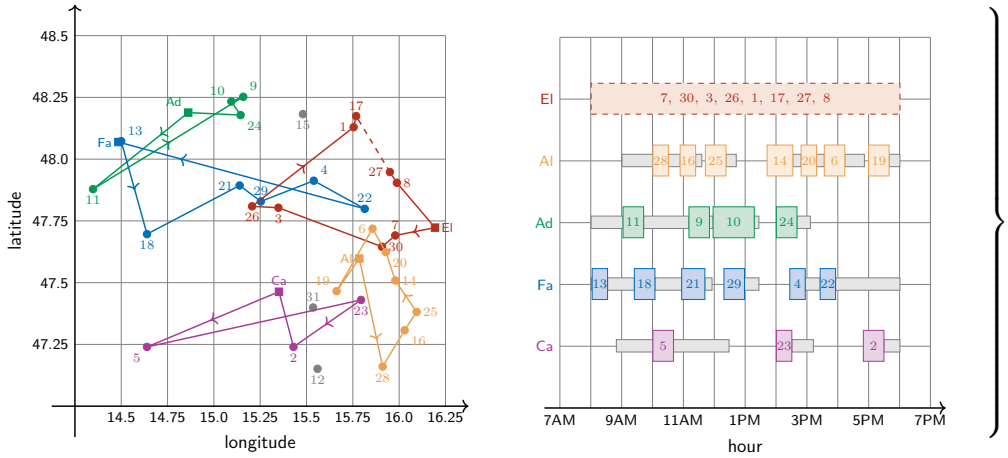


Figure 4.3: Neighborhood $\mathcal{N}(o_1, \mathcal{S})$ of the solution $\mathcal{S}$ represented in Figure 3.1 related to observation o_1 = "Ellen is not performing task 27 just after task 17".

Decision problem related to an observation. Given an end-user observation o about $\mathcal{S}$, feasible solution of instance $\mathcal{I}$, we call *decision problem (related to the observation o)*, noted $dp(o, \mathcal{S}, \mathcal{I})$, the yes-no problem: "Is there a neighboring solution $\mathcal{S}' \in \mathcal{N}(o, \mathcal{S})$ that is feasible and better than $\mathcal{S}$ w.r.t. $\mathcal{I}$?". There are two possible outcomes to $dp(o, \mathcal{S}, \mathcal{I})$:

- a *positive outcome*, when the answer is "yes", which we note $dp(o, \mathcal{S}, \mathcal{I})$ = "yes";
- a *negative outcome*, when the answer is "no", which we note $dp(o, \mathcal{S}, \mathcal{I})$ = "no".

Note that if $\mathcal{S}$ is a solution obtained thanks to a WSRP-solving system, $\mathcal{S}$ should be a good solution, possibly even an optimal one, with respect to its instance $\mathcal{I}$. Then, the outcome to $dp(o, \mathcal{S}, \mathcal{I})$ is more likely to be negative than positive. Still, since the WSRP is NP-hard, optimization systems solving the WSRP are generally based on approximate optimization methods, which produce good but sub-optimal solutions. In this case, it may be possible to improve these solutions by applying transformations (*e.g.* insertion, exchange, reordering) such as the ones related to the end-user observation o. Thus, we cannot exclude the possibility of getting a positive outcome to $dp(o, \mathcal{S}, \mathcal{I})$. Besides, if we now consider $\mathcal{I}'$, a relaxation of $\mathcal{I}$, getting a positive outcome to $dp(o, \mathcal{S}, \mathcal{I}')$ is even more likely to happen.

Now that the generic notion of decision problem has been defined, we can introduce the decision-problem interpreted question $dpq(q)$ associated with an end-user question q.

Decision-problem interpreted question. Given a question q, we define the *decision-problem interpreted question* associated with q, noted $dpq(q)$, as follows.

- If q is a contrastive question, *i.e.* $q = q_c(o, \mathcal{S}, \mathcal{I})$, then
  $dpq(q) =$ "$\underbrace{\text{Is there a neighboring solution } \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \text{ that is feasible and better than } \mathcal{S} \text{ w.r.t. } \mathcal{I}?}_{= dp(o, \mathcal{S}, \mathcal{I})}$
  - If there is such an $\mathcal{S}'$, why not considering $\mathcal{S}'$ instead of $\mathcal{S}$?
  - If there is not, why?"
- If q is a scenario question, *i.e.* $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$, with $\mathcal{I}'$ relaxation of $\mathcal{I}$, then
  $dpq(q) =$ "$\underbrace{\text{Is there a neighboring solution } \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \text{ that is feasible and better than } \mathcal{S} \text{ w.r.t. } \mathcal{I}'?}_{= dp(o, \mathcal{S}, \mathcal{I}')}$
  - If there is such an $\mathcal{S}'$, what is $\mathcal{S}'$ content?
  - If there is not, why?"
- If q is a counterfactual question, *i.e.* $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, with $\mathcal{J}'$ set of relaxations of $\mathcal{I}$, then
  $dpq(q) =$ "Is there $\mathcal{I}' \in \mathcal{J}'$ s.t. $\underbrace{\text{there exists } \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \text{ that is feasible and better than } \mathcal{S} \text{ w.r.t. } \mathcal{I}'}_{= (dp(o, \mathcal{S}, \mathcal{I}') = \text{"yes"})}$?
  - If there is such an $\mathcal{I}'$, what are $\mathcal{I}'$ and $\mathcal{S}'$ contents?
  - If there is not, why?"

Thus, $dpq(q)$ can be seen as another way to express q using mathematical terms involving one or several decision problems $dp(.)$. However, in contrast to q, $dpq(q)$ explicitly states what we need to find for answering q: we need to determine whether there exists a neighboring solution $\mathcal{S}'$, which is feasible and better than the original solution $\mathcal{S}$, with respect to some instance (either the original instance $\mathcal{I}$ or a relaxation $\mathcal{I}'$). Two cases can occur:

- the *positive case*, when there exists such a neighboring solution $\mathcal{S}'$, in which case the answer to $dpq(q)$ essentially consists in presenting $\mathcal{S}'$.
- the *negative case*, when no such solutions exist, then reasons must be found.

As we did it in the above paragraph, we will sometimes refer to $\mathcal{I}$ and $\mathcal{S}$ as the *original* instance and solution to distinguish them from other solutions and instances, such as the neighboring solution $\mathcal{S}'$ and the relaxation $\mathcal{I}'$ involved in the definition of $dpq(q)$.

Two comments can be made about decision-problem interpreted questions.

First, in line with our remark on the possibility of obtaining a positive or negative outcome, when $\mathcal{S}$ is a solution that has been obtained thanks to an optimization system solving an instance $\mathcal{I}$, then: if q is a contrastive question, it is very likely that the answer to $dpq(q)$ falls into the negative case; if q is a scenario or a counterfactual question, depending on the alterations applied to $\mathcal{I}$, the answer to $dpq(q)$ may fall in either cases.

Second, the definition of a decision-problem interpreted question for scenario questions is almost a particular case of the one for counterfactual questions: indeed $dpq(q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}'))$ almost coincide with $dpq(q_h(o, \mathcal{S}, \mathcal{I}, \{\mathcal{I}'\}))$. This makes sense since a scenario question considers a unique user-defined relaxation while the counterfactual question considers a set of user-defined relaxations. Furthermore, the definition of a decision-problem interpreted question for scenario questions is also close to the one for contrastive questions: indeed $dpq(q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}'))$ almost coincide with $dpq(q_c(o, \mathcal{S}, \mathcal{I}'))$. As we will see it in Chapter 5, this will influence the way we compute scenario explanation texts since we will be able to use computation techniques very similar to the ones used to generate contrastive explanation texts.

Even though $dpq(q)$ 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 (D), in Figure 4.2, which consists in rephrasing $dpq(q)$ into another question involving an ILP model. Next subsection focuses on this step.

$$lex\ max\ \Big(\sum_{i\in\mathcal{E}}\sum_{j\in\mathcal{T}}\sum_{\substack{k\in\mathcal{A}_i\\k\neq d_i,j}} U_{ijk}\ dt_j,\ -\sum_{i\in\mathcal{E}}\sum_{\substack{j\in\mathcal{A}_i\\j\neq r_i}}\sum_{\substack{k\in\mathcal{A}_i\\k\neq d_i,j}} U_{ijk}\ tr_{jk}\Big) \qquad \text{(M2.1)} \equiv \text{(M1.1)}$$

$s.t.$

$$\sum_{\substack{k\in\mathcal{A}_i\\k\neq d_i}} U_{i,d_i,k} = 1 \qquad \forall\ i\in\mathcal{E} \qquad \text{(M2.2)} \equiv \text{(M1.2)}$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$T_j + dt_j \leq \sum_{i\in\mathcal{E}} U_{i,j,r_i}(ube_i - tr_{jr_i}) + \Big(1 - \sum_{i\in\mathcal{E}} U_{i,j,r_i}\Big) ubt_j \qquad \forall\ j\in\mathcal{T} \qquad \text{(M2.11)} \equiv \text{(M1.11)}$$

$$\varphi(\mathcal{X}) \in \mathcal{N}(o,\mathcal{S}) \qquad \text{(M2.12)}$$

$$\Big(\sum_{i\in\mathcal{E}}\sum_{j\in\mathcal{T}}\sum_{\substack{k\in\mathcal{A}_i\\k\neq d_i,j}} U_{ijk}\ dt_j,\ -\sum_{i\in\mathcal{E}}\sum_{\substack{j\in\mathcal{A}_i\\j\neq r_i}}\sum_{\substack{k\in\mathcal{A}_i\\k\neq d_i,j}} U_{ijk}\ tr_{jk}\Big) \geq (twt(\mathcal{S}),\ -trt(\mathcal{S})) \qquad \text{(M2.13)}$$

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

$$U_{ijk} \in \{0,1\} \quad \forall\ i\in\mathcal{E},\ \ \forall\ j\in\mathcal{A}_i\setminus\{r_i\},\ \ \forall\ k\in\mathcal{A}_i\setminus\{d_i,j\}$$

Model 2: Foil model $fm(o,\mathcal{S},\mathcal{I})$.

4.3.2 Foil-model interpreted questions

In this subsection, we delve into step (D) of Figure 4.2. The purpose of this step is to leverage ILP modeling in order to rephrase the decision-problem interpreted question $dpq(q)$ into a question referring to one or several ILP models. Thus, we will be able to use such ILP models to identify reasons behind a negative case when it occurs. As we call *foil models* these ILP models, we call *foil-model interpreted question* the rephrased question relying on foil models.

In the paragraphs below, we start by providing the definition of a foil model and tell the reasons behind this name of foil model. Then, we give the definition of the foil-model interpreted question.

Foil model related to an observation. Given an observation o about $\mathcal{S}$, feasible solution of an instance $\mathcal{I}$, we call *foil model*, noted $fm(o,\mathcal{S},\mathcal{I})$, the ILP model obtained by extending the main model $mm(\mathcal{I})$ (see Model 1 in Subsection 3.2.1), so that the answer to $dp(o,\mathcal{S},\mathcal{I})$ matches the one to "Is $fm(o,\mathcal{S},\mathcal{I})$ feasible?". We recall that there are two sets of decision variables in $mm(\mathcal{I})$: 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; for each employee $i\in\mathcal{E}$ and each pair of activities $(j,k)\in(\mathcal{A}_i\setminus\{r_i\})\times(\mathcal{A}_i\setminus\{d_i,j\})$, 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. As shown in Model 2, $fm(o,\mathcal{S},\mathcal{I})$ is obtained from $mm(\mathcal{I})$ by keeping the same decision variables, objective function and constraints and by adding new constraints defined as follows.

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

We give the name of foil model to $fm(o,\mathcal{S},\mathcal{I})$ with reference to the terminology related to contrastive questions and explanations. As seen in Section 2.3, a contrastive question has the following form "Why is this fact instead of this foil?", where "this fact" is something observed, something which occurs, and "that foil" is an alternative to it, something which could have occurred. The foil in the contrastive question $q_c(o,\mathcal{S},\mathcal{I})$ = "Why o is observed in $\mathcal{S}$, solution of $\mathcal{I}$?" is $\neg o$, which is observed in the neighboring solutions of $\mathcal{N}(o,\mathcal{S})$. Thus, as $fm(o,\mathcal{S},\mathcal{I})$ enforces solutions to be in $\mathcal{N}(o,\mathcal{S})$, we found the term "foil model" to be aptly descriptive.

Let us illustrate neighborhood constraints on an example. Consider again $\mathcal{S}$ represented in Figure 4.1, feasible solution of $\mathcal{I}$, as well as the observation o_1 = "Ellen is not performing task 27 just after task 17". Then, neighborhood constraints in $fm(o_1,\mathcal{S},\mathcal{I})$ are the following:

- for $i\in\mathcal{E}\setminus\{1\}$, all the variables (U_{ijk}) and all the variables (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.

Now that we have introduced the foil model $fm(o, \mathcal{S}, \mathcal{I})$, we can continue with the definition of the foil-model interpreted question, which we denote $fmq(q)$. Essentially, $fmq(q)$ rephrases the decision-problem interpreted question $dpq(q)$ by replacing $dp(o, \mathcal{S}, \mathcal{I})$ with "Is $fm(o, \mathcal{S}, \mathcal{I})$ feasible?". Then, similarly to $dpq(q)$, $fmq(q)$ structure highlights that there are two cases when answering q: positive and negative.

Foil-model interpreted question. Given a question q, we define the *foil-problem interpreted question* associated with q, noted $fmq(q)$, as follows.

- If q is a contrastive question, *i.e.* $q = q_c(o, \mathcal{S}, \mathcal{I})$, then
  $fmq(q) =$ "Is $fm(o, \mathcal{S}, \mathcal{I})$ feasible?
  - If it is feasible, *i.e.* there exists $\mathcal{X}$, feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I})$, why not considering $\mathcal{S}' = \varphi(\mathcal{X})$ instead of $\mathcal{S}$?
  - If not, why?"
- If q is a scenario question, *i.e.* $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$, then
  $fmq(q) =$ "Is $fm(o, \mathcal{S}, \mathcal{I}')$ feasible?
  - If it is feasible, *i.e.* there exists $\mathcal{X}$, feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$, what is $\mathcal{S}' = \varphi(\mathcal{X})$ content?
  - If not, why?"
- If q is a counterfactual question, *i.e.* $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, then
  $fmq(q) =$ "Is there $\mathcal{I}' \in \mathcal{J}'$ s.t. $fm(o, \mathcal{S}, \mathcal{I}')$ is feasible?
  - If there is such $\mathcal{I}'$, with $\mathcal{X}$, feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$, what are $\mathcal{I}$ and $\varphi(\mathcal{X}) = \mathcal{S}'$ contents?
  - If there is not, why?"

Thanks to $fmq(q)$, it is clearer where to find reasons when a negative case arises: we should look for infeasible subsets of constraints making the foil model(s) infeasible. Thus, we can now move on to the next step and define the notion of explanation in our context.

4.3.3 Explanations

This last subsection focuses on step (E) of Figure 4.2 and presents the way we model explanations. In line with the fact that the foil-model interpreted question $fmq(q)$ may have either a positive or a negative answer, we define two kinds of explanations: positive and negative explanations.

Before defining these two kinds of explanations, let us give some intuitions about them. Consider that end-users make an observation o about the solution $\mathcal{S}$ and raise a question q (either contrastive $q = q_c(o, \mathcal{S}, \mathcal{I})$, scenario $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$ or counterfactual $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$. If it exists, a positive explanation basically confirms that end-users are right to wonder about observing o in $\mathcal{S}$ because there exists a neighboring solution $\mathcal{S}'$ that is feasible and better than $\mathcal{S}$ w.r.t. a certain instance (either $\mathcal{I}$, $\mathcal{I}'$ an end-user-defined relaxation of $\mathcal{I}$ or $\mathcal{I}' \in \mathcal{J}'$ a relaxation to identify in the end-user-defined set $\mathcal{J}'$). On the opposite, a negative explanation basically confirms that $\mathcal{S}$ is a good quality solution because none of the neighboring solutions of $\mathcal{N}(o, \mathcal{S})$ are feasible and better than $\mathcal{S}$ w.r.t. to certain instances, and it provides a justification to that affirmation.

We now give the definition of positive explanations.

Positive explanation. A *positive explanation* $x(q)$, answering a question q, is defined when the answer to $fmq(q)$ falls into the positive case.

- If q is a contrastive question, *i.e.* $q = q_c(o, \mathcal{S}, \mathcal{I})$, then $x(q)$ is

  because $\mathcal{S}$ was not an optimal solution *then* o could be observed in $\mathcal{S}$,

  however $\neg o$ can be observed in $\mathcal{S}'$, with $\mathcal{S}'$ feasible w.r.t. $\mathcal{I}$ and $\mathcal{S}' \geq_{\mathcal{I}} \mathcal{S}$

  where $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{X}$ feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I})$.
- If q is a scenario question, *i.e.* $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$, then $x(q)$ is

  o is observed in $\mathcal{S}$, solution of $\mathcal{I}$ *but* by changing $\mathcal{I}$ into $\mathcal{I}'$

  then $\neg o$ can be observed, for instance in $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{S}'$ feasible w.r.t. $\mathcal{I}'$ and $\mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}$

  where $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{X}$ feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$.
- If q is a counterfactual question, $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, then $x(q)$ is

  o is observed in $\mathcal{S}$, solution of $\mathcal{I}$ *but* by changing $\mathcal{I}$ into $\mathcal{I}' \in \mathcal{J}'$

  then $\neg o$ can be observed, for instance in $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{S}'$ feasible w.r.t. $\mathcal{I}'$ and $\mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}$

  where $\mathcal{I}' \in \mathcal{J}'$ and $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{X}$ feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$.

On the opposite, when answering $fmq(q)$ falls into the negative case, a negative explanation is given. We consider two kinds of negative explanations. The first kind aims to provide an exhaustive and complete justification to the negative case. In other words, it aims at providing a proof. The second kind seeks to provide a convincing example to illustrate the negative case, not a mathematical proof of it. In other words, it seeks to provide an argument. In the following paragraphs, we define these two kinds of negative explanations. For proof-like ones, we resort to the notion of *infeasible subset of constraints* of an ILP model, *i.e.* a subset of constraints such that the set of solutions satisfying these constraints is empty.

Proof-like negative explanation. A *proof-like negative explanation* $x(q)$, answering a question q, can be defined when the answer to $fmq(q)$ falls into the negative case.

- If q is a contrastive question, *i.e.* $q = q_c(o, \mathcal{S}, \mathcal{I})$, then $x(q)$ is
$$\mathcal{U} \text{ is an infeasible subset of constraints of } fm(o, \mathcal{S}, \mathcal{I}),$$
$$\textit{therefore not } (\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \text{ feasible w.r.t. } \mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}} \mathcal{S}).$$
- If q is a scenario question, *i.e.* $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$, then $x(q)$ is
$$\mathcal{U} \text{ is an infeasible subset of constraints of } fm(o, \mathcal{S}, \mathcal{I}'),$$
$$\textit{therefore not } (\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \text{ feasible w.r.t. } \mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}).$$
- If q is a counterfactual question, $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, then $x(q)$ is
$$\text{for all } \mathcal{I}' \in \mathcal{J}',\ \text{there exists } \mathcal{U} \text{ an infeasible subset of constraints of } fm(o, \mathcal{S}, \mathcal{I}'),$$
$$\textit{therefore not } (\exists\ \mathcal{I}' \in \mathcal{J}',\ \exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \text{feasible w.r.t. } \mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}).$$

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

Argument-like negative explanation. An *argument-like negative explanation* $x(q)$, answering a question q, can be defined when the answer to $fmq(q)$ falls into the negative case.

- If q is a contrastive question, *i.e.* $q = q_c(o, \mathcal{S}, \mathcal{I})$, then $x(q)$ is
$$\text{not } (\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \text{ feasible w.r.t. } \mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}} \mathcal{S}).$$
$$\textit{For example, } \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \textit{ but not } (\text{feasible w.r.t. } \mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}} \mathcal{S}).$$
- If q is a scenario question, *i.e.* $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$, then $x(q)$ is
$$\text{not } (\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \text{ feasible w.r.t. } \mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}).$$
$$\textit{For example, } \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \textit{ but not } (\text{feasible w.r.t. } \mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}).$$
- If q is a counterfactual question, $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, then $x(q)$ is
$$\text{not } (\exists\ \mathcal{I}' \in \mathcal{J}',\ \exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \text{feasible w.r.t. } \mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}).$$
$$\textit{For example, } \mathcal{I}' \in \mathcal{N}(o, \mathcal{S}),\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \textit{ but not } (\text{feasible w.r.t. } \mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}).$$

Note that in this definition, the neighboring solution $\mathcal{S}' \in \mathcal{N}(o, \mathcal{S})$ that is used as an example is not required to satisfy any property, *e.g.* some kind of optimality. However, in order to be convincing, we need to find a striking example for $\mathcal{S}'$. In Chapter 5, we will discuss how we choose and compute such a striking example for $\mathcal{S}'$. Besides, a similar comment can be made about the instance relaxation $\mathcal{I}'$ selected in the set $\mathcal{J}'$ that is involved in the counterfactual explanation.

4.4 Conclusion

Throughout this chapter, we described steps from (A) to (E) of the explanation process depicted in Figure 4.2. These steps lead from an end-user *observation* o about a solution $\mathcal{S}$, to a *question* q and finally to an *explanation* $x(q)$. We introduce the notions of *positive*, *proof-like negative* and *argument-like negative* explanations. Questions and their corresponding explanations can be of three types: either *contrastive*, *scenario* or *counterfactual*. Table 4.3 provides a concise comparison of how we model these steps for each of these types.

At step (E), explanations are essentially formulated in mathematical terms. Especially, proof-like negative explanations are based on infeasible subsets of constraints, a concept which has no practical meaning for non-experts in combinatorial optimization. Therefore, these explanations are not suitable products to communicate to end-users (see Section 2.2). Besides, we have not yet discussed how to obtain in practice the pieces of information that are involved in such explanations. In other words, we have not yet discussed how to compute an explanation. Thus, in Chapter 5, we will describe how explanations can be both computed and narrated into texts for end-users.

		Contrastive	Scenario	Counterfactual
Question	q	$q = q_c(o, \mathcal{S}, \mathcal{I}) =$ $\begin{cases} \text{“Why } o \text{ is observed in } \mathcal{S},\\ \text{solution of } \mathcal{I}?\text{”} \end{cases}$	$q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}') =$ $\begin{cases} \text{“}o \text{ is observed in } \mathcal{S}, \text{ solution of } \mathcal{I},\\ \text{but what if } \mathcal{I} \text{ is changed into } \mathcal{I}'?\text{”} \end{cases}$ where the instance $\mathcal{I}'$ is obtained by relaxing $\mathcal{I}$	$q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}') =$ $\begin{cases} \text{“}o \text{ is observed in } \mathcal{S}, \text{ solution of } \mathcal{I},\\ \text{but how to observe } \neg o \text{ considering that } \mathcal{I}\\ \text{can be changed into } \mathcal{I}' \in \mathcal{J}'?\text{”} \end{cases}$ where $\mathcal{J}'$ is a set of instances obtained by relaxing $\mathcal{I}$
Neighborhood	$\mathcal{N}(o, \mathcal{S})$	Set of solutions obtained from $\mathcal{S}$ by applying the transformation related to o		
Decision pb.	$dp(o, \mathcal{S}, \mathcal{I})$	“Is there a neighboring solution $\mathcal{S}' \in \mathcal{N}(o, \mathcal{S})$ that is feasible and better than $\mathcal{S}$ w.r.t. $\mathcal{I}$?”		
Decision-pb. interpreted qu.	$dpq(q)$	$dp(o, \mathcal{S}, \mathcal{I}) +$ “If there is such an $\mathcal{S}'$, why not considering $\mathcal{S}'$ instead of $\mathcal{S}$? If there is not, why?”	$dp(o, \mathcal{S}, \mathcal{I}') +$ “If there is such an $\mathcal{S}'$, what is $\mathcal{S}'$ content? If there is not, why?”	“Is there $\mathcal{I}' \in \mathcal{J}'$ s.t. $(dp(o, \mathcal{S}, \mathcal{I}') =$ ‘yes’)? If there is such an $\mathcal{I}'$, what are $\mathcal{I}'$ and $\mathcal{S}'$ contents? If there is not, why?”
Foil model	$fm(o, \mathcal{S}, \mathcal{I})$	$fm(o, \mathcal{S}, \mathcal{I})$ obtained from $mm(\mathcal{I})$ by adding neighboring constraints based on $\mathcal{N}(o, \mathcal{S})$ and by adding an improvement constraint		
Foil-model interpreted qu.	$fmq(q)$	“Is $fm(o, \mathcal{S}, \mathcal{I})$ feasible? If it feasible, *i.e.* there exists $\mathcal{X}$, feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I})$, why not considering $\mathcal{S}' = \varphi(\mathcal{X})$ instead of $\mathcal{S}$? If not, why?”	“Is $fm(o, \mathcal{S}, \mathcal{I}')$ feasible? If it feasible, *i.e.* there exists $\mathcal{X}$, feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$, what is $\varphi(\mathcal{X}) = \mathcal{S}'$ content? If not, why?”	“Is there $\mathcal{I}' \in \mathcal{J}'$ s.t. $fm(o, \mathcal{S}, \mathcal{I}')$ is feasible? If there is such $\mathcal{I}'$, with $\mathcal{X}$, feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$, what are $\mathcal{I}$ and $\varphi(\mathcal{X}) = \mathcal{S}'$ contents? If there is not, why?”
Positive explanation	$x(q)$	*because* $\mathcal{S}$ was actually not an optimal *then* o could be observed in $\mathcal{S}$, *however* $\neg o$ can be observed in $\mathcal{S}'$, with $\mathcal{S}'$ feasible w.r.t. $\mathcal{I}$ and $\mathcal{S}' \geq_{\mathcal{I}} \mathcal{S}$ (where $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{X}$ feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I})$)	o is observed in $\mathcal{S}$, solution of $\mathcal{I}$ *but* by changing $\mathcal{I}$ into $\mathcal{I}'$ *then* $\neg o$ can be observed, for instance in $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{S}'$ feasible w.r.t. $\mathcal{I}'$ and $\mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}$ (where $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{X}$ feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$)	o is observed in $\mathcal{S}$, solution of $\mathcal{I}$ *but* by changing $\mathcal{I}$ into $\mathcal{I}' \in \mathcal{J}'$ *then* $\neg o$ can be observed, for instance in $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{S}'$ feasible w.r.t. $\mathcal{I}'$ and $\mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}$ (where $\mathcal{I}' \in \mathcal{J}'$ and $\mathcal{S}' = \varphi(\mathcal{X})$ with $\mathcal{X}$ feasible ILP-solution of $fm(o, \mathcal{S}, \mathcal{I}')$)
Proof-like negative explanation	$x(q)$	$\mathcal{U}$ is an infeasible subset of constraints of $fm(o, \mathcal{S}, \mathcal{I})$ *therefore* not $(\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \mathcal{S}'$ feasible w.r.t. $\mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}} \mathcal{S})$	$\mathcal{U}$ is an infeasible subset of constraints of $fm(o, \mathcal{S}, \mathcal{I}')$ *therefore* not $(\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid \mathcal{S}'$ feasible w.r.t. $\mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S})$	for all $\mathcal{I}' \in \mathcal{J}'$, there exists $\mathcal{U}$ an infeasible subset of constraints of $fm(o, \mathcal{S}, \mathcal{I}')$ *therefore* not $(\exists\ \mathcal{I}' \in \mathcal{J}',\ \exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid$ feasible w.r.t. $\mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S})$
Argument-like negative explanation	$x(q)$	not $(\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid$ feasible w.r.t. $\mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}} \mathcal{S})$. *For instance*, $\mathcal{S}' \in \mathcal{N}(o, \mathcal{S})$ *but* not (feasible w.r.t. $\mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}} \mathcal{S}$)	not $(\exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid$ feasible w.r.t. $\mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S})$. *For instance*, $\mathcal{S}' \in \mathcal{N}(o, \mathcal{S})$ *but* not (feasible w.r.t. $\mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}$).	not $(\exists\ \mathcal{I}' \in \mathcal{J}',\ \exists\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S}) \mid$ feasible w.r.t. $\mathcal{I}',\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S})$. *For instance*, $\mathcal{I}' \in \mathcal{N}(o, \mathcal{S}),\ \mathcal{S}' \in \mathcal{N}(o, \mathcal{S})$ *but* not (feasible w.r.t. $\mathcal{I},\ \mathcal{S}' \geq_{\mathcal{I}'} \mathcal{S}$).

Table 4.3: Summary of the main concepts involved in the modeling of the question-to-explanation pathway for each of the three types of explanations.

Chapter 5 Approach for generating explanation texts

5.1 Introduction

As described in Chapter 3, this thesis aims at developing techniques for explaining solutions of an optimization problem to the end-users of a system solving this problem. In Section 3.2, we specified the optimization problem we focus on: a use case of *Workforce Scheduling and Routing Problem* (*WSRP*), modeled as a bi-objective Integer Linear Program (ILP) which we refer to as *main model*. In Section 3.3, we detailed various *transformations* of solutions (*e.g.* given the solution represented in Figure 3.1, by inserting task 27 in Ellen's planning between every pair of consecutive activities). Such transformations define solution *neighborhoods* (*e.g.* the solutions obtained by applying the above-mentioned insertion).

In Chapter 4, we described our approach for modeling mathematically explanations about solutions. It is summarized in Figure 4.2. Suppose that end-users have a solution that is feasible with respect to an instance and want to obtain explanations about this solution. We usually refer to the end-users' solution and instance as the *current solution* and the *current instance*, by opposition to other solutions and instances that may be identified as part of the explanations. The starting point of our approach is an *observation*, made by end-users, about the current solution (*e.g.* "Ellen is not performing task 27 between any pair of consecutive activities of her planning"). Observations are supposed to be related to a solution transformation (*e.g.* the above-mentioned insertion) (see Subsection 4.2.1). Then, from their observation, end-users express a *question* (*e.g.* "Why is Ellen not performing task 27 between any pair of consecutive activities of her planning?"). We considered three forms of questions corresponding to three types of explanations (see Subsection 4.2.2). First, "why" questions are associated with *contrastive explanations*, whose aim is to clarify why one fact (the observation made about the current solution) occurred in contrast to another. "What if" questions are associated with *scenario explanations* describing how changes in the parameters of the current instance affect the observation made about the current solution. Changes of instance parameters are assumed to be set by end-users and be such that they define a *relaxation* of the current instance, *i.e.* an instance whose set of feasible solutions includes the one of the current instance. Third, "How to" questions are associated with *counterfactual explanations* which aim at identifying what parameters could be changed in the current instance and by how much, within a certain range, so as to make a solution expected by end-users feasible and better than the current one. The allowed changes of instance parameters, in number and magnitude, are assumed to be defined by end-users and to be such that they define again relaxations of the current instance. Finally, in our modeling approach, a series of mathematical steps leads from the end-user question to an *explanation* expressed in mathematical terms (*e.g.* "Ellen is not performing task 27 between any pair of consecutive activities of her planning" because a subset of constraints of an ILP model, that we called *foil model*, is infeasible) (see Subsection 4.3.3).

Chapter 4 dealt with the explanation process from observations to mathematical explanations, spanning from step (A) to (E) in Figure 4.2. However, explanations expressed in mathematical terms (*e.g.* using infeasible subsets of constraints) are not intelligible for end-users. Thus, in accordance with our guideline (G1) presented in Section 2.2, we must address the crucial step (F) of producing *explanation texts* that are adapted to end-users. In this chapter, we elaborate on our methods for generating automatically user-friendly explanation texts. While the methods in [LGMO22, LGMO23] focused on contrastive and counterfactual explanation types, this chapter extends them to encompass all three explanation types.

A key notion in our generation approach is the one of *support content*, which is either a *support solution* in the case of contrastive and scenario explanations, or a *support relaxation-solution pair* in the case of counterfactual ones. Suppose that, in addition to the current instance and solution, end-users have a question based on an observation about this solution. Since observations are assumed to be related to transformations that define neighborhoods, this question can be related to a neighborhood. Then, intuitively, the support solution is the "best" neighboring solution. It is defined as follows: among all the solutions of this neighborhood, the support solution is the best feasible one or, in the absence of feasible solutions in the neighborhood, it is the *nearest-to-feasibility* one - where the notion of distance to feasibility will be specified in this chapter. In the context of contrastive explanations, the feasibility and the quality of the support solution are measured with respect to the current instance; in the context of scenario explanations, they are measured with respect to the relaxation specified by the end-users in their scenario question. As regards counterfactual explanations, the support content is not only made of a solution, the support solution, but also of an instance, the support relaxation. Intuitively the support relaxation is the "best" relaxation, in terms of magnitude and numbers of parameter alterations, associated with the "best" support solution we can get. Regardless of the type of explanations, the purpose of the support contents is to provide explanatory information that can be exploited within explanation texts. Especially, in the case where a support solution is infeasible (with respect to the current instance or a relaxation, depending on the type of explanations), infeasible subsets of constraints can be identified and narrated within explanation texts thanks to the support contents. In our approach, explanation texts are built thanks to *template texts* which are filled with the explanatory information obtained from the support contents. To be consistent with the observation questioned by the end-users, the template texts are also filled in with some observation-dependent predefined texts that we call *typical expressions*.

Finally, in the perspective of integrating the generation of explanation texts within an interactive *system* where end-users ask questions and obtain explanations in return (see Chapter 6), we look for designing algorithms generating explanation texts within a time frame that is compatible with end-users' *near-real-time usage* of explanations. Therefore, in this chapter, we also study the numerical performances of our algorithms for generating explanation texts.

The remainder of this chapter is organized as follows. Section 5.2 is a preliminary section which presents the typical expressions related to observations. Then, from Section 5.3 to Section 5.5, for each of the three types of explanations, namely contrastive, scenario and counterfactual, we describe our approach for generating explanation texts, including the algorithms used for identifying the support contents. Along with these algorithms, we provide some numerical analysis about the performances of these algorithms.

5.2 Typical expressions

Suppose that end-users have a question related to an observation about a solution - the question and observation being based on templates from Tables 4.1 and 4.2. In this chapter, we will show how to build an explanation text answering such a question by concatenating and filling in pieces of template texts. On one hand, in order to produce a final explanation text that is consistent with the question, some parts of the text must refer specifically to its related observation. On the other hand, in order not to design a custom explanation template text for every single observation, every single explanation type (contrastive, scenario or counterfactual) and every single explanation case (positive, proof-like negative, argument-like), we must bring out and exploit some genericity in this wide range of explanation template texts.

To comply with this need to be both specific and generic, we introduce the notion of *typical expressions*. These are texts corresponding to generic ideas (*e.g.* the "neighbors" *i.e.* neighboring solutions) but which can be written in specific terms depending on the observation (*e.g.* the "solutions obtained by inserting ⟨task j^*⟩ just after ⟨activity k^*⟩ in ⟨employee i^*⟩'s planning" for a question based on (Ins,C) observation template).

Typical expressions. Table 5.1 associates each observation label listed in Table 4.1 with several *typical expressions*, which are pairs of generic names and specific template texts. Given an observation, we detail below the essence of the template text corresponding to each generic name.

- "⟨the fact (is observed)⟩": its template text is equivalent to the observation template text - we introduce this typical expression because it helps making the reading of the explanation template texts more natural.
- "⟨meeting the foil⟩": its template text specifies what it is for a solution to comply with the foil - recall that, in the terminology of contrastive explanations, the foil refers to the alternative to the observed fact (see Chapter 3).
- "⟨neighbors⟩": its template text delineates the set of neighboring solutions featuring the foil instead of the observed fact.

Labels	Generic names	Specific template texts
(Ins,C)	"⟨the fact (is observed)⟩"	"⟨task j^*⟩ is not performed by ⟨employee i^*⟩ just after ⟨activity k^*⟩"
	"⟨meeting the foil⟩"	"having ⟨employee i^*⟩ perform ⟨task j^*⟩ just after ⟨activity k^*⟩"
	"⟨neighbors⟩"	"solutions obtained by inserting ⟨task j^*⟩ just after ⟨activity k^*⟩ in ⟨employee i^*⟩'s planning"
(Ins,P,a)	"⟨the fact (is observed)⟩"	"⟨task j^*⟩ is not performed by ⟨employee i^*⟩ between any pair of consecutive activities of their planning"
	"⟨meeting the foil⟩"	"having ⟨employee i^*⟩ perform ⟨task j^*⟩ between two consecutive activities of their planning"
	"⟨neighbors⟩"	"solutions obtained by inserting ⟨task j^*⟩ between every pair of consecutive activities of ⟨employee i^*⟩'s planning"
⋮	⋮	⋮
(Ord,E)	"⟨the fact (is observed)⟩"	"⟨employee i^*⟩ is not performing their activities in any other order"
	"⟨meeting the foil⟩"	"having ⟨employee i^*⟩ perform their activities in another order"
	"⟨neighbors⟩"	"solutions obtained by permuting the activities in ⟨employee i^*⟩'s planning"

Table 5.1: Template texts of typical expressions associated with each observation label (from the list presented in Table 4.1) and each typical expression name.

The purpose of typical expressions is to be used as pieces of text involved in the explanation template texts that we design in this chapter. Thanks to typical expressions, on one hand, we can design observation-independent explanation template texts, by using the generic names of typical expression. On the other, given an observation, these explanation template texts become observation-specific by replacing in it generic names with corresponding specific template texts according to Table 5.1. For instance, suppose that we design an explanation template text containing the sentence "Despite the changes in the instance, ⟨meeting the foil⟩ remains impossible." As such, this explanation template text is observation-independent. Suppose now that we use this explanation template text for answering a question about an (Ins,C) observation, then this sentence becomes "Despite the changes in the instance, having ⟨task j^*⟩ performed by ⟨employee i^*⟩ just after ⟨activity k^*⟩ remains impossible." Now, the explanation template text is observation-specific. Note that the generic names of typical expressions contain the symbol "⟨.⟩", which we use throughout this work for fields of template texts, as they can be interpreted as fields which are replaced by specific contents.

To sum up, thanks to typical expressions, we do not need to prepare specific explanation template text for each possible observation template. We can instead design explanation template texts in a observation-independent fashion by using the generic names of typical expressions as elements of texts. Then, given an observation, these generic names correspond to fields which can be replaced with associated specific template texts, making explanation template texts become observation-specific. In the following sections, typical expressions will be used as part of our approach for building explanation texts.

5.3 Generating contrastive explanation texts

This section focuses on the generation of *contrastive explanations*, *i.e.* explanations aiming at clarifying why one fact occurred in contrast to another (see Section 2.3 and Subsection 4.2.2). Consider that end-users have a feasible solution $\mathcal{S}$ of an instance $\mathcal{I}$, as well as a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$, based on one of the question templates of Table 4.2, with o an observation about $\mathcal{S}$. We recall that observations, such as o, are assumed to be related to solution transformations and therefore solution neighborhoods, such as $\mathcal{N}(o, \mathcal{S})$ (see Subsections 4.2.1 and 4.3.1).

Structure of the generation of contrastive explanation texts. The algorithmic procedure for generating contrastive explanation texts in response to contrastive questions consists in two phases.

- **Phase 1.** *Preliminary checks* are carried out. They check whether conditions, which are *necessary* for the neighborhood $\mathcal{N}(o, \mathcal{S})$ to contain solutions that are feasible and better than $\mathcal{S}$ (with respect to $\mathcal{I}$), are satisfied.
- **Phase 2.** If the preliminary checks are satisfied, then, *complete checks* are performed. They *explore* the neighborhood $\mathcal{N}(o, \mathcal{S})$ in order to look for a solution that is feasible and better than $\mathcal{S}$ (with respect to $\mathcal{I}$).

See Figure 5.1 for a graphical description of this procedure in which the two phases are framed by yellow dashed boxes.

Based on the results of these checks, contrastive explanation texts are built. The texts either correspond to *proof-like negative*, *argument-like negative* or *positive* explanations (see Subsection 4.3.3). We recall that a positive explanation basically confirms that end-users are right to wonder about observing o in $\mathcal{S}$ because there exists a neighboring solution $\mathcal{S}'$ that is feasible and better than $\mathcal{S}$ w.r.t. $\mathcal{I}$. On the opposite, a negative explanation basically confirms that $\mathcal{S}$ is a good quality solution because none of the neighboring solutions of $\mathcal{N}(o, \mathcal{S})$ are feasible and better than $\mathcal{S}$ w.r.t. $\mathcal{I}$ and it provides a justification to that affirmation. More specifically, a proof-like negative explanation provides an exhaustive and complete justification (in other words a proof) while the argument-like negative explanation provides a convincing example to illustrate (in other words an argument).

In the next subsection, we describe preliminary checks of Phase 1. In the two following ones, we deal with complete checks which relate to exploring neighborhoods. Finally, we present numerical experiments.

5.3.1 Preliminary checks

Depending on the contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$, and especially the observation template of o, it may be possible to identify some conditions involving o, $\mathcal{S}$ and $\mathcal{I}$, which are necessary for $\mathcal{N}(o, \mathcal{S})$ to contain solutions that are feasible and better than $\mathcal{S}$. Checking these conditions as preliminary checks is relevant first if they can be computed in polynomial time, without having to build neighboring solutions, and second if they can provide narratable proof-like negative explanations. Let us illustrate such preliminary checks on an example and then provide some general comments.

Let us assume that end-users ask the contrastive question q "Why Fabian is not performing task 18 in addition to the activities of his route?", which is based on (Ins,E) template. For this question, we propose two preliminary checks: one related to skill considerations and one related to time considerations. Note that the solutions in $\mathcal{N}(o, \mathcal{S})$ are such that task 18 is assigned to Fabian.

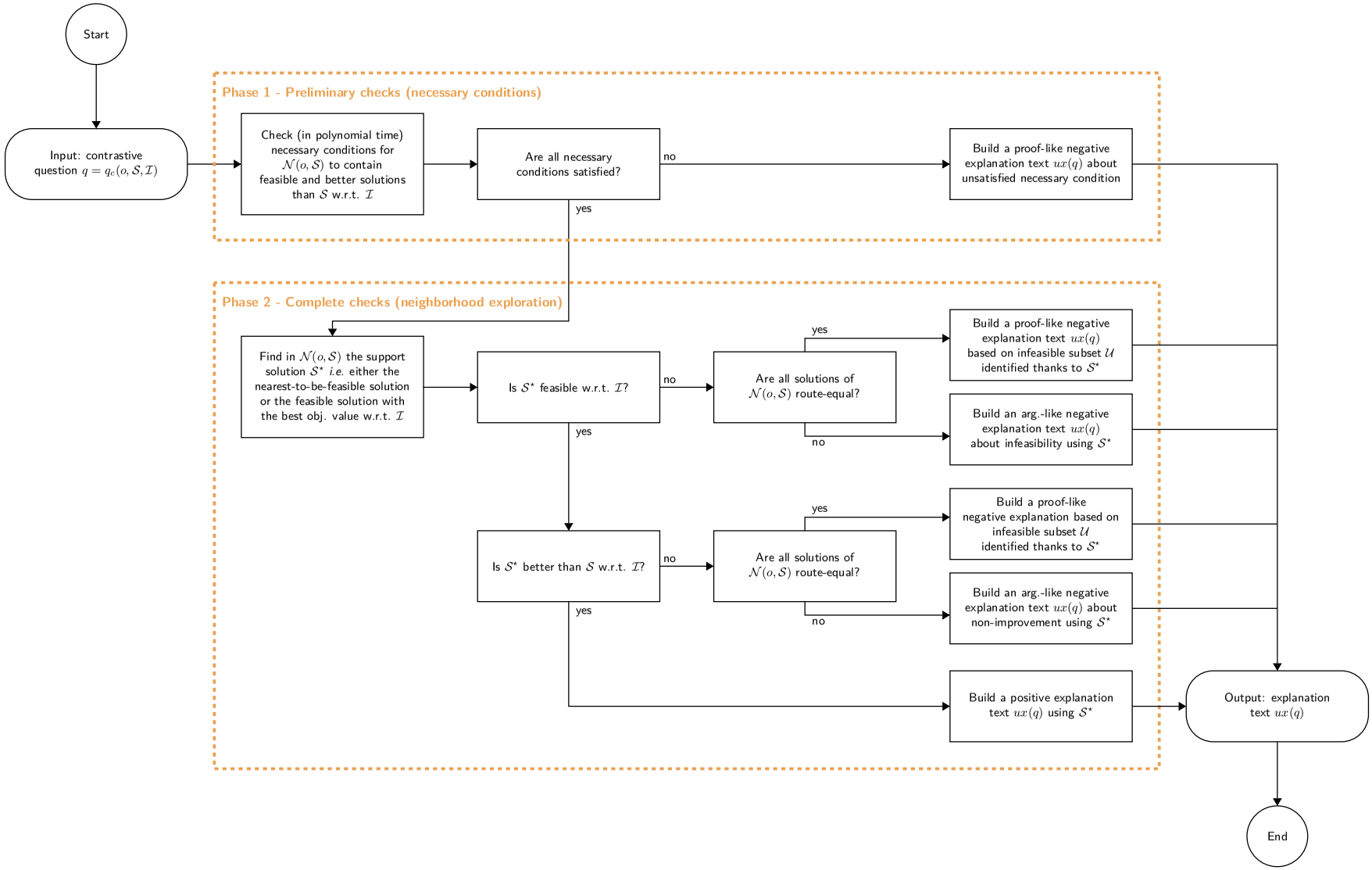


Figure 5.1: Algorithmic framework for generating contrastive explanation texts.

- Regarding skills, a necessary condition for $\mathcal{N}(o, \mathcal{S})$ to contain feasible solutions is that Fabian is skilled enough for performing task 18. Checking this condition can be performed in constant time, without having to build any neighboring solutions. If this condition is not satisfied, then the subset of constraints $\{$ (M2.5), (M2.12) $\}$ of the foil model $fm(o, \mathcal{S}, \mathcal{I})$ is infeasible (see Model 2 in Subsection 4.3.2 for the ILP definition of the foil model). It can be used in a proof-like negative explanation which can be narrated as follows:
  "*In the current solution, Fabian is not performing task* 18 *in addition to the activities of his route because Fabian has a skill level of 1 while the task* 18 *has a skill level of 2*".
- Regarding time, in order to find a feasible solution such that Fabian performs task 18 *in addition* to the tasks of his planning, it is necessary that it exists a feasible solution such that Fabian performs *at least* task 18. In other words, a necessary condition for $\mathcal{N}(o, \mathcal{S})$ to contain feasible solutions is that Fabian must have enough time to leave his home location at the beginning of his working time window, travel to task 18, perform it within its availability time window and come back to his home location before the end of his working time window. Again checking this condition can be performed in constant time, without having to build any neighboring solutions. If this condition is not satisfied, then the subset of constraints $\{$ (M2.2) to (M2.4), (M2.7) to (M2.11) and the constraint among (M2.12) enforcing Fabian to perform task 18 $\}$ of the foil model $fm(o, \mathcal{S}, \mathcal{I})$ is infeasible. It can be used in a proof-like negative explanation which can be narrated as follows: "*In the current solution, Fabian is not performing task* 18 *in addition to the activities of his planning. Even if Fabian task* 18 *was the only task in his planning, Fabian could not perform task* 18 *while it is available. By performing task* 18 *after leaving home, Fabian would end it at the earliest at 1:00PM. However, task* 18 *must be ended by 12PM. Thus, having Fabian perform task* 18 *in addition to the activities of his planning is impossible.*".

More generally, these two preliminary checks can be applied to other observation templates than (Ins,E).

Preliminary checks. Whenever the transformation from $\mathcal{S}$ to any neighboring solution of $\mathcal{N}(o, \mathcal{S})$ requires one task j^* to be assigned to one employee i^* (which includes transformations related to (Ins,C), (Ins,P,a) and (Ins,E) as well as (Ex,C), (Ex,P,a) and (Ex,E) observation templates), we propose to check *two groups of necessary conditions related to skills and to time*. If these conditions are not satisfied, proof-like negative explanation texts are built thanks to template texts using typical expressions as well as data related to the checks computation.

- **Necessary condition regarding skills.** Employee i^* must be skilled enough to perform task j^* *i.e.* $ske_{i^*} \geq skt_{j^*}$. If this condition is not satisfied, then an explanation text can be built with the following template text:
  "*In the current solution,* $\langle$*the fact (is observed)*$\rangle$ *because* $\langle$*employee* $i^*\rangle$ *has a skill level of* $\langle ske_{i^*}\rangle$ *while* $\langle$*task* $j^*\rangle$ *has a skill level of* $\langle skt_{j^*}\rangle$."
- **Necessary conditions regarding time.** There are two conditions to be satisfied :
  1. employee i^* must have enough time to leave his home location at the beginning of his working time window, travel to task j^* and perform it before the end of its availability, *i.e.* $lbe_{i^*} + tr_{d_{i^*}j^*} + dt_{j^*} \leq ubt_{j^*}$.
  2. i^* must have then enough time to leave j^* and come back home before the end of their working hours, *i.e.* $max(lbe_{i^*} + tr_{d_{i^*}j^*},\ lbt_{j^*}) + dt_{j^*} + tr_{j^*r_{i^*}} \leq ube_{i^*}$.

  If one of these conditions is not satisfied, then an explanation text can be built with a template text starting with:
  "*In the current solution,* $\langle$*the fact (is observed)*$\rangle$ *because* $\langle$*employee* $i^*\rangle$ *could not perform* $\langle$*task* $j^*\rangle$*, even as the only task of their planning. Indeed, consider a solution such that* $\langle$*employee* $i^*\rangle$ *would perform only* $\langle$*task* $j^*\rangle$."
  Then, depending on the unsatisfied condition, this template text is extended by:
  1. "*By performing* $\langle$*task* $j^*\rangle$ *at the earliest possible time after leaving home,* $\langle$*employee* $i^*\rangle$ *would end it at the earliest at* $\langle lbe_{i^*} + tr_{d_{i^*}j^*} + dt_{j^*}\rangle$*. However,* $\langle$*task* $j^*\rangle$ *must be ended by* $\langle ubt_{j^*}\rangle$."
  2. "*By performing* $\langle$*task* $j^*\rangle$ *at the earliest possible time after leaving home,* $\langle$*employee* $i^*\rangle$ *would end it at the earliest at* $\langle lbe_{i^*} + tr_{d_{i^*}j^*} + dt_{j^*}\rangle$*. However,* $\langle$*task* $j^*\rangle$ *must be ended at the latest at* $\langle ube_{i^*} - tr_{j^*r_{i^*}}\rangle$ *so that* $\langle$*employee* $i^*\rangle$ *has enough time to come back home before* $\langle ube_{i^*}\rangle$."

  Finally, it is ended with "Therefore, such a solution would be infeasible."

In this subsection, we focused on Phase 1 *i.e.* preliminary checks. Such checks are performed because, in the case where they are unsatisfied, they provide proof-like negative explanation texts in polynomial time. In the two following subsections, we deal with Phase 2 *i.e.* complete checks. These checks relate to exploring neighborhoods. The first subsection define the notion of support solutions and describes how we determine such solutions. The second one presents how we then build explanation texts using information extracted from the support solutions.

5.3.2 Complete checks - identifying a support solution

We remind that, in this section, we considered that end-users have a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$ with $\mathcal{S}$ feasible solution of the instance $\mathcal{I}$. From now on, we refer to $\mathcal{S}$ as the *current solution* to distinguish it from other solutions such as neighboring solutions.

As shown in Figure 5.1, the first step of Phase 2 consists in identifying a *support solution*. This is a particular neighboring solution whose properties (including whether it is feasible and whether it is better than the current solution $\mathcal{S}$) will be exploited for building relevant explanations texts. Since the definition of the support solution uses a notion of *nearest-to-feasibility*, we first need to define this notion. In the following paragraphs, we first give the definition of nearest-to-feasibility, then the one of the support solution, and we finally describe how to compute support solutions.

In Subsections 4.2.1 and 4.2.2, we assumed that contrastive questions are based on observations and observations are related to transformations which define neighborhoods. Therefore, the contrastive question q, which is based on the observation o, is related to a transformation tf and a neighborhood $\mathcal{N}(o, \mathcal{S})$[1]. Suppose now that tf is applied on $\mathcal{S}$ to obtain a neighboring solution $\mathcal{S}'$ of $\mathcal{N}(o, \mathcal{S})$. In Section 3.3 (summarized in Table 3.1), we noticed that, regardless of the transformation, the process of building a neighboring solution from a given solution consistently requires to apply a single insertion (either an elementary insertion transformation or an insertion with permutation). However, as we saw it in Section 3.3, insertions are not always feasible. In order to know whether an insertion is feasible and if not by how much, we introduced the *feasibility gap*: if the insertion is feasible, the feasibility gap is null; if the insertion is infeasible, it is positive and measures the difference between the earliest start time at which the inserted task can be started, without violating any time constraints (availability, working hours and sequencing constraints) in the sequence of activities located before the insertion, and the latest start time at which the inserted task can be started, without violating any time constraints in the sequence of activities located after the insertion. Since every transformation process requires to apply a single insertion, we can associate this transformation with a feasibility gap measuring the (in)feasibility of this transformation. In particular, we can associate the transformation from $\mathcal{S}$ to $\mathcal{S}'$ with a feasibility gap. Now consider all the neighboring solutions in $\mathcal{N}(o, \mathcal{S})$. Each one can be associated with a feasibility gap. We call *nearest-to-feasibility* neighboring solutions the solutions with minimum feasibility gap.

Nearest-to-feasibility solution. Consider a feasible solution $\mathcal{S}$ of an instance $\mathcal{I}$ as well as an observation o about $\mathcal{S}$. For any neighboring solution $\mathcal{S}'$ in $\mathcal{N}(o, \mathcal{S})$, building $\mathcal{S}'$ from $\mathcal{S}$ requires to apply an insertion (either an elementary insertion transformation or an insertion with permutation) whose feasibility can be measured thanks to the feasibility gap. Then, we say that a neighboring solution in $\mathcal{N}(o, \mathcal{S})$ is a *nearest-to-feasibility* solution (relatively to $\mathcal{I}$) if it minimizes the feasibility gap.

There are two remarks to be made about this definition. Firstly, there may be several solutions minimizing the feasibility gap, that is why we write in the definition *a* nearest-to-feasibility solution rather than *the* nearest-to-feasibility solution. But in this document, we sometimes abuse language and write *the* nearest-to-feasibility solution. Secondly, note that if $\mathcal{N}(o, \mathcal{S})$ does not contain any feasible solutions, then the feasibility gap of each of its solutions, especially the one of a nearest-to-feasibility solution, is positive. In other words, the nearest-to-feasibility solution tells us about the feasibility of $\mathcal{N}(o, \mathcal{S})$ and its feasibility gap can be seen as a way of quantifying its potential infeasibility.

Now that we have introduced the notion of nearest-to-feasibility solution, we can proceed to the definition of the support solution. This second notion is based on the following assumption. If end-users make an observation o about the current solution $\mathcal{S}$ and want to transform $\mathcal{S}$ in consequence, then they would like to find a neighboring solution $\mathcal{S}^\star$ that is the best feasible one. However, if there is no feasible neighboring solution, then we would be interested in knowing the solution that is the nearest to be feasible.

Support solution. Given a feasible solution $\mathcal{S}$ of an instance $\mathcal{I}$ as well as an observation o about $\mathcal{S}$, we define a *support solution*, noted $\mathcal{S}^\star$, as a solution in $\mathcal{N}(o, \mathcal{S})$, such that:

- if $\mathcal{N}(o, \mathcal{S})$ does not contain any feasible solutions (with respect to $\mathcal{I}$), then $\mathcal{S}^\star$ is a nearest-to-feasibility solution in $\mathcal{N}(o, \mathcal{S})$ (relatively to $\mathcal{I}$);
- else, $\mathcal{S}^\star$ is the best feasible solution in $\mathcal{N}(o, \mathcal{S})$.

Intuitively, the support solution is the "best" or the "most convincing" neighboring solution to present to end-users.

[1] We recall that we made an abuse of the neighborhood notation. Given an observation o about a solution $\mathcal{S}$ and given tf the transformation related to o, while the neighborhood notation was first introduced in Section 3.3 as $\mathcal{N}(tf, \mathcal{S},)$ the neighborhood of $\mathcal{S}$ induced by the transformation tf, we decided to write in Subsection 4.3.1 $\mathcal{N}(o, \mathcal{S})$ the neighborhood related to the observation o (induced by its related transformation tf). In other words, $\mathcal{N}(tf, \mathcal{S}) \equiv \mathcal{N}(o, \mathcal{S})$.

As the notion of support solution is now defined, we can move on to computing such a support solution. We recall that, in Section 3.3, we introduced the notion of *route-equality*. Two solutions of a same instance are said *route-equal* if they have the same employee routes. Then, leveraging this notion of route-equality, we defined three categories of transformations based on the structure of their corresponding neighborhoods. A neighborhood is said to have a *constant-size structure* (resp. *polynomial-size* or *exponential-size* ones), if the number of groups of route-equal neighboring solutions is constant (resp. polynomial or exponential) in n, number of employees in $\mathcal{E}$, and m, number of tasks in $\mathcal{T}$. A transformation is said constant-size (resp. polynomial-size or exponential-size) if its neighborhood has a constant-size (resp. polynomial-size or exponential-size) structure.

In order to identify a support solution, the neighborhood $\mathcal{N}(o, \mathcal{S})$ must be explored. Therefore, the structure of the neighborhood $\mathcal{N}(o, \mathcal{S})$ has a direct impact on the nature of the algorithm used to find a support solution. When $\mathcal{N}(o, \mathcal{S})$ has a constant or polynomial structure, *polynomial-time* algorithms may be used. They explore $\mathcal{N}(o, \mathcal{S})$ by examining each of its subsets of route-equal solutions - given that there is a polynomial number of such subsets. However, when $\mathcal{N}(o, \mathcal{S})$ has an exponential structure, examining an exponential number of subsets of route-equal solutions of $\mathcal{N}(o, \mathcal{S})$ would be computationally intractable. Therefore, we resort to *ILP-based* algorithms which implicitly explore $\mathcal{N}(o, \mathcal{S})$ by solving an ILP model. We discuss in what follows both kinds of algorithms.

Before, we recall the notions of *Backward Earliest start Time* (*BET*) and *Forward Latest start Time* (*FLT*) introduced in Subsection 3.3.1. Suppose that one wants to insert task j in the planning of employee i between activity k_1 and activity k_2, then the BET (resp. FLT), noted st_j^b (resp. st_j^f), is the earliest time (resp. 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 which satisfy time constraints. The, the feasibility gap that we mentioned earlier in this section corresponds to the quantity $max(st_j^b - st_j^f, 0)$ and indicates whether inserting j between k_1 and k_2 in the planning of i is feasible, and if not it measures by how much.

We present now the polynomial-time algorithms used for computing support solutions in relation with constant-size or polynomial-size transformations.

Polynomial-time algorithms for computing support solutions. Given a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$ such that the neighborhood $\mathcal{N}(o, \mathcal{S})$ has a constant-size or polynomial-size structure, we have a *polynomial-time algorithm* for computing a support solution $\mathcal{S}^\star$ used for answering q. In other words, we have a polynomial-size algorithm to compute a support solution for o based on any of the following templates: (Ins,C), (Ins,P,a), (Ins,P,b), (Ins,P,c), (Ex,C), (Ex,P,a), (Ex,P,b), (Ex,P,c), (Ord,C,a), (Ord,C,b), (Ord,P,a), (Ord,P,b) and (Ord,P,c).

The combination of Algorithm 5.1 and Table 5.2 allows to obtain a polynomial-time algorithm for each of the above-mentioned observation templates.

- Algorithm 5.1 is a generic algorithm that shows the common structure of any polynomial algorithm but it also relies on specific parameters and instructions which depend on the observation template of o. The parameters are $\mathcal{E}^\star$ a subset of $\mathcal{E}$, $\mathcal{T}^\star$ a subset of $\mathcal{T}$ and $\mathcal{K}(i, j)$ an application returning a subset of position indices of the activities of $\mathcal{R}_i$, route of a given employee i (and that may also depends of a given activity j).
  Algorithm 5.1 essentially consists of applying on the given solution $\mathcal{S}$ an elementary insertion transformation (at line 9) for various choices of task to insert (in $\mathcal{T}^\star$), employee planning to affect (in $\mathcal{E}^\star$) and position of the task to insert withing this employee planning (in $\mathcal{K}(i, j)$). Before and after this elementary insertion, specific instructions A and B (involved at lines 5 and 8) are either elementary removing transformation or none instruction. Along with with the elementary insertion, they allow to compute insertion, exchange or reordering transformation.
- Table 5.2 specifies, for each of the above-mentioned observation templates, the choice of parameters and specific instructions. In the specification of $\mathcal{K}(i, j)$, a function noted index$(k, \mathcal{R}_i)$ is involved: given an employee i and an activity k, index$(k, \mathcal{R}_i)$ returns the position index of k in the route $\mathcal{R}_i$ of i. Besides, $|\mathcal{R}_i|$ corresponds the number of activities in $\mathcal{R}_i$.

Thus, by combining the generic structure of Algorithm 5.1 with the specific contents of Table 5.2 related to the template of o, we obtain a specific polynomial algorithm for computing the support solution used for answering q.

Finally, all the algorithms based on Algorithm 5.1, along with parameters and instructions from Table 5.2, are polynomial. Indeed, the loops at lines 2, 3 and 6 are repeated respectively $\mathcal{O}(n)$, $\mathcal{O}(m)$ and $\mathcal{O}(m)$ times. And all of the instructions within these loops, including especially the specific instructions at lines 5 and 8 as well as the insertion at line Algorithm 5.1, are computed in constant $\mathcal{O}(1)$ or linear $\mathcal{O}(m)$ times (see Section 3.3).

We can move on to the ILP-based algorithms used for computing support solutions in relation with exponential-size transformations.

Obs. labels	**Subset of empl. $\mathcal{E}^\star$**	**Subset of tasks $\mathcal{T}^\star$**	**Subset of indices $\mathcal{K}(i,j)$ ($i \in \mathcal{E}^\star$, $j \in \mathcal{T}^\star$)**	**Specific instr. A ($i \in \mathcal{E}^\star$, $j \in \mathcal{T}^\star$)**	**Specific instruction B ($i \in \mathcal{E}^\star$, $j \in \mathcal{T}^\star$, $k \in \mathcal{K}(i,j)$)**
(Ins,C)	$\{i^*\}$	$\{j^*\}$	$\{\text{index}(k^*, \mathcal{R}_i)\}$	-	-
(Ins,P,a)	$\{i^*\}$	$\{j^*\}$	$\{1, \ldots, \lvert\mathcal{R}_i\rvert - 1\}$	-	-
(Ins,P,b)	$\{i^*\}$	$\{j \in \mathcal{T} \mid j$ not performed in $\mathcal{S}\}$	$\{1, \ldots, \lvert\mathcal{R}_i\rvert - 1\}$	-	-
(Ins,P,c)	$\mathcal{E}$	$\{j^*\}$	$\{1, \ldots, \lvert\mathcal{R}_i\rvert - 1\}$	-	-
(Ex,C)	$\{i^*\}$	$\{j^*\}$	$\{\text{index}(k^*, \mathcal{R}_i) - 1\}$	-	Remove task at index $k+1$ from i's planning
(Ex,P,a)	$\{i^*\}$	$\{j^*\}$	$\{1, \ldots, \lvert\mathcal{R}_i\rvert - 2\}$	-	Remove task at index $k+1$ from i's planning
(Ex,P,b)	$\{i^*\}$	$\{j \in \mathcal{T} \mid j$ not performed in $\mathcal{S}\}$	$\{1, \ldots, \lvert\mathcal{R}_i\rvert - 2\}$	-	Remove task at index $k+1$ from i's planning
(Ex,P,c)	$\mathcal{E}$	$\{j^*\}$	$\{1, \ldots, \lvert\mathcal{R}_i\rvert - 2\}$	-	Remove task at index $k+1$ from i's planning
(Ord,C,a)	$\{i^*\}$	$\{j^*\}$	$\{\text{index}(k^*, \mathcal{R}_i) - 1\}$	Remove j from i's planning	-
(Ord,C,b)	$\{i^*\}$	$\{j^*\}$	$\{\text{index}(k^*, \mathcal{R}_i) - 1\}$	Remove j from i's planning	-
(Ord,P,a)	$\{i^*\}$	$\{j^*\}$	$\{\text{index}(k^*, \mathcal{R}_i) - 1, \ldots, \lvert\mathcal{R}_i\rvert - 2\}$	Remove j from i's planning	-
(Ord,P,b)	$\{i^*\}$	$\{j^*\}$	$\{1, \ldots, \text{index}(k^*, \mathcal{R}_i) - 1\}$	Remove j from i's planning	-
(Ord,P,c)	$\{i^*\}$	$\{j^*\}$	$\{1, \ldots, \text{index}(j, \mathcal{R}_i) - 2, \text{index}(j, \mathcal{R}_i), \ldots, \lvert\mathcal{R}_i\rvert - 2\}$	Remove j from i's planning	-

Table 5.2: Parameters and instructions involved in Algorithm 5.1 for various possible observation templates. In the table header, "obs.", "empl." and "instr." stand respectively for "observation", "employees" and "instruction". In the table content, $\mathcal{S}$ is the feasible solution of the instance $\mathcal{I}$ both given as inputs in Algorithm 5.1. The symbol "-" for first and second specific instructions means that there is no specific instruction to apply.

Algorithm 5.1: Polynomial-time algorithm for computing a support solution[2]

Inputs:
$\mathcal{I}$ an instance
$\mathcal{S}$ a feasible solution of $\mathcal{I}$
o an observation about $\mathcal{S}$ (such that $\mathcal{N}(o, \mathcal{S})$ has a constant or polynomial structure)

Parameters (which depend on the observation template of o, *cf.* Table 5.2):
$\mathcal{E}^\star$ a subset of the set of employees $\mathcal{E}$
$\mathcal{T}^\star$ a subset of the set of tasks $\mathcal{T}$
$\mathcal{K}(i,j) \subset \mathbb{N}$ a subset of planning indices, function of $i \in \mathcal{E}^\star$ and $j \in \mathcal{T}^\star$

1 $\mathcal{S}^\star \leftarrow \varnothing$
2 **for** $i \in \mathcal{E}^\star$ **do**
3 **for** $j \in \mathcal{T}^\star$ **do**
4 $\mathcal{S}' \leftarrow \mathcal{S}$
5 Apply to $\mathcal{S}'$ specific instruction A (which depends on the template of o, *cf.* Table 5.2)
6 **for** $k \in \mathcal{K}(i,j)$ **do**
7 $\mathcal{S}'' \leftarrow \mathcal{S}'$
8 Apply to $\mathcal{S}''$ specific instruction B (which depends on the template of o, *cf.* Table 5.2)
9 Insert task j in the planning of employee i after its k^{th} activity, in $\mathcal{S}''$, so as to minimize the feasibility gap of this insertion $max(st_j^b - st_j^f, 0)$
10 **if** $\mathcal{S}^\star = \varnothing$ **or**
 (both $\mathcal{S}''$ and $\mathcal{S}^\star$ are infeasible) and ($\mathcal{S}''$ has a smaller infeasibility gap than $\mathcal{S}^\star$) **or**
 $\mathcal{S}''$ feasible and not $\mathcal{S}^\star$ **or**
 (both $\mathcal{S}''$ and $\mathcal{S}^\star$ are feasible) and ($\mathcal{S}''$ is better than $\mathcal{S}^\star$) **then**
11 $\mathcal{S}^\star \leftarrow \mathcal{S}''$
12 Along with $\mathcal{S}^\star$, save information related to the task insertion that has been applied at line 9: task j is inserted in the planning of employee i after its k^{th} activity, with BET st_j^b and FLT st_j^f

Output:
$\mathcal{S}^\star$ a support solution (as well as the information mentioned at line 12)

The polynomial-time algorithm presented here meets its goal, namely computing a support solution for any constant-size or polynomial-size transformation. However, there exists a more efficient algorithm meeting the same goal that we do not present for simplicity but that we can outline. The efficiency improvement of this other algorithm compared to the one presented here relies on two remarks: *i)* copying solutions as it is done at lines 7 or 11 is costly (linear cost) *ii)* we do not need to practically remove, insert or exchange a task from an employee planning in order to know the feasibility of a single removing, insertion or exchanging transformation and the impact on the bi-objective function values, we can anticipate these (by using the BTS and FTS see Subsection 5.3.1, and by simply anticipating the variations of objectives). Thus, instead of copying $\mathcal{S}'$ into $\mathcal{S}''$ and $\mathcal{S}''$ into $\mathcal{S}^\star$ at line, we can anticipate the feasibility and the variation of the bi-objective function values of the solution resulting from the instructions at lines 8 and 9 without computing it. We can also compare this "non-built" solution (feasibility and bi-objective function values) with previous similar solutions, including the best "non-built" support solution found so far. If it is better, the solution becomes the best "non-built" support solution found so far, and we save it, not the solution itself (which would require a solution copy) but the operations to apply to $\mathcal{S}$ in order to build it. So that, finally, after examining the neighborhood, the support solution $\mathcal{S}^\star$ is actually built, once, by applying the saved best operations to apply to $\mathcal{S}$. Given a constant-size or polynomial-size transformation, this efficient algorithm has a computational complexity corresponding to the maximum between the one of the transformation feasibility check and the one for building a neighboring solution.

ILP-based algorithms developed for computing support solution use various ILP models which depend on the template of the observation involved in the contrastive question. All these models are based on a generic ILP model that we call *contrastive transformation model* and introduce below. We recall that we introduced the main model $mm(\mathcal{I})$ (see Model 1 in Subsection 3.2.2) for modeling our WSRP use case and the foil model $fm(o, \mathcal{S}, \mathcal{I})$ (see Model 2 in Subsection 4.3.2) for modeling the theoretical exploration in the neighborhood $\mathcal{N}(o, \mathcal{S})$ of a solution that would be feasible and better than $\mathcal{S}$.

Contrastive transformation model. Given a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$ such that $\mathcal{N}(o, \mathcal{S})$ has an exponential-size structure, *i.e.* o is based on one of the observation templates (Ins,E), (Ex,E) or (Ord,E), we call *contrastive transformation model*, noted $ctm(o, \mathcal{S}, \mathcal{I})$, the bi-objective ILP model used for finding a support solution $\mathcal{S}^\star$ with the aim to answer q. 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^*.

$ctm(o, \mathcal{S}, \mathcal{I})$ is obtained by combining Model 3 with observation-dependent parameters and constraints from Table 5.3.

- Model 3 is a generic bi-objective ILP model which 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 $\mathcal{S}^\star$. Model 3 is generic because it uses a set $\mathcal{T}^\star$ and constraint (M3.13), whose explicit mathematical contents depend on the observation template of o and need to be specified to get a fully defined model.

- The purpose of Table 5.3 is actually to specify, for each of the above-mentioned observation templates, the subset $\mathcal{T}^\star$ and the constraints (M3.13). $\mathcal{T}^\star$ is a subset of the set of tasks $\mathcal{T}$. It contains 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^*). (M3.13) corresponds to neighboring constraints which ensure that, if the planning of employee i in $\mathcal{S}$ is changed for the planning obtained by solving Model 3, then the obtained solution is a neighboring solution of $\mathcal{N}(o, \mathcal{S})$. Neighboring constraints are noted $\varphi(\mathcal{X}) \in \mathcal{N}(o, \mathcal{S})$ in Model 3 in reference to the neighboring constraints in the foil model $fm(q)$ which play a similar role.

$ctm(o, \mathcal{S}, \mathcal{I})$ is based on main model $mm(\mathcal{I})$ and foil model $fm(o, \mathcal{S}, \mathcal{I})$. However, $ctm(o, \mathcal{S}, \mathcal{I})$ differs from $mm(\mathcal{I})$ on various aspects which we detail below.

- **General differences.** An overall difference between the two models is that $ctm(o, \mathcal{S}, \mathcal{I})$ focuses on optimizing only one employee planning, namely the planning of i^*, whereas $mm(\mathcal{I})$ deals with a solution *i.e.* the plannings of all the employees of $\mathcal{E}$.
  The main consequence of this focus on the planning of i^* is that not all the decision variables of $mm(\mathcal{I})$ are involved in $ctm(o, \mathcal{S}, \mathcal{I})$: only the path decision variables U_{ijk} with $i = i^*$ and the start time ones T_j with $j \in \mathcal{T}^\star$ are indeed necessary for $ctm(o, \mathcal{S}, \mathcal{I})$. In other words, optimizing $ctm(o, \mathcal{S}, \mathcal{I})$ can be seen as fixing in $mm(\mathcal{I})$ all the variables U_{ijk} with $i \neq i^*$ as well as all the variables T_j with $j \in \mathcal{T} \setminus \mathcal{T}^\star$ to the values they take in $\mathcal{S}$ and focusing the optimization over the set of the remaining variables - plus other decision variables which will be described below.
  Other consequences of this focus on the planning of i^* are that: there are no sums over $\mathcal{E}$ or constraints repeated over $\mathcal{E}$ in $ctm(o, \mathcal{S}, \mathcal{I})$; sums indexed over $\mathcal{T}$ in $mm(\mathcal{I})$ are indexed over $\mathcal{T}^\star$ in $ctm(o, \mathcal{S}, \mathcal{I})$; constraints repeated over $\mathcal{T}$ in $mm(\mathcal{I})$ are repeated over $\mathcal{T}^\star$ in $ctm(o, \mathcal{S}, \mathcal{I})$; *etc.*
  Finally, this restriction leads to a drastic decrease in the size of $ctm(o, \mathcal{S}, \mathcal{I})$ compared with $mm(\mathcal{I})$, which helps to solve $ctm(o, \mathcal{S}, \mathcal{I})$ faster.

- **Decision variables.** In addition to the path decision variables, U_{i^*jk} with $(j, k) \in \mathcal{A}^2_{i^*}$ such that $j \neq k$, and the start time ones, T_j for $j \in \mathcal{T}^\star$, inherited from $mm(\mathcal{I})$, two new integer variables $T^{lb}_{j^*}$ and $T^{ub}_{j^*}$ are brought into play in $ctm(o, \mathcal{S}, \mathcal{I})$. Like T_{j^*}, they are related to the time at which the task j^* starts to be performed (by i^*): $T^{lb}_{j^*}$ (resp. $T^{ub}_{j^*}$) 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 $T^{lb}_{j^*}$ and $T^{ub}_{j^*}$, *BET* and *FLT decision variables*, as they are the decision-variable equivalents of the BET st^b_j and FLT st^f_j.
  Thanks to the way we involve $T^{lb}_{j^*}$ and $T^{ub}_{j^*}$ in $ctm(o, \mathcal{S}, \mathcal{I})$, these two variables allow us to guarantee the feasibility of this model: either $T^{lb}_{j^*} > T^{ub}_{j^*}$ and it means that inserting j^* in the planning of i^* is infeasible, even if we permute the order of the tasks; or $T^{lb}_{j^*} = T^{ub}_{j^*}$ 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.

- **Multi-objective function.** Similarly to $mm(\mathcal{I})$, in $ctm(o, \mathcal{S}, \mathcal{I})$, a bi-objective function (M3.1) is minimized according to a lexicographic order. However, the objective are not the same.
  1. The first objective aims at tightening the gap between the two variables $T_{j^*}^{lb}$ and $T_{j^*}^{ub}$ by minimizing the difference $T_{j^*}^{lb} - T_{j^*}^{ub}$, as having $T_{j^*}^{lb} = T_{j^*}^{ub}$ means that the insertion of j^* in the planning of i^* is feasible. Note that the constraint (M3.12.a) prevents the difference $T_{j^*}^{lb} - T_{j^*}^{ub}$ from being negative.
  2. The second objective is related to the second objective of $mm(\mathcal{I})$ as it minimizes the total traveling time of the planning of i^*.

- **Constraints.** Most constraints of the $mm(\mathcal{I})$ still apply in the $ctm(o, \mathcal{S}, \mathcal{I})$ but must be adapted.
  - Flow constraints (M1.2) to (M1.4) correspond to constraints (M3.2) to (M3.4); however (M3.2) to (M3.4) zoom on employee i^* and subset $\mathcal{T}^\star$ while (M1.2) to (M1.4) involve the whole sets $\mathcal{E}$ and $\mathcal{T}$.
  - No equivalent of the skill constraint (M1.5) appears in the $ctm(o, \mathcal{S}, \mathcal{I})$ as i^* is supposed to have a higher skill level than the ones of all the tasks of $\mathcal{T}^\star$ (especially j^*).
  - Occurrence constraint (M1.6) coincides with constraint (M3.6) but (M3.6) zoom on employee i^* and subset $\mathcal{T}^\star$.
  - Availability, working hours and sequencing constraints spanning labels from (M1.7) to (M1.11) in $mm(\mathcal{I})$ are also involved, with some changes, in $ctm(o, \mathcal{S}, \mathcal{I})$ as labels spanning from (M3.7.a) to (M3.11.b). Each original constraint of $mm(\mathcal{I})$ (*e.g.* constraint (M1.7)) is split into two or three constraints in $ctm(o, \mathcal{S}, \mathcal{I})$ (*e.g.* constraints (M3.7.a) and (M3.7.b)) in order to separate the case of j^*, which involves $T_{j^*}^{lb}$ and $T_{j^*}^{ub}$, from the one of any other task j in $\mathcal{T}^\star$, which involves T_j.

  In addition, three new constraints are introduced in $ctm(o, \mathcal{S}, \mathcal{I})$.

  - As mentioned earlier, constraint (M3.12.a) prevents the difference $T_{j^*}^{lb} - T_{j^*}^{ub}$ from being negative.
  - Constraint (M3.12.b) is simply used for controlling T_{j^*} value which is no longer involved in any constraint.
  - Constraint (M3.13) corresponds to the neighboring constraints introduced above and detailed in Table 5.3.

ILP-based algorithms for computing support solutions. Given a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$ such that $\mathcal{N}(o, \mathcal{S})$ has an exponential-size structure, *i.e.* o is based on one of the observation templates (Ins,E), (Ex,E) or (Ord,E), we propose an *ILP-based algorithm* for computing a support solution pair $\mathcal{S}^\star$ used for answering the question q. Regardless of the observation template of o, this algorithm is described in Algorithm 5.2. It consists in solving (at line 2) the contrastive transformation model $ctm(o, \mathcal{S}, \mathcal{I})$ which we described above. The results of this solving are then used to build the support solution (at lines 4 and 5).

In this subsection, we described the first step of Phase 2: given a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$, computing a support solution $\mathcal{S}^\star$ - by resorting to either polynomial-time or ILP-based algorithms depending on the structure of $\mathcal{N}(o, \mathcal{S})$. In the following subsection, we detail the remaining steps of Phase 2: using $\mathcal{S}^\star$ in order to build an explanation text $ux(q)$ answering q.

$$lex\ min\ \Big(\ T^{lb}_{j^*} - T^{ub}_{j^*},\ \sum_{j\in\mathcal{A}_{i^*},\ j\neq r_{i^*}}\ \sum_{k\in\mathcal{A}_{i^*},\ k\neq d_{i^*},j} U_{i^*jk}\ tr_{jk}\ \Big) \tag{M3.1}$$

$s.t.$

$$\sum_{k\in\mathcal{A}_{i^*},\ k\neq d_{i^*}} U_{i^*d_{i^*}k} = 1 \tag{M3.2}$$

$$\sum_{j\in\mathcal{A}_{i^*},\ j\neq r_{i^*}} U_{i^*jr_{i^*}} = 1 \tag{M3.3}$$

$$\sum_{j\in\mathcal{A}_{i^*},\ j\neq k,r_{i^*}} U_{i^*jk} = \sum_{j'\in\mathcal{A}_{i^*},\ j'\neq d_{i^*},k} U_{i^*kj'} \qquad \forall\ k\in\mathcal{T}^\star \tag{M3.4}$$

$$\sum_{k\in\mathcal{A}_{i^*},\ k\neq d_{i^*},j} U_{i^*jk} \leq 1 \qquad \forall\ j\in\mathcal{T}^\star \tag{M3.6}$$

$$lbt_j \leq T_j \qquad \forall\ j\in\mathcal{T}^\star\setminus\{j^*\} \tag{M3.7.a}$$

$$lbt_{j^*} \leq T^{lb}_{j^*} \tag{M3.7.b}$$

$$T_j \leq ubt_j - dt_j \qquad \forall\ j\in\mathcal{T}^\star\setminus\{j^*\} \tag{M3.8.a}$$

$$T^{ub}_{j^*} \leq ubt_{j^*} - dt_{j^*} \tag{M3.8.b}$$

$$lbe_{i^*} + tr_{d_{i^*}k} \leq T_k \qquad \forall\ k\in\mathcal{T}^\star\setminus\{j^*\} \tag{M3.9.a}$$

$$lbe_{i^*} + tr_{d_{i^*}j^*} \leq T^{lb}_{j^*} \tag{M3.9.b}$$

$$T_j + dt_j + U_{i^*jk}\ tr_{jk} \leq T_k + \Big(1 - U_{i^*jk}\Big)ubt_j \qquad \forall\ j\neq k\in\mathcal{T}^\star\setminus\{j^*\} \tag{M3.10.a}$$

$$T^{ub}_{j^*} + dt_{j^*} + U_{i^*j^*k}\ tr_{j^*k} \leq T_k + \Big(1 - U_{i^*j^*k}\Big)ubt_{j^*} \qquad \forall\ k\in\mathcal{T}^\star\setminus\{j^*\} \tag{M3.10.b}$$

$$T_j + dt_j + U_{i^*jj^*}\ tr_{jj^*} \leq T^{lb}_{j^*} + \Big(1 - U_{i^*jj^*}\Big)ubt_j \qquad \forall\ j\in\mathcal{T}^\star\setminus\{j^*\} \tag{M3.10.c}$$

$$T_j + dt_j \leq U_{i^*jr_{i^*}}(ube_{i^*} - tr_{jr_{i^*}}) + \Big(1 - U_{i^*jr_{i^*}}\Big)ubt_j \qquad \forall\ j\in\mathcal{T}^\star\setminus\{j^*\} \tag{M3.11.a}$$

$$T^{ub}_{j^*} + dt_{j^*} \leq U_{i^*j^*r_{i^*}}(ube_{i^*} - tr_{j^*r_{i^*}}) + \Big(1 - U_{i^*j^*r_{i^*}}\Big)ubt_{j^*} \tag{M3.11.b}$$

$$T^{lb}_{j^*} - T^{ub}_{j^*} \geq 0 \tag{M3.12.a}$$

$$T^{ub}_{j^*} \leq T_{j^*} \leq T^{lb}_{j^*} \tag{M3.12.b}$$

$$\varphi(\mathcal{X}) \in \mathcal{N}(o,\mathcal{S}) \tag{M3.13}$$

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

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

$$T^{lb}_{j^*},\ T^{ub}_{j^*}\in\mathbb{N}$$

Model 3: Contrastive transformation model $ctm(o,\mathcal{S},\mathcal{I})$, *i.e.* bi-objective ILP model used to identify the support planning $(\mathcal{R}^\star_{i^*},\mathcal{C}^\star_{i^*})$ of employee i^* (mentioned in o) so as to build a support solution $\mathcal{S}^\star$ for answering the contrastive question $q_c(o,\mathcal{S},\mathcal{I})$.

Labels	Subset of tasks $\mathcal{T}^\star$	Neighborhood constraints
(Ins,E)	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} < \lvert\mathcal{R}_{i^*}\rvert - 2 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = 1 \qquad \forall\, j \in \mathcal{T}^\star \end{cases}$
(Ex,E)	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} < \lvert\mathcal{R}_{i^*}\rvert - 3 \\ \sum\limits_{j \in \mathcal{T}^\star \setminus \{j^*\}} \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = \lvert\mathcal{R}_{i^*}\rvert - 3 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j^*\}} U_{i^*j^*k} = 1 \end{cases}$
(Ord,E)	$\mathcal{T} \cap \mathcal{R}_{i^*}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} < \lvert\mathcal{R}_{i^*}\rvert - 4 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = 1 \qquad \forall\, j \in \mathcal{T}^\star \end{cases}$

Table 5.3: Parameters and constraints involved in Model 3 for various possible observation templates. In the table content, $\mathcal{S}$ is the feasible solution of the instance $\mathcal{I}$ both given as inputs in Algorithm 5.2.

Algorithm 5.2: ILP-based algorithm for computing a support solution

Inputs:
$\mathcal{I}$ an instance
$\mathcal{S}$ a feasible solution of $\mathcal{I}$
o an observation about $\mathcal{S}$ (such that $\mathcal{N}(o, \mathcal{S})$ has an exponential structure)

1 $\mathcal{S}^\star \leftarrow \mathcal{S}$
2 Solve the contrastive transformation model $ctm(o, \mathcal{S}, \mathcal{I})$ (*cf.* Model 3)
3 Obtain the values of $T^{lb}_{j^*}$ and $T^{ub}_{j^*}$ and store them respectively as BET st^b_j and FLT st^f_j
4 Build the support planning $(\mathcal{R}^\star_{i^*}, \mathcal{C}^\star_{i^*})$ of employee i^* given the values of spatial and temporal decision variables resulting from solving $ctm(o, \mathcal{S}, \mathcal{I})$ and save related information:
task j^* is inserted in the planning of employee i^* after its k^{th} activity, with BET st^b_j and FLT st^f_j
5 Create support solution $\mathcal{S}^\star$ by copying $\mathcal{S}$ and replacing the planning of i^* with the support planning $(\mathcal{R}^\star_{i^*}, \mathcal{C}^\star_{i^*})$

Output:
$\mathcal{S}^\star$ a support solution (as well as information mentioned at line 4)

5.3.3 Complete checks - building an explanation text using the support solution

As shown in Figure 5.1, once the support solution $\mathcal{S}^\star$ has been identified for answering the contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$, there are three different cases.

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

In the following paragraphs, we first describe general information related to the support solution that we use in explanation texts and then present how we write explanation texts in each of the three cases.

Information related to support solution. Both Algorithms 5.1 and 5.2 build a support solution $\mathcal{S}^\star$ by considering an elementary insertion transformation. The output data of these algorithms not only contain the support solution but also details about this insertion. They describe which specific task j^* is inserted in the planning of which specific employee i^* and after which specific activity k in order to build $\mathcal{S}^\star$ from $\mathcal{S}$. Finally, the output data also provide BET st_j^b and FLT st_j^f related to this insertion.

Thanks to these details, we can deduce which of the three above-presented cases $\mathcal{S}^\star$ is in. If the feasibility gap $max(st_j^b - st_j^f, 0)$ is positive, then $\mathcal{S}^\star$ is infeasible, which corresponds to case 1. If it is null, then $\mathcal{S}^\star$ is feasible, and we can compare the objectives of $\mathcal{S}$ and $\mathcal{S}^\star$ in order to deduce whether $\mathcal{S}^\star$ is in case 2 or 3. Note that, as skill constraints are checked as part of preliminary checks (see Subsection 5.3.1), if $\mathcal{S}^\star$ is infeasible, it is necessarily related to time constraints (including availability, working hours and sequencing constraints). That is why we need to check just time-related considerations and not skill-related ones.

Identifying in which above-presented case is $\mathcal{S}^\star$ is not the only information that we can obtain from the results of Algorithms 5.1 and 5.2 and that we can use to build explanation texts.

- We can prepare a typical expression "⟨applying the specific transformation to get the support *solution*⟩" describing what specific transformation must be applied to $\mathcal{S}$ in order to obtain $\mathcal{S}^\star$ (specifying what are the specific employees, tasks or activities concerned by the transformation).
- In the case where the elementary insertion is infeasible, we can deduce whether it is specifically due to a conflict between the sequence of activities before the insertion of task j^* by checking if $st_j^b + dt_{j^*} > ubt_{j^*}$.
- BET st_j^b (resp. FLT st_j^f) can be used within the explanation text as earliest (resp. latest) time at which task j^* should be started so that the sequence of activities before (resp. after) j^* can be performed while respecting time constraints.
- In the case where the elementary insertion is infeasible, we can also deduce what are the backward and forward critical activities (see Section 3.3) related to this insertion and use them in the explanation text.

Now that we described all the information we can obtain from the computation of $\mathcal{S}^\star$ and use within the explanation texts, we can present how we build explanation texts for each of the three above-mentioned cases.

Case of infeasible support solution. Consider the case where $\mathcal{S}^\star$ is not feasible. We must then provide a negative explanation text describing the infeasibility of the solutions of $\mathcal{N}(o, \mathcal{S})$. As shown in Figure 5.1, there are two different sub-cases: either all the solutions in $\mathcal{N}(o, \mathcal{S})$ are route-equal, then we write a text corresponding to a proof-like negative explanation; or, solutions in $\mathcal{N}(o, \mathcal{S})$ are not route-equal, then we write a text corresponding to a argument-like negative explanation. However, in both cases, we use $\mathcal{S}^\star$ and its related information to write these texts.

Figure 5.2 details how we build (proof-like and argument-like) negative explanation texts about time-infeasibility. We concatenate four parts of template texts whose fields are filled with typical expressions from Table 5.1 or data related to the computation of $\mathcal{S}^\star$.

1. The first part of text expresses in quite general terms why the explanation is negative: "due to time constraints".
2. The second part of text refers to a transformation of the current solution and introduces a "new solution" which corresponds to $\mathcal{S}^\star$. This text depends on the structure of $\mathcal{N}(o, \mathcal{S})$, more specifically it depends on whether $\mathcal{N}(o, \mathcal{S})$ is constant-size or not. In this second case, the use of "for example" brings out the fact that the text corresponds to an argument-like negative explanation.
3. The third part of text depends on whether the task insertion applied to build $\mathcal{S}^\star$ is backward infeasible or forward infeasible. In both cases, the text refers to information related to $\mathcal{S}^\star$ (including backward or forward critical activity, earliest start or end time) as well as information related to o and $\mathcal{I}$.
4. The fourth and last part of text is a conclusion.

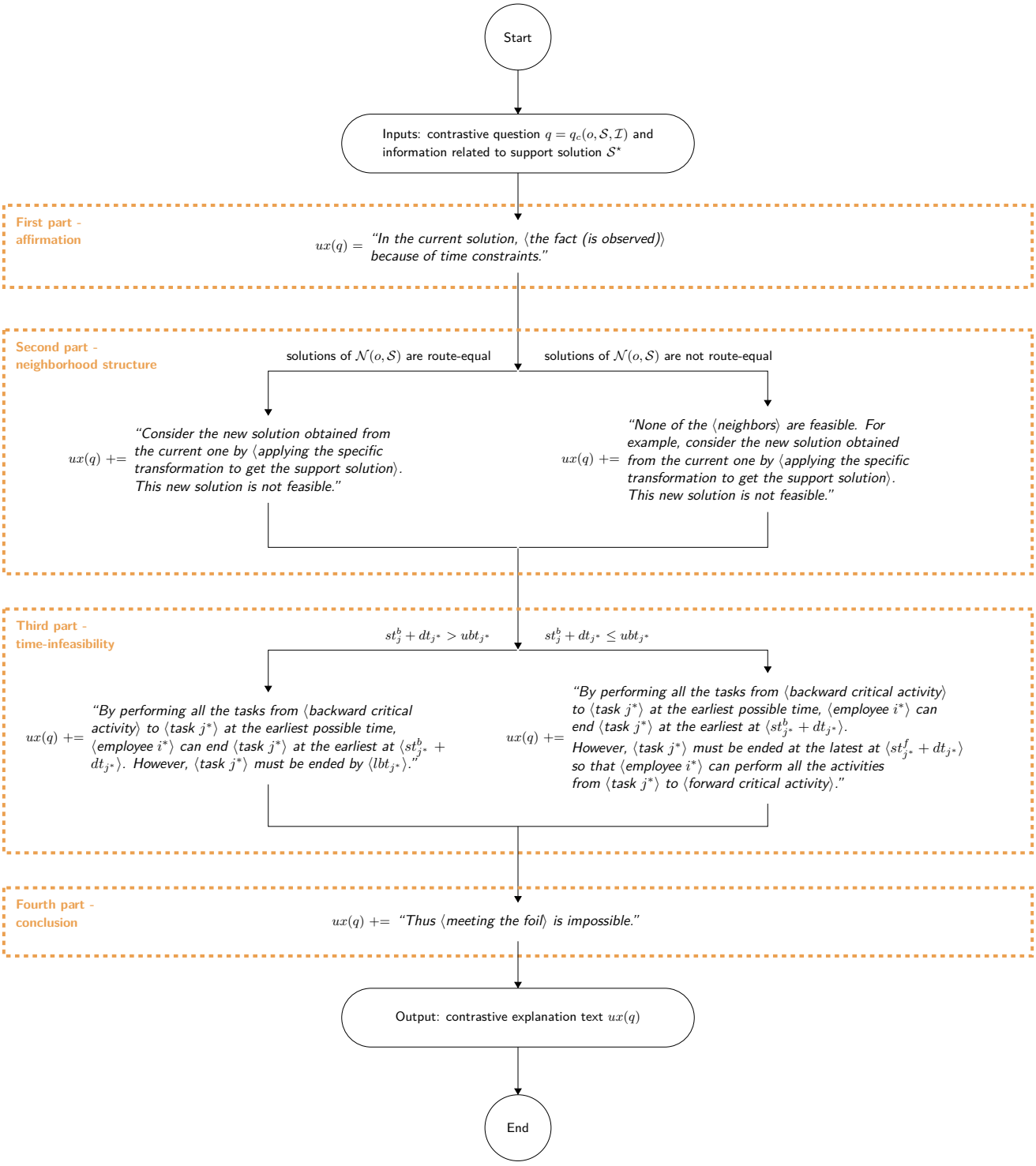


Figure 5.2: Building contrastive negative explanation text about time-infeasibility given a support solution.

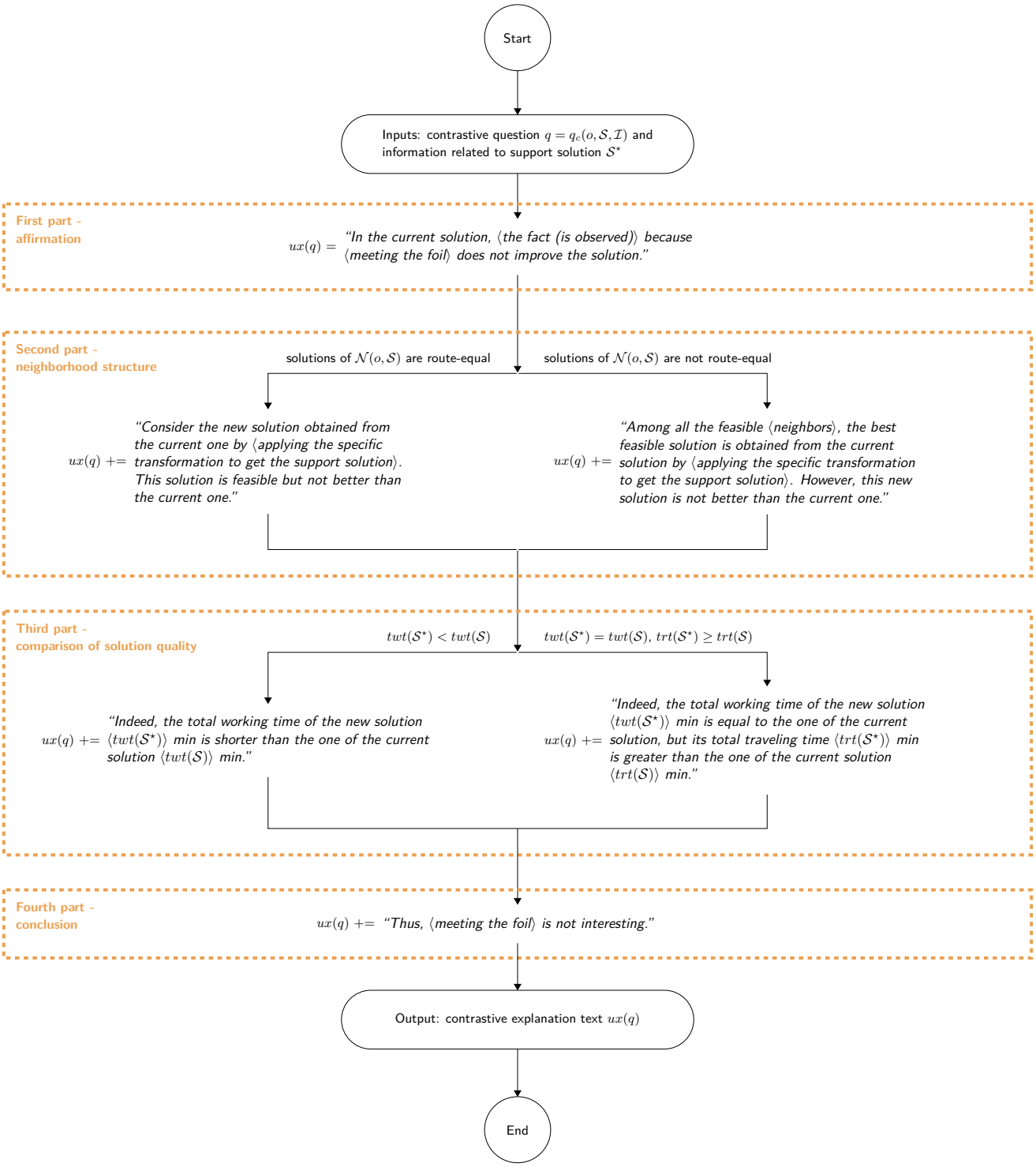


Figure 5.3: Building a contrastive negative explanation text about non-improvement given a support solution.

Case of feasible but non-improving support solution. Consider the case where $\mathcal{S}^\star$ is feasible but not better than the current solution $\mathcal{S}$. We must then provide a negative explanation text describing why the feasible solutions of $\mathcal{N}(o, \mathcal{S})$ are not better than $\mathcal{S}$. Again, the negative explanation is either proof-like or argument-like, depending on the structure of $\mathcal{N}(o, \mathcal{S})$, but in both cases, we use the knowledge of $\mathcal{S}^\star$ to write these texts.

Figure 5.3 details how we build (proof-like and argument-like) negative explanation texts about non-improvement. Similarly to the case of time-infeasibility, we concatenate four parts of template texts using typical expressions and data related to the computation of $\mathcal{S}^\star$.

Case of feasible and improving support solution. Consider the case where $\mathcal{S}^\star$ is feasible and better than $\mathcal{S}$. We must then build a positive explanation text, which consists essentially in presenting $\mathcal{S}^\star$. It can be expressed as follows: "*Because the current solution is actually not optimal, ⟨the fact (is observed)⟩. However, indeed, ⟨meeting the foil⟩ is possible and provides better solutions such as the solution obtained by ⟨applying the specific transformation to get the support solution⟩.*"

In the three last subsections, we presented how we generate contrastive explanation texts. In the next and last subsection, we study the performances of this generation in terms of computation time.

5.3.4 Numerical experiments

In the perspective of integrating the generation of explanation texts within a system where end-users ask questions and obtain explanation texts in return, we would like that the algorithms we designed for generating contrastive explanation texts run within a time-frame compatible with near-real-time use of explanations by end-users. This subsection presents the numerical experiments that we conduct to assess the computation times required to generate contrastive explanation texts on large-scale WSRP instances and solutions. First, we describe the sets of instances and solutions that we use 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 numerical experiments over 96 pairs of instances and solutions[3]. 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 is computed thanks to a heuristic algorithm solving WSRP instances. Thus, solutions may not be optimal and explanations may be positive.

Experimental setting. In Chapter 4, we present a list of contrastive question templates in Table 4.2 based on the list of observation templates in Table 4.1. Let $\mathcal{S}$ be a feasible solution of an instance $\mathcal{I}$. From a given question template, many questions can be defined by filling its fields with values from $\mathcal{I}$ (*e.g.* the name of an employee). The number of questions depends on the template itself, the contents of $\mathcal{I}$ and $\mathcal{S}$. For instance, from (Ord,E) template, n questions can be defined, with n the number of employees in $\mathcal{I}$, by filling in the field "⟨employee i^*⟩" with the name of each of the n employees. From (Ins,E) template, many more questions can be defined since not only an employee i^* must be chosen to fill in the field "⟨employee i^*⟩" but also a task not performed by i^* to fill in the field "⟨task j^*⟩".

In our numerical experiments, for each instance-solution pair and for each contrastive question template of Table 4.2, we define a random sample of questions based on this template. We randomly draw a sample of 40 different questions for each template[4]. Moreover, we limit the time allowed for computing the explanation text answering each of these questions to 15s, 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 15s. The computations lasting for less than 15s are counted as *completed* computations and their times are saved. The others are counted as *interrupted* computations.

All algorithms are implemented in Python 3.9 and all ILP models are solved with Gurobi Optimizer 9.5. All experiments are run in MacOS 10.15.7 on a 2.3 GHz Quad-Core Intel Core i7 processor with 16GB RAM.

Results. We separate the results about computation times in two tables: Table 5.4 for the computations related to polynomial-time algorithms (based on Algorithm 5.1) and Table 5.5 for the ones related to ILP-based algorithms (based on Algorithm 5.2). In both tables, each line provides statistics about the computation times associated with one contrastive question 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 15s 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.

[3]Instances and solutions are available in an online repository: https://github.com/MathieuLerouge/WSRP-data

[4]In the case where fewer than 40 different questions can be defined from a question template (*e.g.* with (Ord,E) template if $n < 40$), the sample is made of all the possible questions that can be defined.

Question template	Computation times			
	Median in s	3rd quartile in s	Maximum in s	Average in s
(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 5.4: Statistics about the contrastive explanation computation times using polynomial-time algorithms.

Question template	Completed computation rate	Computation times		
		1st quartile	Median	3rd quartile
	%	in s	in s	in s
(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

Table 5.5: Statistics about the contrastive explanation computation times using ILP-based algorithms.

In Table 5.4, for each template, we compute the median, third quartile, maximum and average of its corresponding computation times. For instance, the generated explanation texts answering contrastive questions based on (Ins,C) template have been computed on average in 0,025s; among these texts, at least 50% have been computed in less than 0,022s, 75% in less than 0,035s, and the longest to be computed has required 0,162s.

In Table 5.5, for each question template, we first compute its completed computations rate *i.e.* the proportion of completed computations (*i.e.* computed in less than 15s) 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 instance, among all the computations of contrastive explanation texts answering questions based on (Ins,E) template, 69,84% are completed in less than 15s, at least 25% have lasted for less than 0,126s and at least 50% for less than 1,176s. Note that, for (Ins,E) template, the third quartile value is given as "$>$ 15s" since less than 75% of the computations related to this template have been completed in less than 15s.

Analysis and insights. Computational experiments show that contrastive explanation texts produced using polynomial-time algorithms are computed in a very short time: most of them are computed in less than 0,3s. Such performances were expected since these algorithms are polynomial in the number of employees and/or tasks, as mentioned in Subsection 5.3.2. The experiments also show that, within question templates related to a given transformation family (insertion, exchange or reordering), explanation texts related to polynomial-size transformations generally require more time to be computed than the ones related to constant-size transformations (*e.g.* on average, (Ins,P,c) texts require 0,067s to be computed while (Ins,C) texts require 0,025s). Such a trend could be anticipated for "Ins" and "Ex" families since the feasibility check for constant-size transformations is performed in constant time while it is performed in polynomial time for polynomial-size transformations (see Table 3.1). However, note that the results (Ins,C) and (Ins,P,a) are about the same. This can be explained: even though the feasibility check related to (Ins,C) is computed in constant time, the process for building a neighboring solution is linear, so that the overall explanation generation process is linear, like for (Ins,P,a). A similar comment can be made for the other families. Finally, (Ins,P,b) and (Ex,P,b) templates present the longest computation times (respectively 3,395s and 3,855s) which is related to their corresponding feasibility check complexity in $\mathcal{O}(m^2)$.

Regarding contrastive explanation texts produced thanks to ILP-based algorithms, whatever the question template, experiments show that more than 50% of the explanation texts computations last for a short time: less than 1,2s. However, a significant proportion of theses computations have been interrupted as they have reached the time limit of 15s. Depending on the question template, this proportion varies from 8,97 to 30,16%.

Thus, execution times required for computing explanation texts thanks to the algorithms described in the previous subsections are mostly compatible with an online use in an interactive system. However, for questions related to exponential-size transformations, we have no guarantee that they can be answered in a reasonable amount of time, shorter than 15s.

To sum up, in this section, we described our method for generating contrastive explanations. Given a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$, a two-phase checking process involving o, $\mathcal{S}$ and $\mathcal{I}$ is carried out. In Phase 1, preliminary checks are performed. These are fast checks used to assess whether some conditions are satisfied, and if not, to output proof-like negative explanation texts. In Phase 2, complete checks are conducted. These are based on the exploration of the set of neighboring solutions $\mathcal{N}(o, \mathcal{S})$. Depending on o, we use either polynomial or ILP-based algorithms to explore $\mathcal{N}(o, \mathcal{S})$, and then produce negative or positive explanation texts. Execution times required for computing contrastive explanation texts thanks to these algorithms are mostly compatible with a near-real-time use of explanations by end-users. However, for questions related to exponential-size neighborhoods, we have no guarantee that their contrastive explanation texts can be computed in a reasonable amount of time, shorter than 15s.

5.4 Generating scenario explanation texts

This section focuses on the generation of *scenario explanations*, that-is-to-say explanations describing how changes in the parameters of the current instance, suggested by end-users, affect the observation they made about the current solution (see Section 2.3 and Subsection 4.2.2). Let us then consider $\mathcal{S}$ a feasible solution of an instance $\mathcal{I}$, as well as $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$ a scenario question based on a question template of Table 4.2, where o is an observation about $\mathcal{S}$ and $\mathcal{I}'$ is a relaxation of $\mathcal{I}$. We recall that $\mathcal{I}'$ is said to be a *relaxation* of $\mathcal{I}$ if the set of feasible solutions of $\mathcal{I}$ is included in the one of $\mathcal{I}'$ (see Subsection 4.2.2).

In Chapter 4, we defined various notions for modeling the explanation process, each one of them specified for each of the three explanation types. We remarked that these notions applied to the cases of contrastive and scenario explanations are very similar, especially the notion of the decision-problem interpreted question noted $dpq(.)$: $dpq(q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}'))$ almost coincides with $dpq(q_c(o, \mathcal{S}, \mathcal{I}'))$. In line with this remark, in order to generate scenario explanation texts, we can actually use the method for generating contrastive ones and adapt it. This is detailed in the two following paragraphs.

Using the method for generating contrastive explanations... Let q be a scenario question $q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$. Let us apply the method for generating a contrastive explanation text to the contrastive question $q' = q_c(o, \mathcal{S}, \mathcal{I}')$, by following the principles described in Figure 5.1, but ignoring the part where explanation texts are built.

- Firstly, we apply preliminary checks: we check whether conditions that are necessary for the set of neighboring solutions $\mathcal{N}(o, \mathcal{S})$ to contain solutions that are feasible and better than $\mathcal{S}$ w.r.t. $\mathcal{I}'$ are satisfied.
- If preliminary checks are satisfied, then complete checks related to neighborhood exploration are performed. A support solution is computed: among the neighboring solutions of $\mathcal{N}(o, \mathcal{S})$, it is the best feasible solution or, in the absence of feasible neighboring solutions, the nearest-to-feasibility solution, where the feasibility and the distance to feasibility are evaluated w.r.t. $\mathcal{I}'$.

Since these checks are performed w.r.t. the relaxation $\mathcal{I}'$, they provide exactly the explanatory content that we need for building a scenario explanation answering q.

As the method described in Figure 5.1 is meant for contrastive explanations, we can not use the part of it which builds the explanation texts because the resulting texts would not be consistent with scenario questions. We need to change this part to build scenario explanation texts.

... and adapting the contrastive template texts to obtain scenario explanation texts. In order to obtain scenario explanation texts, we replace the template texts used within the method designed for contrastive explanations with others that are consistent with scenario questions. We need to do so within the prelimanary checks and the complete checks.

- **Regarding preliminary checks,** we adapt the template texts designed for building contrastive explanation texts related to skill and time considerations. We replace the beginning of these template texts "*In the current solution, ⟨the fact (is observed)⟩ because...*" with "*Despite the changes in the instance, ⟨meeting the foil⟩ remains impossible because ...*" For instance, the template text corresponding to preliminary checks related to skills becomes:
"*Despite the changes in the instance, ⟨meeting the foil⟩ remains impossible because ⟨employee i^*⟩ has a skill level of ⟨ske_{i^*}⟩ while ⟨task j^*⟩ has a skill level of ⟨skt_{j^*}⟩.*"
- **Regarding complete checks,** we adapt the template texts designed for building contrastive negative explanation texts about time-infeasibility and non-improvement as well as positive explanation texts as follows.

- **Case of time-infeasible support solution.** As described in Figure 5.2, in order to build contrastive negative explanation texts about time infeasibility, we concatenate four parts of texts. In order to obtain scenario ones, we simply have to change the first part. This is shown in Figure 5.4. Since the second, third and fourth parts are not affected by any changes, they are identical to the ones of Figure 5.3 and are cut from Figure 5.4.
- **Case of feasible but non-improving support solution.** Similarly, we change the first part of contrastive negative explanation texts about non-improvement, described in Figure 5.3, so as to obtain scenario explanation texts, which is depicted in Figure 5.5.
- **Case of feasible and improving support solution.** Finally, the contrastive positive explanation text is fully replaced by the following scenario positive explanation text:
"*Thanks to the changes in the instance, ⟨meeting the foil⟩ is possible and provides better solutions than the current one such as the solution obtained by ⟨applying the specific transformation to get the support solution⟩.*"

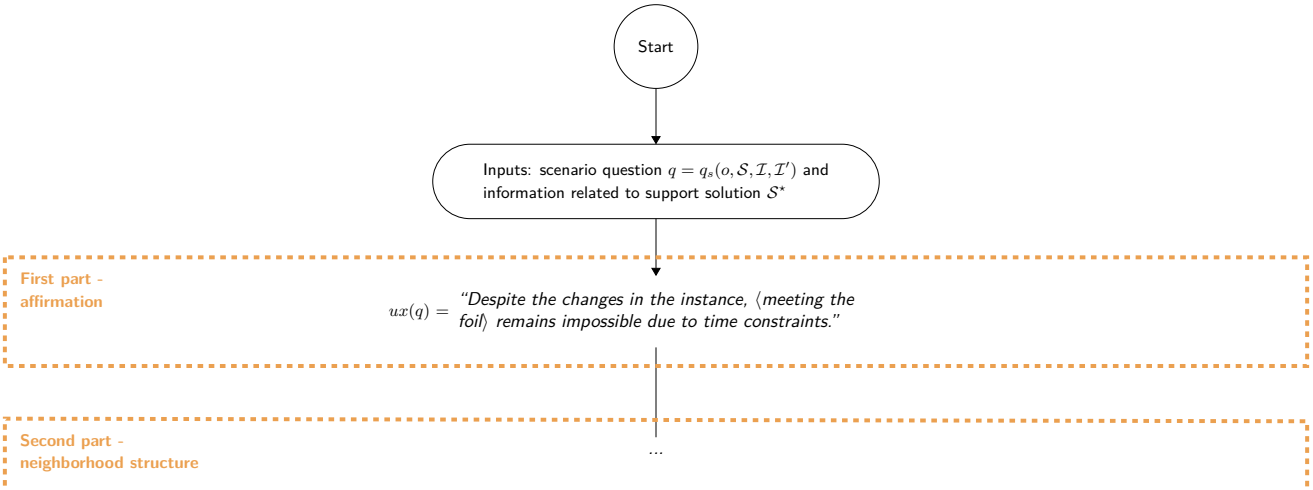


Figure 5.4: Building a scenario negative explanation texts about time-infeasibility given a support solution. Second, third and fourth parts are cut from the figure because they are identical to the ones of Figure 5.2.

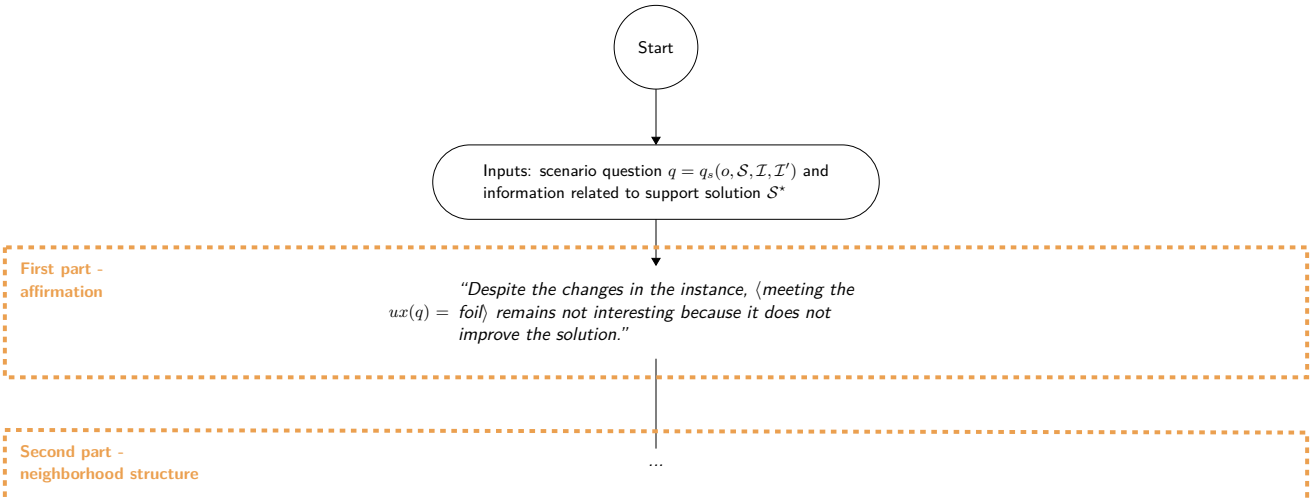


Figure 5.5: Building a scenario negative explanation text about non-improvement given a support solution. Second, third and fourth parts are cut from the figure because they are identical to the ones of Figure 5.3.

To conclude, we described in this section our method for generating scenario explanations. It is essentially based on the method related to contrastive explanations. Only some adaptations of the explanation texts were needed to output texts which are consistent with scenario questions. Moreover, as the computation part of generation scenario explanations coincides with the one of contrastive explanations, we did not carry out numerical experiments.

5.5 Generating counterfactual explanation texts

This section deals with *counterfactual explanations i.e.* explanations which aim at identifying what parameters could be changed in the current instance and by how much, so as to make a solution expected by end-users feasible and better than the current solution (see Section 2.3 and Subsection 4.2.2). Consider that end-users have a solution $\mathcal{S}$ of a WSRP instance $\mathcal{I}$, as well as a counterfactual question $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, with o an observation about $\mathcal{S}$ and $\mathcal{J}'$ a set of relaxations of $\mathcal{I}$.

Before describing the algorithmic procedure for generating counterfactual explanation texts, we recall that a positive counterfactual explanation basically confirms that end-users are right to wonder about observing o in $\mathcal{S}$ because there exists a neighboring solution $\mathcal{S}'$ that is feasible and better than $\mathcal{S}$ w.r.t. a relaxation $\mathcal{I}'$ of $\mathcal{J}'$. On the opposite, a negative counterfactual explanation basically confirms that $\mathcal{S}$ is a good quality solution because none of the neighboring solutions of $\mathcal{N}(o, \mathcal{S})$ are feasible and better than $\mathcal{S}$ w.r.t. any relaxation of $\mathcal{J}'$ and it provides a justification to that affirmation. More specifically, a proof-like negative explanation provides an exhaustive and complete justification (in other words a proof) while the argument-like negative explanation provides a convincing illustrative example (in other words an argument). See Subsection 4.3.3 for details about explanations.

Structure of the generation of counterfactual explanation texts. The algorithmic procedure for answering the counterfactual question q is represented in Figure 5.6. Like the one for answering contrastive explanations (see Figure 5.1 in Section 5.3), it consists in two parts which play similar roles to the ones they have in the contrastive case.

- **Phase 1.** *Preliminary checks* assess whether some conditions involving o, $\mathcal{S}$, $\mathcal{I}$ and $\mathcal{J}'$ are satisfied. These conditions are necessary for the set of neighboring solutions $\mathcal{N}(o, \mathcal{S})$ to contain solutions that are feasible and better than $\mathcal{S}$ with respect to at least one relaxation $\mathcal{I}'$ in $\mathcal{J}'$. Checking these conditions as preliminary checks is relevant if they can be computed in polynomial time, without having to build neighboring solutions, and if we manage to provide narratable proof-like negative explanations when these checks are not satisfied.
- **Phase 2.** If the preliminary checks are satisfied, then, *complete checks* are performed to explore simultaneously the set of relaxations $\mathcal{J}'$ and the set of neighboring solutions $\mathcal{N}(o, \mathcal{S})$ in order to look for a relaxation as well as a neighboring solution that is feasible and better than $\mathcal{S}$ with respect to this relaxation.

Then, based on the results of these checks, explanation texts are built.

Preliminary checks are discussed in the following paragraphs while complete checks are detailed in the two following subsections.

Preliminary checks. In the case of contrastive explanations, whenever the transformation from $\mathcal{S}$ to any neighboring solution of $\mathcal{N}(o, \mathcal{S})$ requires one task j^* to be newly assigned to one employee i^*, we proposed two preliminary checks: one related to skill considerations and the other to time considerations:

- employee i^* must be skilled enough to perform task j^*;
- employee i^* must be able to perform j^* as a single task in their planning, *i.e.* i^* must have enough time to leave their home location at the beginning of their working time window, travel to task j^*, perform it before the end of its availability time window, and come back home before the end of their working hours.

In the counterfactual case, while it is still possible to carry out a preliminary check and generate a proof-like negative explanation related to skill considerations, this seems more difficult for time considerations.

- Regarding the skill considerations, we can apply the preliminary check related to skill to all the relaxations of $\mathcal{J}'$. If the check fails for all of them, we can easily build a proof-like negative explanation text which stands for all the relaxations.
- Regarding the time considerations, suppose that we apply the above-mentioned preliminary check related to time to all the relaxations of $\mathcal{J}'$. If the check fails for all of them, it seems more complicated to build a proof-like negative explanation text which tells in a few sentences the time conflicts happening for every relaxation.

As a consequence, in the counterfactual case, we only carry out preliminary check related to skills.

Now that we described above the preliminary checks that we carry out within the procedure for generating counterfactual explanations, we deal with the complete checks in the two following subsections. The first subsection defines the notion of relaxation-solution pair, which plays a similar role as the support solution in the case of contrastive explanations, and describes how to compute such a pair. The second one presents how we can adapt the method that we developed in Subsection 5.3.3 for building contrastive explanation texts given a support solution to get a method for building counterfactual explanation texts given a support relaxation-solution pair. Finally, after these two subsections, we provide some numerical results about the execution time of our method for generating counterfactual explanations texts.

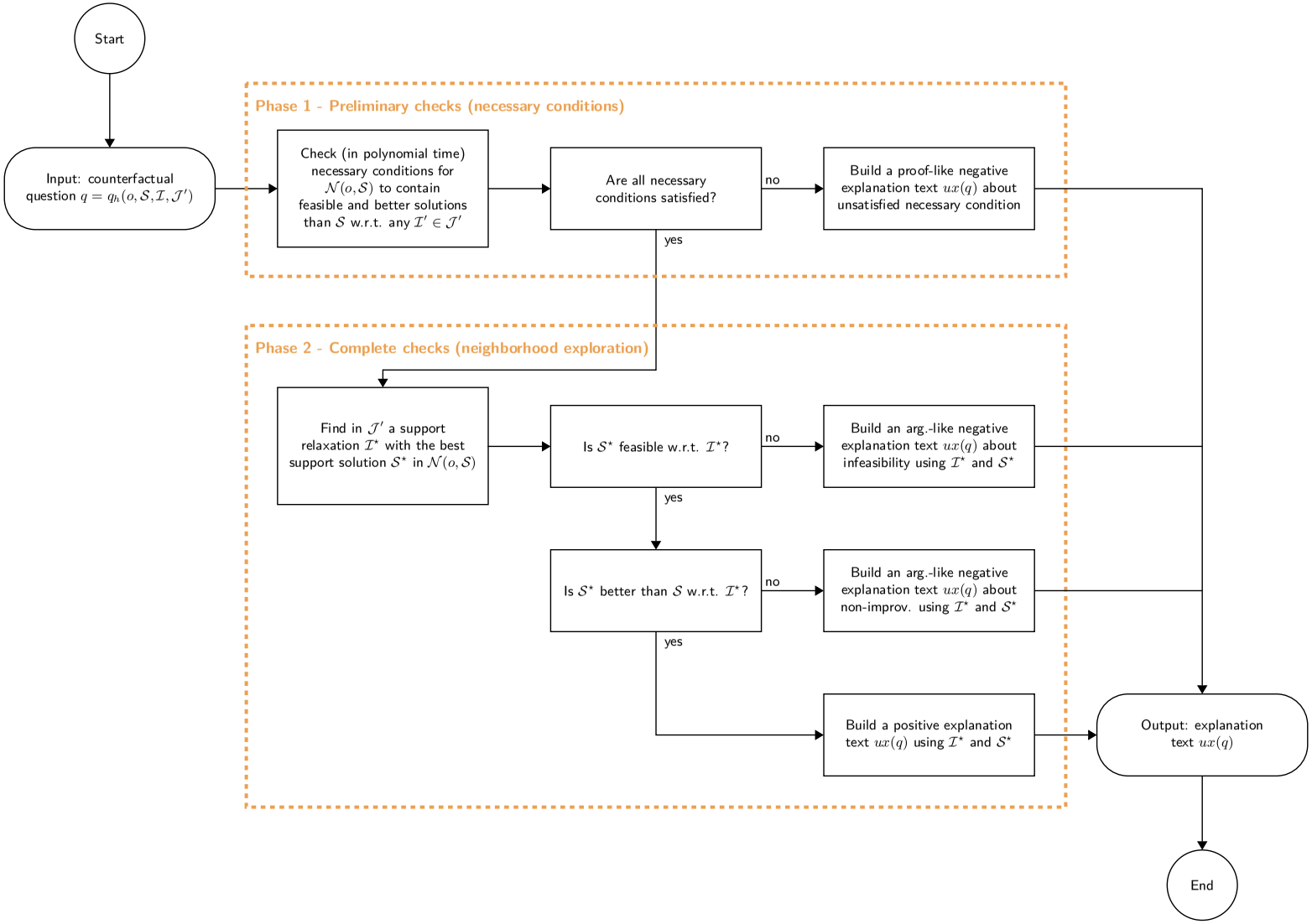


Figure 5.6: Algorithmic framework for generating counterfactual explanation texts.

5.5.1 Identifying support relaxation-solution pair

As shown in Figure 5.6, the first step within the complete checks is to identify a *support relaxation-solution pair*, which is made of a particular relaxation of $\mathcal{J}'$ along with a particular neighboring solution of $\mathcal{N}(o, \mathcal{S})$ and whose properties will be then exploited for building relevant counterfactual explanations texts.

We recall that, in Subsection 5.3.2, for the case of contrastive explanations, we introduced the notion of *support solution* which intuitively corresponds to the "best" or the "most convincing" neighboring solution to present to end-users. More precisely, the support solution is *i)* the best feasible neighboring solution in $\mathcal{N}(o, \mathcal{S})$ or *ii)* in the absence of feasible neighboring solutions in $\mathcal{N}(o, \mathcal{S})$, the *nearest-to-feasibility* neighboring solution. In this second case, the distance to feasibility of a neighboring solution is measured thanks to the *feasibility gap* (see Subsection 3.3.1) of the task insertion involved in the transformation turning $\mathcal{S}$ into this neighboring solution; in other words, the feasibility gap allows to compare neighboring solutions, to rank them and select the "best" one (relatively to the feasibility gap).

For the counterfactual case, we must have not only criteria for identifying the "best" neighboring solution but also criteria for identifying the "best" relaxation, so as to obtain a support relaxation-solution pair. In the following paragraphs, we describe our assumptions on the relaxations allowed by end-users and on their preferences regarding these relaxations. This allows us then to define the notion of support relaxation-solution pair.

Assumptions about allowed alterations for defining relaxations. In Subsection 4.2.2, we proposed that end-users wishing to ask a counterfactual question do not explicitly specify the set $\mathcal{J}'$ but rather implicitly define it by indicating a set of allowed alterations of the parameters of $\mathcal{I}$. Therefore, we assume that $\mathcal{J}'$ can be described by the choices of alterations that end-users allow to apply to the parameters of $\mathcal{I}$.

Choices of alterations depend on the application context. In some cases, it might be possible to extend the availability time window of a task; sometimes, it might be inconceivable. Ultimately, end-users should be the ones who decide which parameters can be altered or not. To illustrate our approach, in this section, we assume that end-users allow changes of the availability time window $[lbt_j,\ ubt_j]$ and duration dt_j of every task $j \in \mathcal{T}$ concerned by the observation o. Later in this section, we will discuss other possible choices of parameters to alter and how these alternative choices would impact the ILP modeling.

Assumptions about preferences between relaxations. All the relaxations of $\mathcal{I}$ are not equally satisfying. For instance, a relaxation that is obtained from the current instance $\mathcal{I}$ by applying a few small alterations of parameters is usually more satisfying than one obtained by applying numerous large alterations. However, does one prefer one large alteration to various small ones? Again, it depends on the application context.

To illustrate our approach, in this section, we assume that end-users express preferences between relaxations in terms of magnitude and number of alterations that must be applied to the current instance to obtain these relaxations. More precisely, we assume that end-users prefer to minimize, as much as possible, in the following order:

- the sum of task duration alterations as it would reduce the quality of solution;
- the largest alteration value;
- the number of alterations;
- the sum of alterations.

Thanks to these assumptions, we are now able to compare relaxations and select the "best" or "more convincing" one.

Support relaxation-solution pair. Given a counterfactual question $q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, we define a *support relaxation-solution pair*, noted $(\mathcal{I}^\star, \mathcal{S}^\star)$, made of a relaxation $\mathcal{I}^\star \in \mathcal{J}'$ and a neighboring solution $\mathcal{S}^\star \in \mathcal{N}(o, \mathcal{S})$, such that:

- $\mathcal{S}^\star$ is a support solution relatively to $\mathcal{I}^\star$;
- among all the support solutions related to each relaxation in $\mathcal{J}'$, $\mathcal{S}^\star$ has the smallest feasibility gap;
- among all the relaxations in $\mathcal{J}'$ whose support solution has the same feasibility gap as $\mathcal{S}^\star$, $\mathcal{I}^\star$ is the "best" relaxation (relatively to the assumptions about preferences detailed above).

Now that the notion of support relaxation-solution pair is defined, we can move on to computing such a pair. In the case of a contrastive explanation about an observation o related to a solution $\mathcal{S}$, in order to identify a support solution, we proposed two kinds algorithms depending on the structure of the neighborhood $\mathcal{N}(o, \mathcal{S})$: polynomial-time algorithms and ILP-based ones. However, in the case of counterfactual explanations, we exclusively propose *ILP-based algorithms* to identify the support relaxation-solution pair. Namely, in this case, algorithms must explore not only the neighborhood $\mathcal{N}(o, \mathcal{S})$ but also the set of relaxations $\mathcal{J}'$ which is potentially large, therefore analyzing $\mathcal{N}(o, \mathcal{S})$ relatively to each relaxation of $\mathcal{J}'$ would be intractable.

ILP-based algorithms developed for computing support relaxation-solution pairs use various ILP models which depend on the template of the observation o involved in the counterfactual question. All these models are based on a generic ILP model that we call *counterfactual transformation model*.

We recall that we introduced the main model $mm(\mathcal{I})$ (see Model 1 in Subsection 3.2.2) for modeling our WSRP use case, the foil model $fm(o, \mathcal{S}, \mathcal{I})$ (see Model 2 in Subsection 4.3.2) for modeling the theoretical exploration in the neighborhood $\mathcal{N}(o, \mathcal{S})$ of a solution that would be feasible and better than $\mathcal{S}$ and the contrastive transformation model $ctm(o, \mathcal{S}, \mathcal{I})$ (see Model 3 in Subsection 5.3.2) for computing support solution for contrastive explanations related to specific observation templates. In line with $ctm(o, \mathcal{S}, \mathcal{I})$, we are again interested in working with ILP models that are optimizing a single employee planning, as opposed to optimizing the family of all employee plannings as done in $mm(\mathcal{I})$. Narrowing our focus to a single employee planning leads to a drastic decrease in the size of the ILP models compared to $mm(\mathcal{I})$, which enhances the efficiency of their solving. However, this does not allow to deal with counterfactual questions and explanations related to observation templates involving all employees namely (Ins,P,c) and (Ex,P,c). Similarly, we will focus on observation templates comprising a single given task to insert, reassign, exchange or reorder. Thus, the approach developed in this section only applies to the subset of observation templates which specifies a single employee and a single task of interest in their text: (Ins,C), (Ins,P,a), (Ins,E), (Ex,C), (Ex,P,a), (Ex,E), (Ord,C,a), (Ord,P,c) and (Ord,E). Note that each of these templates specifies an employee of interest i^* as well as a task of interest j^* within their text.

In the following paragraphs, we describe first the counterfactual transformation model and then the ILP-based algorithms developed for identifying the support relaxation-solution pair.

Counterfactual transformation model. Given a counterfactual question $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, we call *counterfactual transformation model*, noted $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, the multi-objective ILP model used for finding a support relaxation-solution pair $(\mathcal{I}^\star, \mathcal{S}^\star)$, with the aim to answer q. $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ is obtained by combining Model 4 with observation-dependent parameters and constraints from Table 5.6.

- Model 4 is a generic multi-objective ILP model which aims at finding the alterations to apply to the parameters of $\mathcal{I}$ in order to obtain the support relaxation $\mathcal{I}^\star$ and 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 $\mathcal{S}^\star$. Model 4 is generic because it uses a set $\mathcal{T}^\star$ and constraint (M4.13), whose explicit mathematical contents depend on the observation template of o and need to be specified to get a fully defined model.
- The purpose of Table 5.6 is precisely to specify, for each of the above-mentioned observation templates, the subset $\mathcal{T}^\star$ and the constraints (M4.13). $\mathcal{T}^\star$ is a subset of the set of tasks $\mathcal{T}$. It contains 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^*). (M4.13) corresponds to neighboring constraints which ensure that, if the planning of employee i in $\mathcal{S}$ is changed for the planning obtained by solving Model 4, then the obtained solution is a neighboring solution of $\mathcal{N}(o, \mathcal{S})$. Neighboring constraints are noted $\varphi(\mathcal{X}) \in \mathcal{N}(o, \mathcal{S})$ in Model 4 in reference to the neighboring constraints in the foil model $fm(q)$ which play a similar role.

$htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ is similar to the contrastive transformation model $ctm(o, \mathcal{S}, \mathcal{I})$. However, it differs from it on various aspects which we detail below.

- **Similarities and differences with contrastive transformation model.** Similarly to $ctm(o, \mathcal{S}, \mathcal{I})$, and by opposition to $mm(\mathcal{I})$ which deals with optimizing a solution *i.e.* the plannings of all the employees of $\mathcal{E}$, $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ focuses on optimizing a single employee planning, namely the planning of i^*. Therefore, as for $ctm(o, \mathcal{S}, \mathcal{I})$, only a subset of the decision variables of $mm(\mathcal{I})$ are involved in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$; and there are again no sums over $\mathcal{E}$ or constraints repeated over $\mathcal{E}$ but only a focus on i^*, sums that are indexed over $\mathcal{T}$ and constraints that repeated over $\mathcal{T}$ in $mm(\mathcal{I})$ are over $\mathcal{T}^\star$, *etc.* However, compared to $ctm(o, \mathcal{S}, \mathcal{I})$, since $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ aims at optimizing alterations of instance parameters, there are new decision variables involved in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$; and constraints as well as the objective function must be adapted to this aim.
- **Decision variables.** All the decision variables of $ctm(o, \mathcal{S}, \mathcal{I})$ are also involved in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$:
  - for each task $j \in \mathcal{T}^\star$, the start time decision variable T_j, defining the time at which j starts to be performed by employee - if j is performed;
  - for each pair of activities $(j, k) \in (\mathcal{A}_{i^*} \setminus \{r_{i^*}\}) \times (\mathcal{A}_{i^*} \setminus \{d_{i^*}, j\})$, the path decision variable U_{i^*jk} is equal to 1 if i^* performs the activity j and then moves to the activity k, and to 0 otherwise;
  - the BET and FLT decision variables $T^{lb}_{j^*}$ and $T^{ub}_{j^*}$, where $T^{lb}_{j^*}$ (resp. $T^{ub}_{j^*}$) 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^*.

However, other decision variables are also involved in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$.

- First, several integer variables, that we call *altering variables*, are involved in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ to artificially alter some parameters of the instance $\mathcal{I}$. For $j \in \mathcal{T}^\star$, three integer variables ΔDt_j, ΔLBT_j and ΔUBT_j are introduced allowing respectively to reduce the duration dt_j of task j, to decrease the lower bound lbt_j of its time window and to increase its upper bound ubt_j. In addition to these altering variables, we introduce ΔT_{max}, an integer variable for measuring the greatest value taken by any of the altering variables.
- Second, for $j \in \mathcal{T}^\star$, we associate to the altering variables ΔDt_j, ΔLBT_j and ΔUBT_j, the binary variables XDt_j, $XLBT_j$ and $XUBT_j$, which are equal to 1 or 0 for indicating whether their corresponding altering variable takes a positive value or a null one.

- **Multi-objective function.** The multi-objective function (M4.1) is minimized according to a lexicographic order.
  1. The first objective aims at tightening the gap between the BET and FLT variables $T^{lb}_{j^*}$ and $T^{ub}_{j^*}$ by minimizing the difference $T^{lb}_{j^*} - T^{ub}_{j^*}$. We recall that having $T^{lb}_{j^*} = T^{ub}_{j^*}$ means that the transformation is feasible.
  2. The second objective relates to the first objective of $mm(\mathcal{I})$ as it maximizes the total working time of the planning of i^*.
  3. The third objective relates to the second objective of $mm(\mathcal{I})$ as it minimizes the total traveling time of the planning of i^*.
  4. The fourth objective minimizes the total reduction of tasks duration induced by altering variables ΔDt_j for $j \in \mathcal{T}^\star$. Its purpose is to enable decreasing the duration of the performed tasks only if extending the tasks' time windows, *via* ΔLBT_j and ΔUBT_j, is not enough for successfully inserting j^*. Such a preference for altering first time windows and then duration is applied as decreasing tasks' duration actually depreciates the solution since, by $mm(\mathcal{I})$, the primary objective is to maximize the total tasks duration performed by the employees.
  5. The purpose of the fifth objective which minimizes ΔT_{max} is to prevent as much as possible large alterations of the parameters.
  6. The sixth objective seeks to minimize the number of parameters alterations.
  7. The last objective seeks to minimize the sum of parameters alterations.
- **Constraints.** The constraints of $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ are essentially the same as the ones of $ctm(o, \mathcal{S}, \mathcal{I})$ with possibly some adaptations.
  - Flow constraints (M4.2) to (M4.4) are the same as constraints (M3.2) to (M3.4).
  - As for $ctm(o, \mathcal{S}, \mathcal{I})$, there is no equivalent of the skill constraints (M1.5) of $mm(\mathcal{I})$ in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$.
  - Occurrence constraints (M4.6) are the same as (M3.6).
  - Availability, working hours and sequencing constraints spanning labels from (M3.7.a) to (M3.11.b) in $ctm(o, \mathcal{S}, \mathcal{I})$ are also involved in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ with labels spanning from (M4.7.a) to (M4.11.b), with some changes. Basically, all these constraints of $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ are obtained from the one $ctm(o, \mathcal{S}, \mathcal{I})$ as follows: wherever an instance parameter that is subject to alteration is included in the constraint expression, its corresponding alteration variable is added to this expression.
  - Constraints (M4.12.a) and (M4.12.b) are the same as (M3.12.a) and (M3.12.b).
  - Neighboring constraints (M4.13) are the same as (M3.13) and are detailed in Table 5.6.

  However, a few other constraints are also involved in $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$.
  - Constraints (M4.14.a) to (M4.14.c) ensure that the binary variables XDt_j, $XLBT_j$ and $XUBT_j$ for $j \in \mathcal{T}^\star$ play their expected role, namely indicating whether their corresponding altering variable take a positive value or a null one. Besides, these constraints limit the value of the altering variables ΔDt_j, ΔLBT_j and ΔUBT_j for $j \in \mathcal{T}^\star$ to some "reasonable bounds": the duration dt_j of a task j can not be reduced by more than its own value; there is no point in having the lower and upper bounds of the availability time window of task respectively smaller and higher than the lower and upper bounds of the working time window of employee i^*.
  - Constraint (M4.15) ensures that ΔT_{max} measures as expected the greatest value taken by any of the altering variables.

ILP-based algorithms for computing support relaxation-solution pair. Given a counterfactual question $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, we propose an *ILP-based algorithm* for computing a support relaxation-solution pair $(\mathcal{I}^\star, \mathcal{S}^\star)$ used for answering the question q. Regardless of the observation template of o, this algorithm is described in Algorithm 5.3. It consists in solving (at line 1) the counterfactual transformation model $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ which we described above. The results of this solving are then used to build the support instance (at lines 2 and 3) and the support solution (at lines 5 and 6).

Before ending this subsection, let us comeback to the discussion, that we started in the beginning of this Subsection, about the way $\mathcal{I}$ can be altered, in particular which parameters can be changed and up to what bounds.

$$lex\ min\ \Big(\ T^{lb}_{j^*} - T^{ub}_{j^*},\quad -\sum_{j\in\mathcal{T}^*}\ \sum_{k\in\mathcal{A}_{i^*},\ k\neq d_{i^*},j} U_{i^*jk}\ dt_j,\quad \sum_{j\in\mathcal{A}_{i^*},\ j\neq r_{i^*}}\ \sum_{k\in\mathcal{A}_{i^*},\ k\neq d_{i^*},j} U_{i^*jk}\ tr_{jk},$$
$$\sum_{j\in\mathcal{T}^*}\Delta Dt_j,\quad \Delta T_{max},\quad \sum_{j\in\mathcal{T}^*} XDt_j + XLBT_j + XUBT_j,\quad \sum_{j\in\mathcal{T}^*}\Delta Dt_j + \Delta LBT_j + \Delta UBT_j\ \Big) \tag{M4.1}$$

$s.t.$

$$\sum_{k\in\mathcal{A}_{i^*},\ k\neq d_{i^*}} U_{i^*d_{i^*}k} = 1 \tag{M3.2} \equiv (M4.2)$$

$$\sum_{j\in\mathcal{A}_{i^*},\ j\neq r_{i^*}} U_{i^*jr_{i^*}} = 1 \tag{M3.3} \equiv (M4.3)$$

$$\sum_{j\in\mathcal{A}_{i^*},\ j\neq k, r_{i^*}} U_{i^*jk} = \sum_{j'\in\mathcal{A}_{i^*},\ j'\neq d_{i^*},k} U_{i^*kj'} \qquad \forall\ k\in\mathcal{T}^* \tag{M3.4} \equiv (M4.4)$$

$$\sum_{k\in\mathcal{A}_{i^*},\ k\neq d_{i^*},j} U_{i^*jk} \leq 1 \qquad \forall\ j\in\mathcal{T}^* \tag{M3.6} \equiv (M4.6)$$

$$lbt_j - \Delta LBT_j \leq T_j \qquad \forall\ j\in\mathcal{T}^*\setminus\{j^*\} \tag{M4.7.a}$$

$$lbt_{j^*} - \Delta LBT_{j^*} \leq T^{lb}_{j^*} \tag{M4.7.b}$$

$$T_j \leq ubt_j + \Delta UBT_j - dt_j + \Delta Dt_j \qquad \forall\ j\in\mathcal{T}^*\setminus\{j^*\} \tag{M4.8.a}$$

$$T^{ub}_{j^*} \leq ubt_{j^*} + \Delta UBT_{j^*} - dt_{j^*} + \Delta Dt_{j^*} \tag{M4.8.b}$$

$$lbe_{i^*} + tr_{d_{i^*}k} \leq T_k \qquad \forall\ k\in\mathcal{T}^*\setminus\{j^*\} \tag{M4.9.a}$$

$$lbe_{i^*} + tr_{d_{i^*}j^*} \leq T^{lb}_{j^*} \tag{M4.9.b}$$

$$T_j + dt_j - \Delta Dt_j + U_{i^*jk}\ tr_{jk} \leq T_k + \Big(1 - U_{i^*jk}\Big)ubt_j \qquad \forall\ j\neq k\in\mathcal{T}^*\setminus\{j^*\} \tag{M4.10.a}$$

$$T^{ub}_{j^*} + dt_{j^*} - \Delta Dt_{j^*} + U_{i^*j^*k}\ tr_{j^*k} \leq T_k + \Big(1 - U_{i^*j^*k}\Big)ubt_{j^*} \qquad \forall\ k\in\mathcal{T}^*\setminus\{j^*\} \tag{M4.10.b}$$

$$T_j + dt_j - \Delta Dt_j + U_{i^*jj^*}\ tr_{jj^*} \leq T^{lb}_{j^*} + \Big(1 - U_{i^*jj^*}\Big)ubt_j \qquad \forall\ j\in\mathcal{T}^*\setminus\{j^*\} \tag{M4.10.c}$$

$$T_j + dt_j - \Delta Dt_j \leq U_{i^*jr_{i^*}}(ube_{i^*} - tr_{jr_{i^*}}) + \Big(1 - U_{i^*jr_{i^*}}\Big)ubt_j \qquad \forall\ j\in\mathcal{T}^*\setminus\{j^*\} \tag{M4.11.a}$$

$$T^{ub}_{j^*} + dt_{j^*} - \Delta Dt_{j^*} \leq U_{i^*j^*r_{i^*}}(ube_{i^*} - tr_{j^*r_{i^*}}) + \Big(1 - U_{i^*j^*r_{i^*}}\Big)ubt_{j^*} \tag{M4.11.b}$$

$$T^{lb}_{j^*} - T^{ub}_{j^*} \geq 0 \tag{M3.12.a} \equiv (M4.12.a)$$

$$T^{ub}_{j^*} \leq T_{j^*} \leq T^{lb}_{j^*} \tag{M3.12.b} \equiv (M4.12.b)$$

$$\varphi(\mathcal{X}) \in \mathcal{N}(o,\mathcal{S}) \tag{M3.13} \equiv (M4.13)$$

$$\Delta Dt_j \leq XDt_j\ dt_j \qquad \forall\ j\in\mathcal{T}^* \tag{M4.14.a}$$

$$\Delta LBT_j \leq XLBT_j\ max(lbt_j - lbe_{i^*},\ 0) \qquad \forall\ j\in\mathcal{T}^* \tag{M4.14.b}$$

$$\Delta UBT_j \leq XUBT_j\ max(ube_{i^*} - ubt_j,\ 0) \qquad \forall\ j\in\mathcal{T}^* \tag{M4.14.c}$$

$$\Delta Dt_j,\ \Delta LBT_j,\ \Delta UBT_j \leq \Delta T_{max} \qquad \forall\ j\in\mathcal{T}^* \tag{M4.15}$$

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

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

$$T^{lb}_{j^*},\ T^{ub}_{j^*}\in\mathbb{N}$$

$$XDt_j,\ XLBT_j,\ XUBT_j\in\{0,1\} \qquad \forall\ j\in\mathcal{T}^*$$

$$\Delta Dt_j,\ \Delta LBT_j,\ \Delta UBT_j\in\mathbb{N} \qquad \forall\ j\in\mathcal{T}^*$$

$$\Delta T_{max}\in\mathbb{N}$$

Model 4: Counterfactual transformation model $htm(o,\mathcal{S},\mathcal{I},\mathcal{J}')$, *i.e.* multi-objective ILP model used to identify instance parameter alterations and a support planning for a specific employee (mentioned in o) so as to build a support relaxation-solution pair for answering the counterfactual question $q_h(o,\mathcal{S},\mathcal{I},\mathcal{J}')$.

Labels	Counterfactual questions	Subset of tasks $\mathcal{T}^\star$	Neighborhood constraints
(Ins,C)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ just after $\langle$activity $k^*\rangle$?"	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	With k' the activity after k^* in $\mathcal{R}_{i^*}$, $\begin{cases} U_{i^*jk} = 1 \quad \forall\ (j,k) \text{ consecutive in } \mathcal{R}_i \\ \qquad\qquad\qquad \neq (k^*, k') \\ U_{i^*k^*j^*} = U_{i^*j^*k'} = 1 \end{cases}$
(Ins,P,a)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ between any pair of consecutive activities of their planning?"	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} = \lvert\mathcal{R}_{i^*}\rvert - 2 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = 1 \qquad \forall\ j \in \mathcal{T}^\star \end{cases}$
(Ins,E)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ in addition to the activities of their planning?"	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} < \lvert\mathcal{R}_{i^*}\rvert - 2 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = 1 \qquad \forall\ j \in \mathcal{T}^\star \end{cases}$
(Ex,C)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ in place of $\langle$task $k^*\rangle$?"	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	With k_1 and k_2 the activities before and after k^* in $\mathcal{R}_{i^*}$, $\begin{cases} U_{i^*jk} = 1 \quad \forall\ (j,k) \text{ consecutive in } \mathcal{R}_i \\ \qquad\qquad\qquad \neq (k_1, k^*), (k^*, k_2,) \\ U_{i^*k_1j^*} = U_{i^*j^*k_2} = 1 \end{cases}$
(Ex,P,a)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ in place of any of the activities of their planning?"	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} = \lvert\mathcal{R}_{i^*}\rvert - 3 \\ \sum\limits_{j \in \mathcal{T}^\star \setminus \{j^*\}} \ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = \lvert\mathcal{R}_{i^*}\rvert - 3 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j^*\}} U_{i^*j^*k} = 1 \end{cases}$
(Ex,E)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ rather than any of their tasks (even if it means changing the order of the activities)?"	$(\mathcal{T} \cap \mathcal{R}_{i^*}) \cup \{j^*\}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} < \lvert\mathcal{R}_{i^*}\rvert - 3 \\ \sum\limits_{j \in \mathcal{T}^\star \setminus \{j^*\}} \ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = \lvert\mathcal{R}_{i^*}\rvert - 3 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j^*\}} U_{i^*j^*k} = 1 \end{cases}$
(Ord,C,a)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ later in their planning, just after $\langle$task $k^*\rangle$?"	$\mathcal{T} \cap \mathcal{R}_{i^*}$	With j_1 and j_2 the activities before and after j^* as well as k' after k^* in $\mathcal{R}_{i^*}$, $\begin{cases} U_{i^*jk} = 1 \quad \forall\ (j,k) \text{ consecutive in } \mathcal{R}_i \\ \qquad\qquad\qquad \neq (k_1, k^*), (k^*, k_2,) \\ U_{i^*k_1j^*} = U_{i^*j^*k_2} = 1 \end{cases}$
(Ord,P,c)	"How to make $\langle$employee $i^*\rangle$ perform $\langle$task $j^*\rangle$ at any other step in their planning?"	$\mathcal{T} \cap \mathcal{R}_{i^*}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} = \lvert\mathcal{R}_{i^*}\rvert - 4 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = 1 \qquad \forall\ j \in \mathcal{T}^\star \end{cases}$
(Ord,E)	"How to make $\langle$employee $i^*\rangle$ perform the activities of their planning in a different order?"	$\mathcal{T} \cap \mathcal{R}_{i^*}$	$\begin{cases} \sum\limits_{\substack{j,k \in \mathcal{R}_i \\ \text{consecutive}}} U_{i^*jk} < \lvert\mathcal{R}_{i^*}\rvert - 4 \\ \sum\limits_{k \in \mathcal{A}_{i^*} \setminus \{j\}} U_{i^*jk} = 1 \qquad \forall\ j \in \mathcal{T}^\star \end{cases}$

Table 5.6: Parameters and constraints involved in Model 4 for various possible observation templates. In the table content, $\mathcal{S}$ is the feasible solution of the instance $\mathcal{I}$ both given as inputs in Algorithm 5.3.

Algorithm 5.3: ILP-based algorithm for computing a support relaxation-solution pair

Inputs:
$\mathcal{I}$ an instance
$\mathcal{S}$ a feasible solution of $\mathcal{I}$
o an observation about $\mathcal{S}$ (which specifies who is employee i^*)
$\mathcal{J}'$ a set of relaxations of $\mathcal{I}$

1 Solve the counterfactual transformation model $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$
2 Obtain the values of the altering decision variables resulting from solving $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$
3 Build the support relaxation $\mathcal{I}^\star$ by applying these instance parameter alterations to $\mathcal{I}$
4 Obtain the values of the BET and FLT decision variables $T^{lb}_{j^*}$ and $T^{ub}_{j^*}$, and store them respectively as BET $st^b_{j^*}$ and FLT $st^f_{j^*}$
5 Build the support planning $(\mathcal{R}^\star_{i^*}, \mathcal{C}^\star_{i^*})$ of employee i^* given the values of spatial and temporal decision variables resulting from solving $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ and save related information:
task j^* is inserted in the planning of employee i^* after its k^{th} activity, with BET $st^b_{j^*}$ and FLT $st^f_{j^*}$
6 Create support solution $\mathcal{S}^\star$ by copying $\mathcal{S}$ and replacing the planning of i^* with the support planning $(\mathcal{R}^\star_{i^*}, \mathcal{C}^\star_{i^*})$

Output:
$(\mathcal{I}^\star, \mathcal{S}^\star)$ a support relaxation-solution pair (as well as the alterations at line 2 and the information at line 5)

Discussions about allowed alterations. It is clear that there are many ways to alter the parameter of $\mathcal{I}$. In this paper, we chose to locate the potential alterations on task availability time window and duration data, to allow such alterations on all the tasks of $\mathcal{T}^\star$ and to limit such alterations to some "reasonable bounds". However, one could also choose, for instance, to locate alterations on employee working time window data in addition to the ones on the task data, to allow task alterations only for some selected tasks of $\mathcal{T}^\star$, and to bound altering variables with arbitrary smaller bounds than our "reasonable bounds".

Still, such other choices could actually be taken into account in the counterfactual transformation model $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ considering some adaptations. One would have *i)* to introduce a pair of integer decision variables ΔLBE_{i^*} and ΔUBE_{i^*}, as well as their corresponding binary decision variables $XLBE_{i^*}$ and $XUBE_{i^*}$, for altering the time window of i^*; *ii)* to only consider ΔLBT_j, ΔUBT_j and ΔDt_j for the selected tasks of $\mathcal{T}^\star$; *iii)* to adapt the fourth objective in the multi-objective function; *iv)* to adapt constraints (M4.7.a) to (M4.11.b) to involve ΔLBE_{i^*} and ΔUBE_{i^*} as well as ΔLBT_j, ΔUBT_j and ΔDt_j only for the selected tasks of $\mathcal{T}^\star$; *v)* replace the "reasonable bounds" in constraints (M4.14.a) to (M4.14.c) with arbitrarily chosen bounds; *etc.*

Thus, we can let the end-user - who is actually the person who knows the best the application context of the WSRP solved - choose the locations and the bounds of the potential alterations to apply on the parameters of $\mathcal{I}$ and still be able to define a counterfactual transformation model $htm(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ adapted to such choices.

In this subsection, given a counterfactual question $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, we described how we compute the support relaxation-solution pair $(\mathcal{I}^\star, \mathcal{S}^\star)$ by resorting to ILP-based algorithms. In the following subsection, we describe how we use $(\mathcal{I}^\star, \mathcal{S}^\star)$ in order to build an explanation text $ux(q)$ answering q.

5.5.2 Building counterfactual explanation texts from support relaxation-solution pair

In Subsection 5.3.3, we developed a method which, given a support solution, builds contrastive explanations texts by concatenating pieces of template texts and by using information related to the support solution. We drew a distinction between three cases: either the support solution is infeasible, either it is feasible but not better than the current solution, or it is feasible and better than the current solution (in each case, with respect to the current instance).

In Section 5.4, we adapted this method, more specifically the pieces of texts involved in the method, to build scenario explanation texts which are consistent with the relaxation $\mathcal{I}'$.

In order to obtain a method which given a support relaxation-solution pair build counterfactual explanations texts, similarly to the scenario case, we adapt the pieces of texts involved in the method related to contrastive explanation texts.

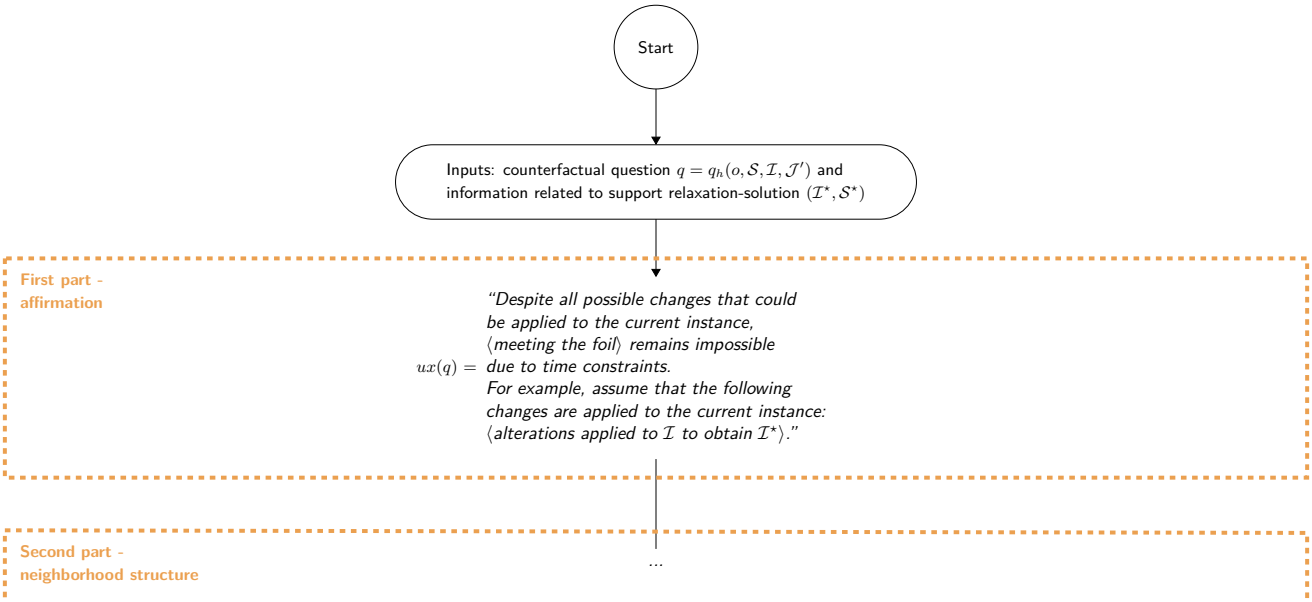


Figure 5.7: Building a counterfactual negative explanation texts about time-infeasibility given a support relaxation-solution pair. Second, third and fourth parts are cut from the figure because they are identical to the ones of Figure 5.2.

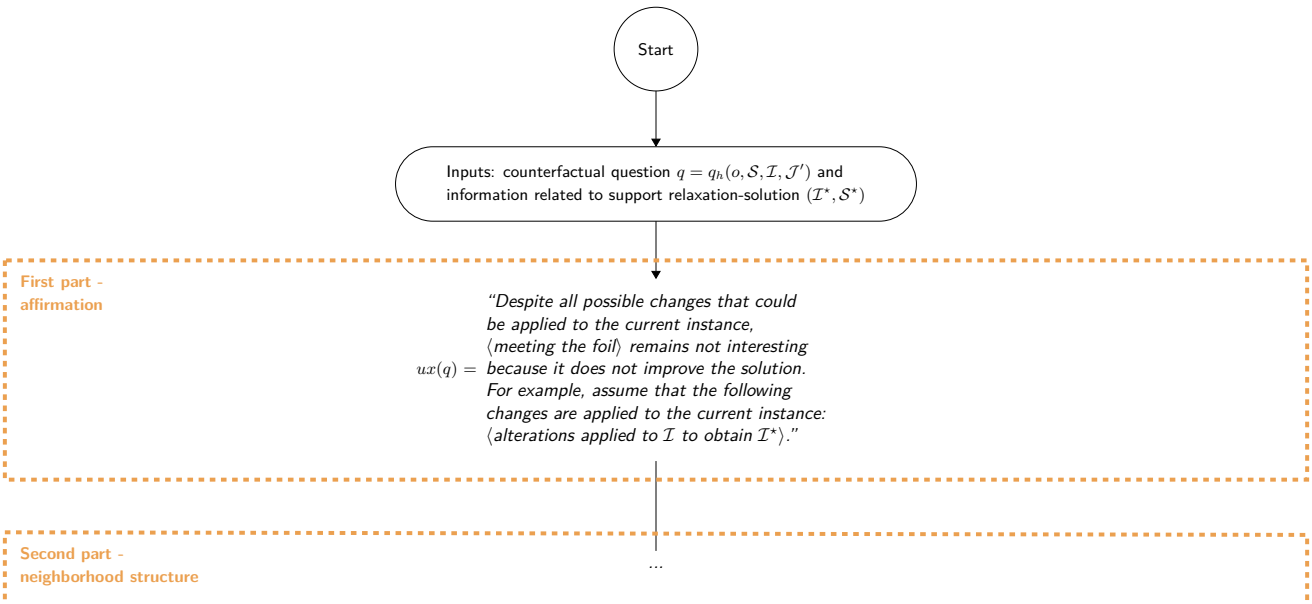


Figure 5.8: Building a counterfactual negative explanation text about non-improvement given a support relaxation-solution pair. Second, third and fourth parts are cut from the figure because they are identical to the ones of Figure 5.3.

Adapting the method for building contrastive explanations texts from support solution to the counterfactual case. Given a counterfactual question $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ and its corresponding support relaxation-solution pair $(\mathcal{I}^\star, \mathcal{S}^\star)$, we propose a method for building counterfactual explanation texts that is adapted from the one for building contrastive explanation texts. It also consists in filling in predefined template texts with information obtained from the support content as well as typical expressions (see Section 5.2). Similarly to the contrastive case, we draw a distinction between three cases: either the support solution is infeasible, either it is but not better than the current solution, or it is feasible and better than the current solution (in each case, with respect to the support relaxation). We describe our method for each of these cases below.

- **Case of infeasible support solution.** We adapt the method for building contrastive negative explanation texts about time-infeasibility, which consists in concatenating four parts of texts as described in Figure 5.2, by changing the first of these parts so as to obtain counterfactual explanation texts. This adaptation is shown in Figure 5.7. Since only the first part of the template texts concatenation is affected by changes, the second, third and fourth parts are cut from the figure, they are identical to the ones of Figure 5.3.
- **Case of feasible but non-improving support solution.** Similarly, we adapt the method for building contrastive negative explanation texts about non-improvement described in Figure 5.3 so as to obtain counterfactual explanation texts which is presented in Figure 5.8.
- **Case of feasible and improving support solution.** Finally, the counterfactual positive explanation text can be adapted as follows:
  "*By applying the following changes to the current instance* ⟨*alterations applied to* $\mathcal{I}$ *to obtain* $\mathcal{I}^\star$⟩, ⟨*meeting the foil*⟩ *is possible and provides better solutions than the current one such as the solution obtained by* ⟨*applying the specific transformation to get the support solution*⟩."

Within the last subsections, we presented how we generate counterfactual explanation texts. In the next and last subsection, we study the performances of this generation.

5.5.3 Numerical experiments

In Subsection 5.3.4, we presented the numerical study conducted to assess the computation times needed for generating contrastive explanation texts on large-scale WSRP instances and solutions. In this subsection, we continue this numerical study but focus this time on the generation of counterfactual explanation texts. The purpose remains the same: in the perspective of integrating explanation text generation into a system where end-users can request and obtain explanations in near-real-time, we seek that the algorithms designed for generating counterfactual explanation texts operate within a time-frame compatible with such usage. In the following paragraphs, we present first our experimental setting, then the obtained results and finally our analysis of these results.

Experimental setting. Most of the experimental setting contents and principles used in the study of contrastive explanations are retained for the study of counterfactual explanations: *i)* we conduct numerical experiments using the same 96 pairs of instances and solutions; *ii)* for each instance-solution pair and for each question template, we prepare a random sample of 40 questions; *iii)* we limit the time allocated for computing the explanation text answering each of these questions to 15s. However, some aspects of the experimental setting of the counterfactual explanation study are different from the one of the contrastive explanation study. First, we conduct these numerical studies only for the question template that are listed in Table 5.6, not all the labels listed in Table 4.1. This limitation is due to the inability of the counterfactual transformation model to handle questions involving all employees, namely questions based on (Ins,P,c) and (Ex,P,c) templates. Second, since we are conducting the numerical studies with counterfactual questions, not contrastive questions, we need to specify within the counterfactual questions the allowed instance parameter alterations. Indeed, any counterfactual question template text includes a field for specifying the allowed instance parameter alterations, which is not the case of the corresponding contrastive question template text. We decide to allow alterations in line with the assumptions presented in Subsection 5.5.1: alterations are exclusively allowed on task availability time-window bounds and task durations, and are limited to with "reasonable bounds".

Results. Table 5.7 presents the results of our computation time study about our method for generating counterfactual explanation texts using algorithms based on Algorithm 5.3. Specifically, the table provides some statistics about these computation times. To calculate these statistics, we aggregate the computation times obtained for all the instance-solution pairs and for all the random samples of questions by question template. For each question template, the table presents first the completed computations rate, *i.e.* the proportion of computations completed in less than 15s; the table then presents the first quartile, median, third quartile and maximum of the computation times. For example, among all the computations of counterfactual explanation texts answering questions based on (Ins,C) template, 100% have been completed in less than 15s, at least 25% haven taken less than 0,057s, at least 50% less than 0,079s, at least 75% less than 0,120s and the

longest computation has taken 0,378s. Given a question template, if computations of counterfactual explanation texts have been interrupted due to the 15s time limit, then quartiles may be unknown and replaced by "$>$ 15s". For example, since less than 75% of the computations related to (Ins,E) have been completed, the third quartile and the maximum are unknown.

Question template	**Completed rate**	**Computation times**				
		1sr quartile	**Median**	**3rd quartile**	**Maximum**	**Average**
	%	in s	in s	in s	in s	in s
(Ins,C)	100	0,057	0.079	0,120	0,378	0,091
(Ins,P,a)	100	0,063	0,089	0,132	0,413	0,100
(Ins,E)	73,72	0,202	1,067	$>$ 15	$>$ 15	-
(Ex,C)	100	0,055	0,076	0,119	0,358	0,089
(Ex,P,a)	100	0,069	0,100	0,150	0,324	0,112
(Ex,E)	71,89	0,195	1,044	$>$ 15	$>$ 15	-
(Ord,C,a)	100	0,057	0,082	0,126	0,233	0,092
(Ord,P,c)	100	0,059	0,084	0,127	0,278	0,094
(Ord,E)	92,27	0,087	0,147	0,302	$>$ 15	-

Table 5.7: Statistics about the counterfactual explanation computation times.

Analysis and insights. Experiments show that the counterfactual explanation computations related to constant-size and polynomial-size transformations are always completed while the ones related to exponential-size transformations may be interrupted. Such differences could be anticipated. In the case of constant-size and polynomial-size transformations, the neighboring constraints of the counterfactual transformation model ensure that the order of the activities in the employee planning does not changed. Moreover, in the case of constant-size transformations, they even fix the values of some path decision variables. On the opposite, in the case of exponential-size transformations, the neighboring constraints allow the activities to be freely ordered in the employee planning. Therefore, the neighboring constraints reduce significantly the feasible solution space of the counterfactual transformation model in the case of constant-size and polynomial-size transformations compared to the case of exponential-size ones. This can help to significantly fasten the solving process.

Regarding constant-size and polynomial-size transformations, experiments show that related counterfactual explanation texts are computed in a very short time: less than 0,5s. A similar conclusion was drawn in the case of contrastive explanations (see Subsection 5.3.4), Still, counterfactual explanation computations are about 2 to 4 times longer than the contrastive ones (*e.g.* on average, (Ins,C) contrastive explanation texts require 0,025s to be computed while counterfactual ones require 0,091s). Such a computation time increase, between contrastive and counterfactual explanations, was expected since counterfactual explanation computations require to solve an ILP model (in order to explore the set of allowed alterations of instance parameters) while corresponding contrastive explanation computations only required polynomial-time algorithms. Moreover, among question templates related to a given transformation topic (insertion, exchange or reordering), counterfactual explanation texts related to polynomial-size transformations generally require more time to be computed than the ones related to constant-size transformations (*e.g.* on average, (Ins,P,a) texts require 0,100s to be computed while (Ins,C) texts require 0,091s). A similar trend was noticed in the case of contrastive explanations, which was expected due to the increase of computational complexity. In the case of counterfactual explanations, this trend could also be expected since the neighboring constraints of the counterfactual transformation model related to constant-size transformations fix many values of path decision variables, hence reducing significantly the feasible solution space of the model, by comparison to the case of polynomial-size transformations.

Regarding question templates related to exponential-size transformations, whatever the template, experiments show that more than 50% of the explanation text computations last for a short time: less than 1,2s. However, a significant proportion of these computations have been interrupted as they have reached the time limit of 15s. About 70-75% of the explanation text computations related to (Ins,E) and (Ex,E) templates are completed in less than 15s and about 90-95% of the explanation text computations related to (Ord,E) templates are completed. These results are quite similar to the ones obtained when computing contrastive explanations related to exponential-size transformations. In other words, here, the exploration of instance parameter alterations does not seem to significantly increase the computation time between contrastive and counterfactual explanations.

To conclude, execution times needed for computing counterfactual explanation texts thanks to our algorithms are mostly compatible with an online use in an interactive system. However, for questions related to exponential-size transformations, we have no guarantee that their corresponding counterfactual explanation texts can be computed in a reasonable amount of time, shorter than 15s. Further investigations would be necessary, including *i)* to explore whether there exists a correlation between certain instance-solution characteristics and fluctuations in computation times, *ii)* to study the size of the counterfactual transformation models between various instance-solution pairs and various question templates, *iii)* to gauge the impact of the choice of allowed instance parameter alterations (number and magnitude) on computation times.

5.6 Conclusion

In this chapter, we described our method for generating explanations texts in response to end-user questions for each of the three types of explanations, namely contrastive, scenario and counterfactual. Regardless of the type, our approach consists essentially in two parts.

- First, given an end-user question q, checks are carried out in order to identify a mathematical content from which useful information can be extracted for writing an explanation text $ux(q)$. In the case of a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$ or a scenario question $q = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$ with $\mathcal{I}'$ relaxation of $\mathcal{I}$, this mathematical content is a solution that we call support solution $\mathcal{S}^\star$. Among all the solutions of the neighborhood $\mathcal{N}(o, \mathcal{S})$, $\mathcal{S}^\star$ is the best feasible neighboring solution or, in the absence of such feasible neighboring solutions, the nearest-to-feasibility neighboring solution, where the feasibility is measured with respect to the instance $\mathcal{I}$ if q is contrastive or the relaxation $\mathcal{I}'$ if q is scenario. In the case of counterfactual explanations $q = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$ with $\mathcal{J}'$ set of relaxations of $\mathcal{I}$, the mathematical content to identify is a pair made of a relaxation from $\mathcal{J}'$ and a neighboring solution, that we call support relaxation-solution pair $(\mathcal{I}^\star, \mathcal{S}^\star)$. Among all the pairs $(\mathcal{I}', \mathcal{S}')$ with $\mathcal{I}'$ in $\mathcal{J}'$ and $\mathcal{S}'$ in $\mathcal{N}(o, \mathcal{S})$ support solution of $\mathcal{I}'$, $(\mathcal{I}^\star, \mathcal{S}^\star)$ is the pair such that $\mathcal{S}^\star$ has the smallest feasibility gap and $\mathcal{I}^\star$ has the least alterations in terms of magnitude and number. Depending on the type of q and depending on the structure of the neighborhood $\mathcal{N}(o, \mathcal{S})$, polynomial-time or ILP-based algorithms are used to compute this support content (support solution $\mathcal{S}^\star$ or support relaxation-solution pair $(\mathcal{I}^\star, \mathcal{S}^\star)$).
- Second, an explanation text $ux(q)$ is built by concatenating pieces of template texts with fields to fill in with information obtained from the support content as well as predefined observation-dependent expressions that we call typical expressions. The choice of pieces of template texts to assemble depends on the type of q, the feasibility and quality of the support solution $\mathcal{S}^\star$ as well as the structure of the neighborhood $\mathcal{N}(o, \mathcal{S})$. Such a method allows us to build explanation texts corresponding to positive, proof-like negative or argument-like negative explanations. Figures 5.9 and 5.10 synthesize this method in the case of negative explanations based on the infeasibility or the non-improvement of the neighboring solutions given any type of explanations.

Numerical studies show that the computations of explanation texts are mostly performed under 15s, which we consider as an execution time that is compatible with end-users usage of explanations. However, for explanation texts computed with ILP-based algorithms, we have no guarantee that their corresponding explanation texts can be computed in a reasonable amount of time, shorter than 15s.

Now that we have methods for generating explanations that are adapted to end-users, we propose in Chapter 6 to design a system integrating these methods and implement a prototype of graphic user interface such that end-users can practically ask questions about their solutions and obtain explanations in return.

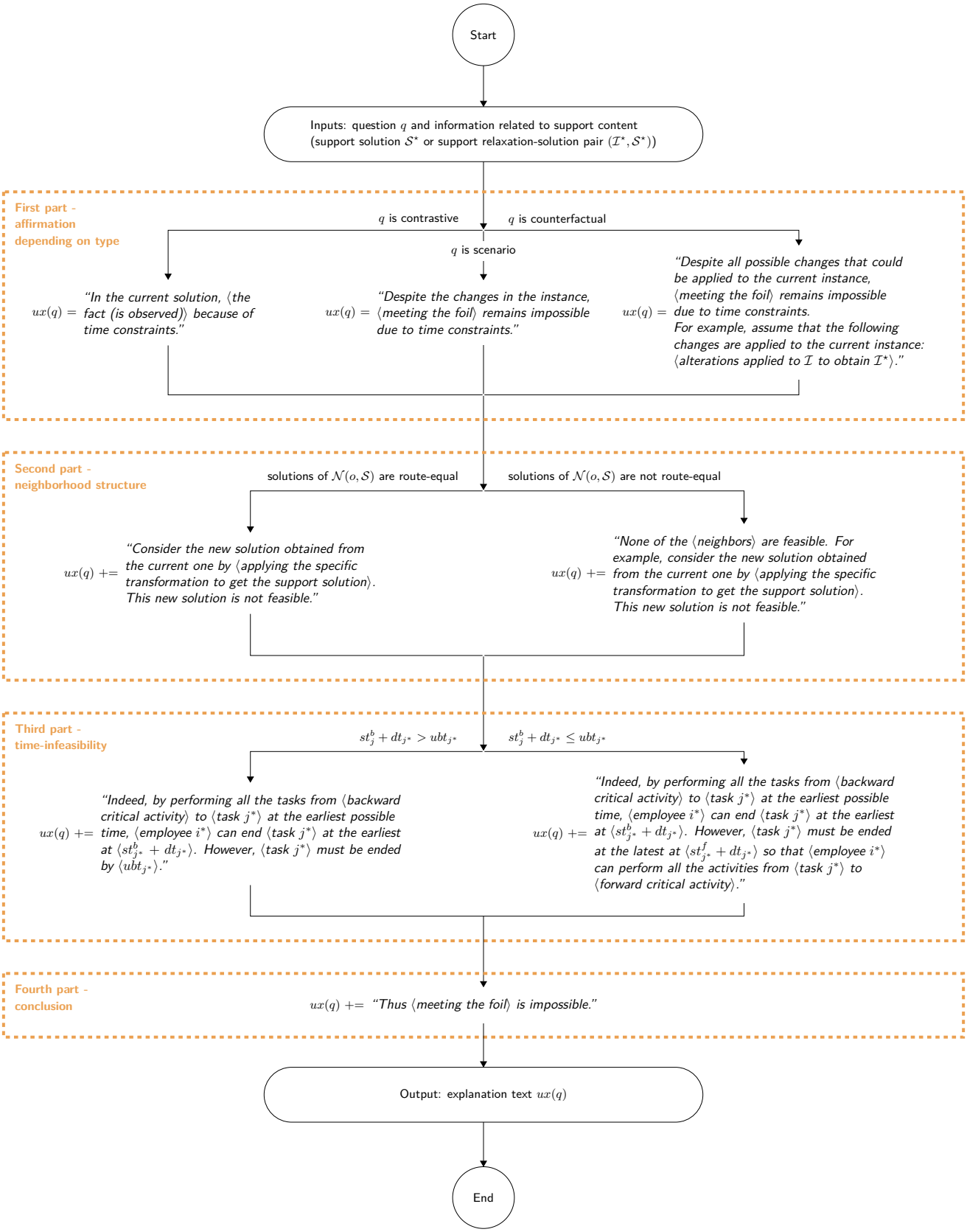


Figure 5.9: Building a negative explanation text about time-infeasibility given any type of question.

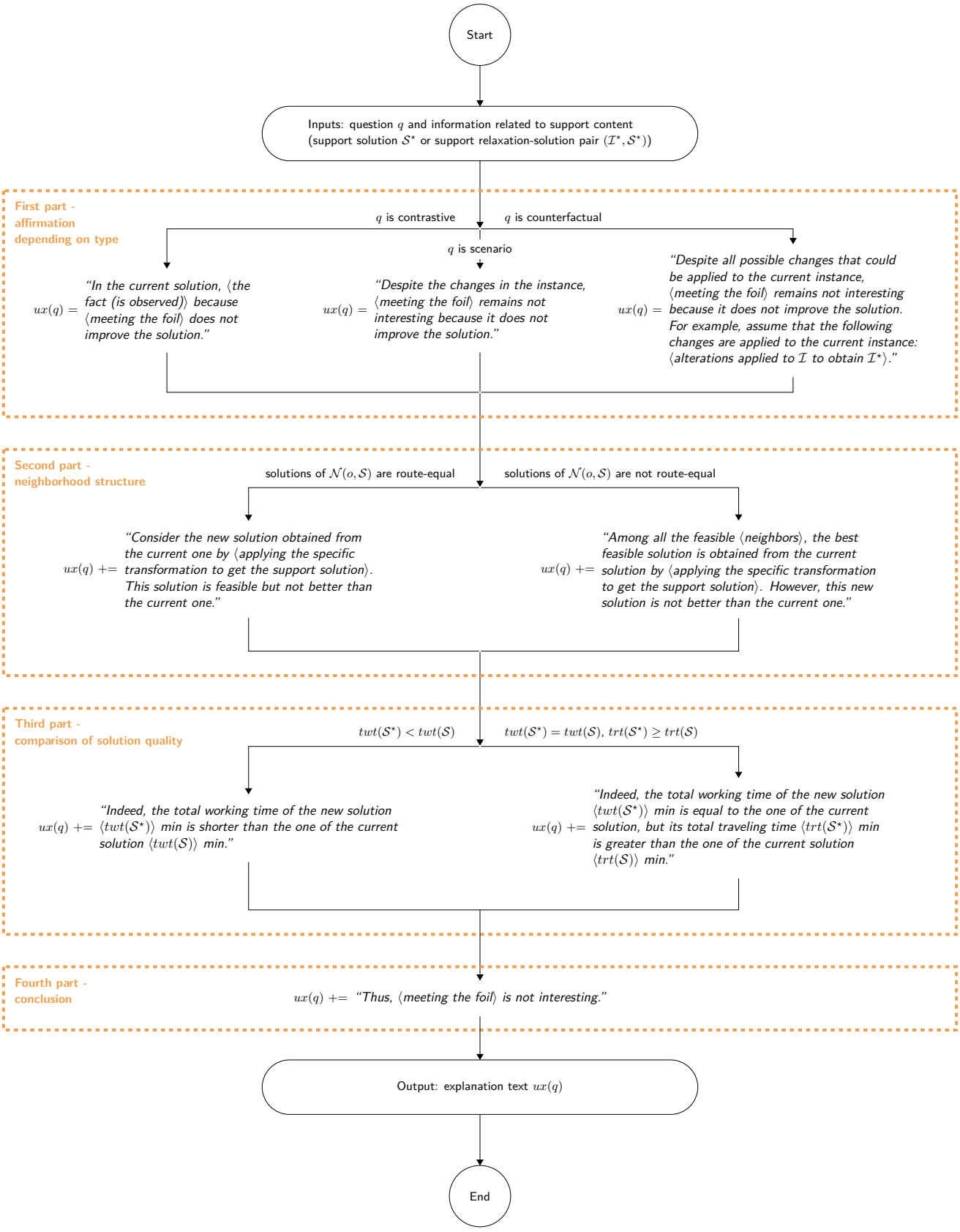


Figure 5.10: Building a negative explanation text about non-improvement given any type of question.

Chapter 6 Designing and implementing a system for presenting explanations to end-users

6.1 Introduction

As described in Chapter 3, the aim of this work is to develop an approach for explaining solutions of an optimization problem to the end-users of a system solving this problem. More precisely, this work focuses on a use case of *Workforce Scheduling and Routing Problem* (*WSRP*) that we specified and modeled in Section 3.2.

In Chapter 4, we presented our approach for modeling explanations about solutions of WSRP instances. It is summarized in Figure 4.2. It considers that, from *observations* about a solution, end-users define *questions* in order to request *explanations*. Three different *types of explanations*, introduced in Section 2.3, were modeled in this chapter: *contrastive explanations* aiming at clarifying why one fact occurred in the solution in contrast to another; *scenario explanations* describing how changes in the input instance parameters affect the fact observed in the output solution; and *counterfactual explanations* which aims at identifying how parameters could be changed in the input instance to obtain a user-defined output solution. Table 4.1 in Subsection 4.2.1 describe all the possible observations that end-users can make about their solution. They are given as *template texts* with fields to fill with values related to the instance and the solution. Then, Table 4.2 in Subsection 4.2.2 shows how each observation can be changed into three forms of questions, "why", "what if" and "how to", also formulated as template texts. Each one of these forms is associated with a different explanation type, respectively contrastive, scenario and counterfactual. Chapter 4 then described the mathematical steps enabling us to find, for a given question asked by the end-users, the mathematical content necessary to answer it. This mathematical content is however not intelligible for the end-users. Given such questions, and after some mathematical steps, end-users theoretically end-up obtaining intelligible *explanation texts*.

We thus detailed in Chapter 5 our method for generating such explanation texts given end-users' questions as inputs. First, regardless of the type of explanation, our method consists essentially in identifying a solution that we call *support solution*. Since questions are based on observations and since observations are assumed to be related to solution transformations which define solution neighborhoods (see Subsection 4.2.1), any end-user question can be related to a solution *neighborhood*. If this neighborhood contains feasible solutions, the support solution is defined as the *best feasible neighboring solution*; if the neighborhood does not contain any feasible solution, the support solution is defined as the neighboring solution that is the *nearest to be feasible* (see Subsection 5.3.2). The way the support solution is computed depends on the type of explanations. Then, given an end-user question, once the support solution has been identified, by analyzing its feasibility and its objective value, we create explanation texts answering the question. More precisely, we use pieces of template texts, whose contents depend on the analysis of the support solutions and the type of explanations, and combine them to build explanation texts (see *e.g.* Figures 5.2 and 5.3).

Now that we developed generation techniques for each type of explanations, we propose to incorporate and structure these techniques into a system which we refer to as an *explanation system*. This system serves two primary purposes: *i)* providing end-users with explanations of various types and on various observations about the solution they have; *ii)* allowing end-users to save a feasible support solution identified during the explanation process, along with its corresponding instance, for potential future use of this support solution as a new starting point for requesting explanations. We recall that in the case of scenario or counterfactual explanations, since the feasibility of a support solution is analyzed with respect to a support relaxation (*i.e.* an instance obtained by altering parameters of the original instance so as to increase its set of feasible solution, see Subsection 4.2.2), both relaxed instance and support solution must be saved. In order to fulfill this second purpose, the explanation system is equipped with a *history*, *i.e.* a set of instances, which are all relaxations of the same instance, such that each instance is mapped with one or several feasible solutions.

The explanation system operates as follows. End-users provide the explanation system with an instance and a feasible solution that they obtained thanks a solving system applied to this instance. This pair of instance and solution initializes the history, and becomes the pair of *current* instance and solution. Then, the system enables end-users to ask questions about this current solution in order to obtain explanations of different types in return. Whenever the support solution on which is based an explanation is feasible, the system enables end-users to add this support solution, along with its instance (either the current instance or a support relaxation) to the history. After having expanded the history, end-users can decide to keep asking questions about the same current solution or to switch to another pair of instance and solution from the history for requesting explanations about them. Thus, this iterative process enables end-users not only to obtain explanations about the initial solution that they got from a solving system, but also to gradually explore the set of feasible solutions (as well as the set of relaxations) and collect explanations about all of these solutions.

Finally, this chapter also presents a *Graphic User Interface* (*GUI*) prototype integrating the explanation system described

above. This GUI can be seen as a proof of concept of all the content developed in this work.

The remainder of this chapter is organized as follows. In Section 6.2, we detail the working principle of the explanation system and illustrate its use on an example. In Section 6.3, we present the GUI prototype of integrating the explanation system by describing its main view and functionalities.

6.2 Explanation system

6.2.1 Design of the explanation system

Figure 6.1 describes the working principle of the explanation system. Before providing a detailed presentation of this working principle, we provide some general remarks. First, we note that the system handles contrastive, scenario and counterfactual questions: each question type is associated with a series of steps surrounded by a yellow dashed box. Second, end-users may repetitively ask requests for explanation of the same type. However, scenario and counterfactual explanations are accessible only after obtaining contrastive explanations and if their support solutions are infeasible. The reason for this design will be explained. We present below the working principle of the explanation system

Working principle of the explanation system. We detail, step by step, the working principle of the explanation system depicted in Figure 6.1.

(1) End-users initialize the system by providing an instance and a feasible solution that they obtained thanks a solving system applied to this instance.

(2) A *history*, *i.e.* a set of instances such that each instance is mapped with one or several feasible solutions, is initialized with the given input pair of instance and solution.

(3) End-users choose within the history an instance $\mathcal{I}$, that we call *current instance*, as well as a feasible solution $\mathcal{S}$ of $\mathcal{I}$, that we call *current solution*.

(4) End-users tell whether they want to request a contrastive explanation about $\mathcal{S}$.
- If they do not want to, the system ends.
- Otherwise, the system begins a series of steps related to contrastive explanations, starting with step (C1).

(C1) End-users make an observation o about $\mathcal{S}$, using one of the observation templates of Table 4.1, and define a contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$.

(C2) An explanation text $ux(q)$ is computed using the method described in Section 5.3. It is based on a support solution $\mathcal{S}_c^\star$. This text $ux(q)$ and the support solution $\mathcal{S}_c^\star$ are presented to end-users.

(C3) If $\mathcal{S}_c^\star$ is feasible with respect to $\mathcal{I}$, then the system goes to step (C4).
If $\mathcal{S}_c^\star$ is not, then the system goes to step (5).

(C4) Since $\mathcal{S}_c^\star$ is a feasible solution of $\mathcal{I}$, end-users can add $\mathcal{S}_c^\star$ to the history, as a feasible solution associated with $\mathcal{I}$. Regardless of the choice made by end-users, the system goes back to step (3).

(5) End-users tell whether they want to request a scenario or a counterfactual explanation about $\mathcal{S}$.
- If end-users want to request a scenario explanation about $\mathcal{S}$, the system begins a series of steps related to scenario explanations, starting with step (S1).

(S1) End-users specify a relaxation $\mathcal{I}'$ of $\mathcal{I}$ so as to define a scenario question $q' = q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$.
Note that the observation o remains the same as the one used for the contrastive question at step (C1).

(S2) An explanation text $ux(q')$ is computed using the method described in Section 5.4. It is based on a support solution $\mathcal{S}_s^\star$. This text $ux(q')$ and the support solution $\mathcal{S}_s^\star$ are presented to end-users.

(S3) If $\mathcal{S}_s^\star$ is feasible with respect to $\mathcal{I}'$, then the system goes to step (S4).
If $\mathcal{S}_s^\star$ is not, then the system goes back to step (5).

(S4) Since $\mathcal{S}_s^\star$ is a feasible solution of $\mathcal{I}'$, end-users can add $\mathcal{I}'$ along with $\mathcal{S}_s^\star$ to the history. Regardless of the choice made by end-users, the system goes back to step (5).

- If end-users want to request a counterfactual explanation about $\mathcal{S}$, the system begins a series of steps related to counterfactual explanations, starting with step (H1).

(H1) End-users specify a set of relaxations $\mathcal{J}'$ of $\mathcal{I}$ so as to define a counterfactual question $q' = q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$.

(H2) An explanation text $ux(q')$ is computed using the method described in Section 5.5. It is based on a support instance $\mathcal{I}^\star$ and a support solution $\mathcal{S}_h^\star$. $ux(q')$, $\mathcal{I}^\star$ and $\mathcal{S}_h^\star$, are presented to end-users.

(H3) If $\mathcal{S}_h^\star$ is feasible with respect to $\mathcal{I}^\star$, then the system goes to step (H4).
If $\mathcal{S}_h^\star$ is not, then the system goes back to step (5).

(H4) End-users can decide whether to save in the history $\mathcal{I}'$ along with $\mathcal{S}_h^\star$, as $\mathcal{S}_h^\star$ is a feasible solution of $\mathcal{I}'$. Regardless of the choice made by end-users, the system goes back to step (5).

- If they do not want to request any scenario or a counterfactual explanation, the system goes back to step (3).

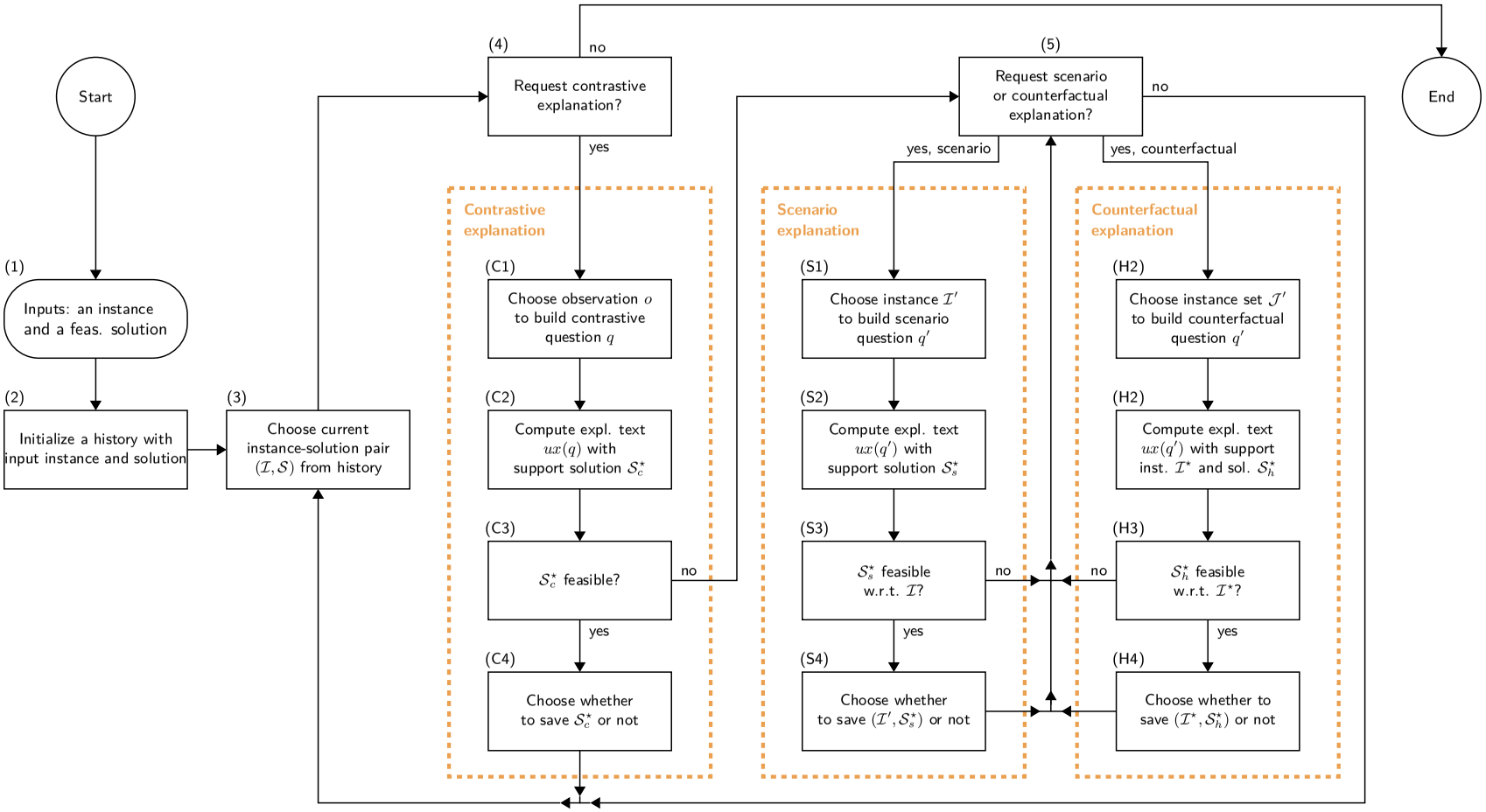


Figure 6.1: General working principle of the explanation system.

Two comments can be made about this system.

First, regardless of the explanation type (contrastive, scenario and counterfactual), the system allows end-users (at steps (C4), (S4) and (H4)) to add to the history support solutions ($\mathcal{S}_c^\star$, $\mathcal{S}_s^\star$ and $\mathcal{S}_h^\star$) if these solutions are feasible with respect to their corresponding instance ($\mathcal{I}$, $\mathcal{I}'$ and $\mathcal{I}^\star$). Thus, it is not required that support solutions are better than the current solution $\mathcal{S}$ to allow end-users to add them to the history. The reason for this choice is that, in our view, such an explanation system may be used by end-users not only to obtain explanations about their solutions, but also to explore the set of solutions that are feasible with respect to their current instance $\mathcal{I}$ (and even explore the set of relaxed instances by altering some data of $\mathcal{I}$ through scenario or counterfactual explanations). In other words, end-users may wish to save other feasible solutions than $\mathcal{S}$, even if they are not as good as $\mathcal{S}$. They may want to navigate within the set of feasible solutions, from solution to solution, to make sure that there is not a better solution than the first solution they use for initializing the history, which they obtain from an optimization system. Besides, end-users may have other criteria than the bi-objective function values to assess the quality of solutions in practice, so that they may prefer a solution with slightly worst bi-objective function values but better performances on these other criteria.

Second, as the system is designed, end-users cannot request scenario or counterfactual explanations directly after picking a current instance and a current solution from the history. They can ask for an explanation answering a scenario question $q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$ or a counterfactual question $q_h(o, \mathcal{S}, \mathcal{I}, \mathcal{J}')$, only after requesting an explanation answering contrastive question $q_c(o, \mathcal{S}, \mathcal{I})$ (with the same observation o for the three questions), and only if this contrastive explanation is negative with no feasible support solutions. We propose such a chaining of explanation types for the following reason. Suppose that, given the contrastive question $q = q_c(o, \mathcal{S}, \mathcal{I})$, there exists $\mathcal{S}' \in \mathcal{N}(o, \mathcal{S})$ that is feasible with respect to $\mathcal{I}$ *i.e.* the explanation $x(q)$ is positive or negative due to non-improvement. Then, $\mathcal{S}'$ is also feasible with respect to any relaxation $\mathcal{I}'$ of $\mathcal{I}$. Therefore, the explanation to $q_s(o, \mathcal{S}, \mathcal{I}, \mathcal{I}')$ is positive or negative due to non-improvement. However, this result has actually nothing to do with the fact that $\mathcal{I}$ has been relaxed into $\mathcal{I}'$, since $\mathcal{S}'$ was already feasible with respect to $\mathcal{I}$. And it would not make sense to write an explanation expressing that the relaxation of $\mathcal{I}$ has helped obtaining $\mathcal{S}'$. Thus, in our opinion, it is relevant to allow end-users to request scenario question only if this contrastive explanation is negative with no feasible support solutions. A similar reasoning can be made for counterfactual explanations.

Now that the working principle of the explanation system has been presented, we propose in the following subsection to illustrate its use on an example.

6.2.2 Usage example of the explanation system

In this subsection, we present a usage example of the explanation system. We assume that end-users have provided the system with a history containing the instance associated with the solution depicted in Figure 3.1 in Subsection 3.2.1. In addition, we assume that they have selected this instance and this solution as current instance and solution.

Observing that task 27 is not performed by Ellen while her route passes through it, end-users first request a contrastive explanation by asking a contrastive question based on observation template (Ins,C):

- Why is Ellen not performing task 27 just after task 17?

The system gives them in return a negative contrastive explanation text as follows:

- In the current solution, Ellen is not performing task 27 just after task 17 because of time constraints. Consider the new solution obtained from the current one by inserting task 27 just after task 17 in Ellen's planning. This new solution is infeasible. Indeed, by performing all the tasks from task 26 to task 27 at the earliest possible time, Ellen can end task 27 at the earliest at 4:37PM. However, task 27 is not available after 3:00PM. Thus, having Ellen perform task 27 just after task 17 is impossible.

In addition, along with this text, the system could provide a visual representation of the negative contrastive explanation, like the one in Figure 6.2 The graph on the left depicts employees' routes and the Gantt chart on the right their schedules, as in Figure 3.1. But, the graph on the left also emphasizes the insertion of task 27 between task 17 and task 8 in Ellen's route with the use of dash lines, and the Gantt chart highlights the time-infeasibility of Ellen's planning in the support solution. In the Gantt chart, the bracket on the left of task 26, at 12:00PM, represents the lower bound lbt_{26} of the availability time-window of task 26. It suggests that task 26 cannot be started earlier (shifted to the left) due to lbt_{26}. In other words, task 26 is the backward critical activity of the insertion of task 27 after task 17 in Ellen's planning (see Figure 3.4 in Part 3.3.1.3). The bracket at 3:00PM represents the upper bound ubt_{27} of the availability time-window of task 27. The vertical dashed line at 4:37PM, on the right of task 27 emphasizes the time at which Ellen ends performing task 27. Along with the two arrows spanning from 4:37PM to 3:00PM, it suggests that task 27 needs to be started earlier so that Ellen could end it by ubt_{27}. But we can understand that it is not possible to apply such a time shift as the sequence of task performances and travels between task 26 and task 27 presents no time interruptions.

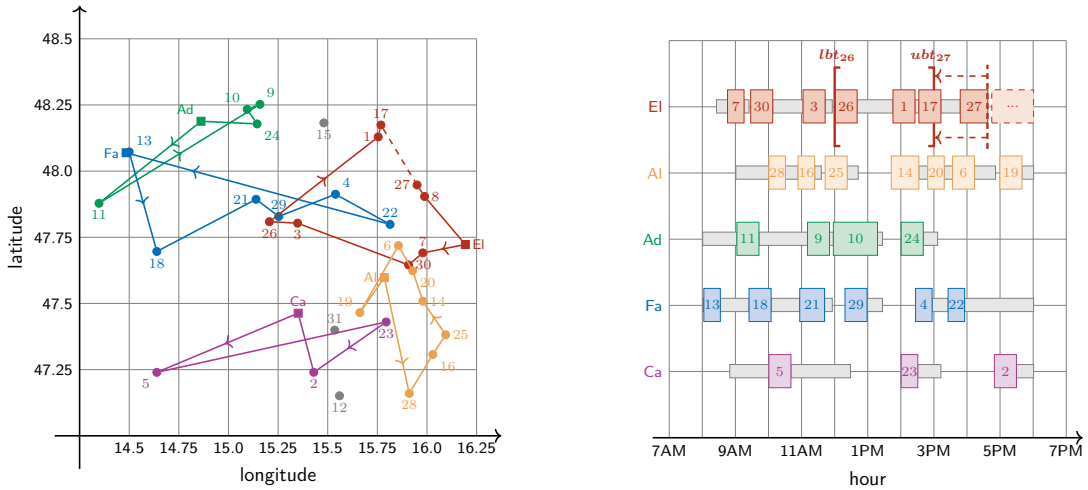


Figure 6.2: Representation of support solution involved in the negative contrastive explanation.

Thanks to the contrastive explanation text, end-users understand that, among other things, the upper-bound ubt_{27} of the availability time-window of task 27 causes time-infeasibility of the neighboring solutions. So, they may wonder if changing the value 3:00PM of ubt_{27} to 4:37PM would help getting a feasible support solution. To see if it is the case, they ask a scenario question (still based on observation template (Ins,C)), as follows:

- Ellen is not performing task 27 just after task 17, but what if task 27 is available until 4:37PM (instead of 3:00PM)?

Then, the system gives them in return a negative scenario explanation text as follows:

- Despite the change in the instance, having Ellen perform task 27 just after task 17 remains impossible due to time constraints. Consider the new solution obtained from the current by inserting task 27 just after task 17 in Ellen's planning. This new solution is infeasible. Indeed, by performing all the tasks from task 26 to task 8 at the earliest possible time, Ellen can end task 8 at the earliest at 5:24PM. However, task 8 is not available after 4:40PM. Thus, having Ellen perform task 27 just after task 17 remains impossible.

Again, along with this text, the system could provide a visual representation of the negative scenario explanation text, like the one in Figure 6.3 for instance. This representation is similar to the one of Figure 6.2. The difference is that, as expressed in the explanation text, the time-infeasibility of the support solution is now related to the availability of task 8, and no longer the one task 27. Therefore, in the Gantt chart on the right is represented a time-infeasibility related to ubt_8. In other words, task 8 is the forward critical activity of the insertion of task 27 after task 17 in Ellen's planning.

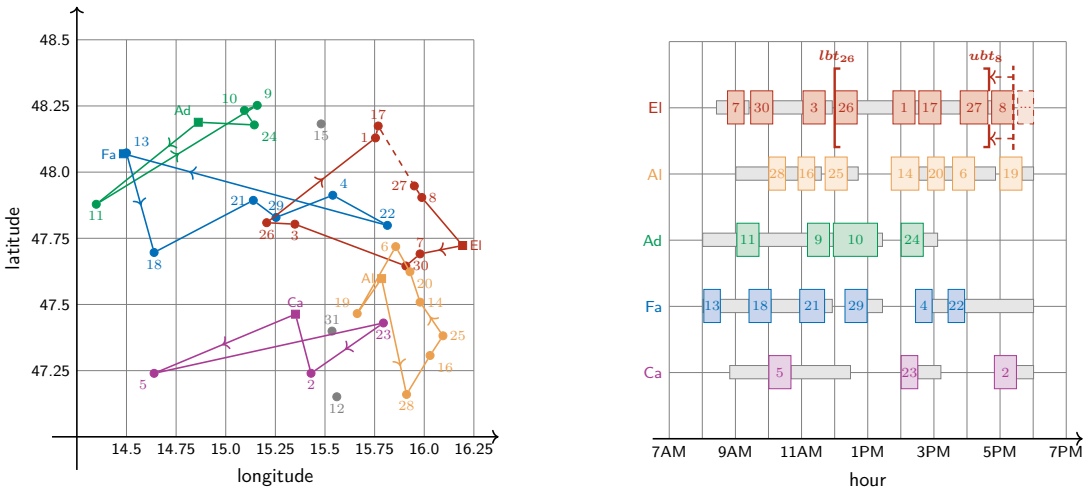


Figure 6.3: Representation of support solution involved in the negative scenario explanation.

Finally, since changing the upper-bound of the availability time-window of task 27 is not enough to get a feasible support solution, end-users may want to let the system find a way to alter the instance parameters to get one. So they request a counterfactual explanation, by asking a counterfactual question (still based on observation template (Ins,C)) in which they restrict alterations to be applied on availability time-windows of the tasks involved in Ellen's planning:

- How to make Ellen perform task 27 just after task 17, considering that at most two of the availability time-window bounds of all the tasks involved in Ellen's planning can be altered?

Then, the system gives them in return a positive counterfactual explanation text as follows:

- Assume that the following changes are applied to the current instance: task 27 is available until 4:37PM (instead of 3:00PM) and task 8 is available until 5:24PM (instead of 4:40PM). Then, having Ellen perform task 27 just after task 17 is possible and provides a better solution than the current one.

Along with this text, the system could provide a visual representation of the feasible support solution as in Figure 6.4.

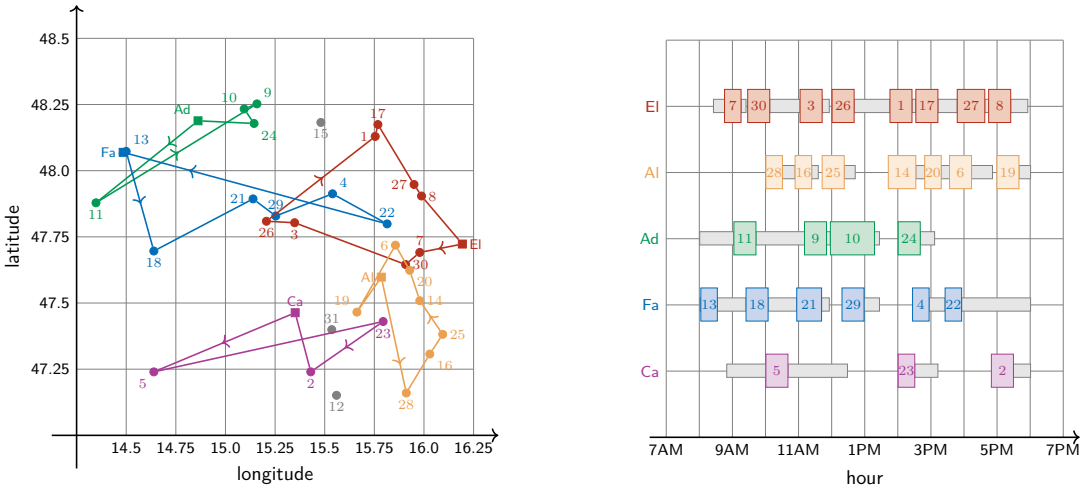


Figure 6.4: Representation of support solution involved in the positive counterfactual explanation.

In this section, we proposed a design of an explanation system allowing end-users to request and obtain contrastive, scenario and counterfactual explanations. We also presented a usage example showing how these three types of explanations can be consecutively involved within a series of interactions between end-users and the system. In the next section, we present an implementation of this system integrated within a Graphic User Interface.

6.3 Graphic User Interface (GUI) prototype

As a way to test all the concepts developed in this manuscript, we implement the explanation system described in the previous section and propose a *Graphic User Interface* (*GUI*) prototype which integrates this system so that end-users can graphically interact with it. In the following subsections, we describe the different functionalities of this GUI prototype, which include the ones related to the explanation system. We provide various annotated screen captures for illustrating these functionalities. For the screen captures, the GUI prototype is initialized with the same history as in the example of usage of Subsection 6.2.2 and the contrastive, scenario and counterfactual questions are also the same as in this example of usage.

6.3.1 Elementary views of the GUI

On start-up, the GUI opens on the "Home" view which is shown Figure 6.5. As all the other views of the GUI, this view is structured in four parts as follows.

- In the top part (1) is a header presenting the name of the system "XWSRP" and a short description of it.
- Just below the header (2) is a horizontal panel for exploring the history content. It consists in two drop-down lists:
  - one for selecting the current instance among the instances stored in the history;
  - one for selecting the current solution among the feasible solutions of the selected instance that are stored in the history.

  In the screen capture, the current instance and solution are respectively "instance_demo.1" and "solution_demo.1.1".

- On the left 3 is a navigation panel with six tabs which can be used to change of views:
  - the “Home” tab which directs to precisely the home view;
  - the “Instance description” tab whose view allows to look at the data of the current instance;
  - the “Instances comparison” tab whose view allows to compare the data of two different instances;
  - the “Solution description” tab whose view allows to look at the data of the current solution;
  - the “Solutions comparison” tab whose view allows to compare the data of two different solutions;
  - the “Solution explanation” tab whose view allows to question the current solution.

  The tab corresponding to the current view has a darker background, a we can see it on the screen capture with the “Home” tab.
- On the right of the navigation panel 4 is the main content of the view. In the case of the “Home” view, this main content is simply a text describing the overall purpose of the explanation system and the specific purposes of the different views of the GUI that can be accessed by clicking on the tabs of the navigation panel. The main content is essentially the part that changes between the different views.

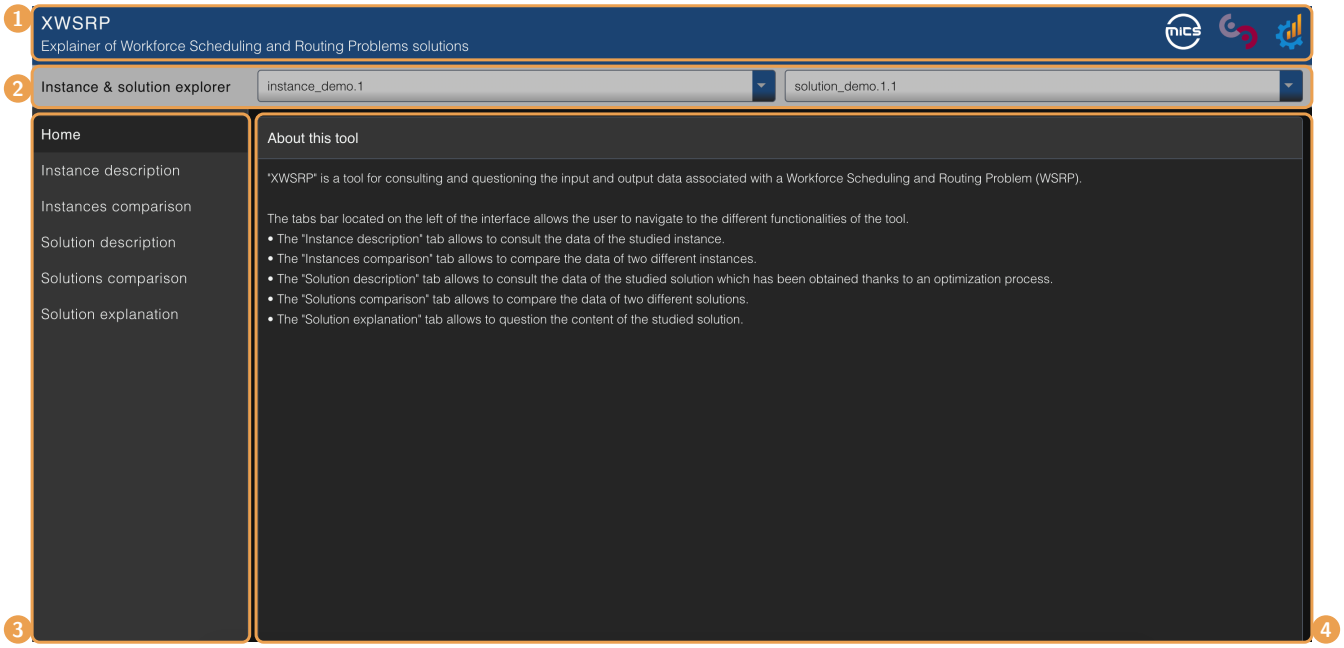


Figure 6.5: GUI “Home” view.

We present below the two views that display the data of the current instance and the current solution.

First, Figure 6.6 shows the “Instance description” view which can be used for inspecting the data of the current instance. From left to right and top to bottom, the main content contains five parts:

- a table 1 presenting the data related to employees, namely their names, their skill levels and their working time-window bounds (*e.g.* “Ellen”, “2”, “08:00AM” and “06:00PM”);
- a table 2 presenting the data related to tasks, namely their ids, their skill levels, their availability time-window bounds and their duration (*e.g.* “T1”, “1”, “08:00AM” and “06:00PM”, and “40”);
- a map 3 showing a geographic area of the world with colored dots corresponding to the locations of the employees, each employee having their own color (*e.g.* Ellen’s color is a dark blue);
- a map 4 showing the same geographic area with dots corresponding to the locations of tasks;
- a panel 5 with some key figures about the instance, including the number of employees, the number of tasks, the total available employee working time and the total task duration.

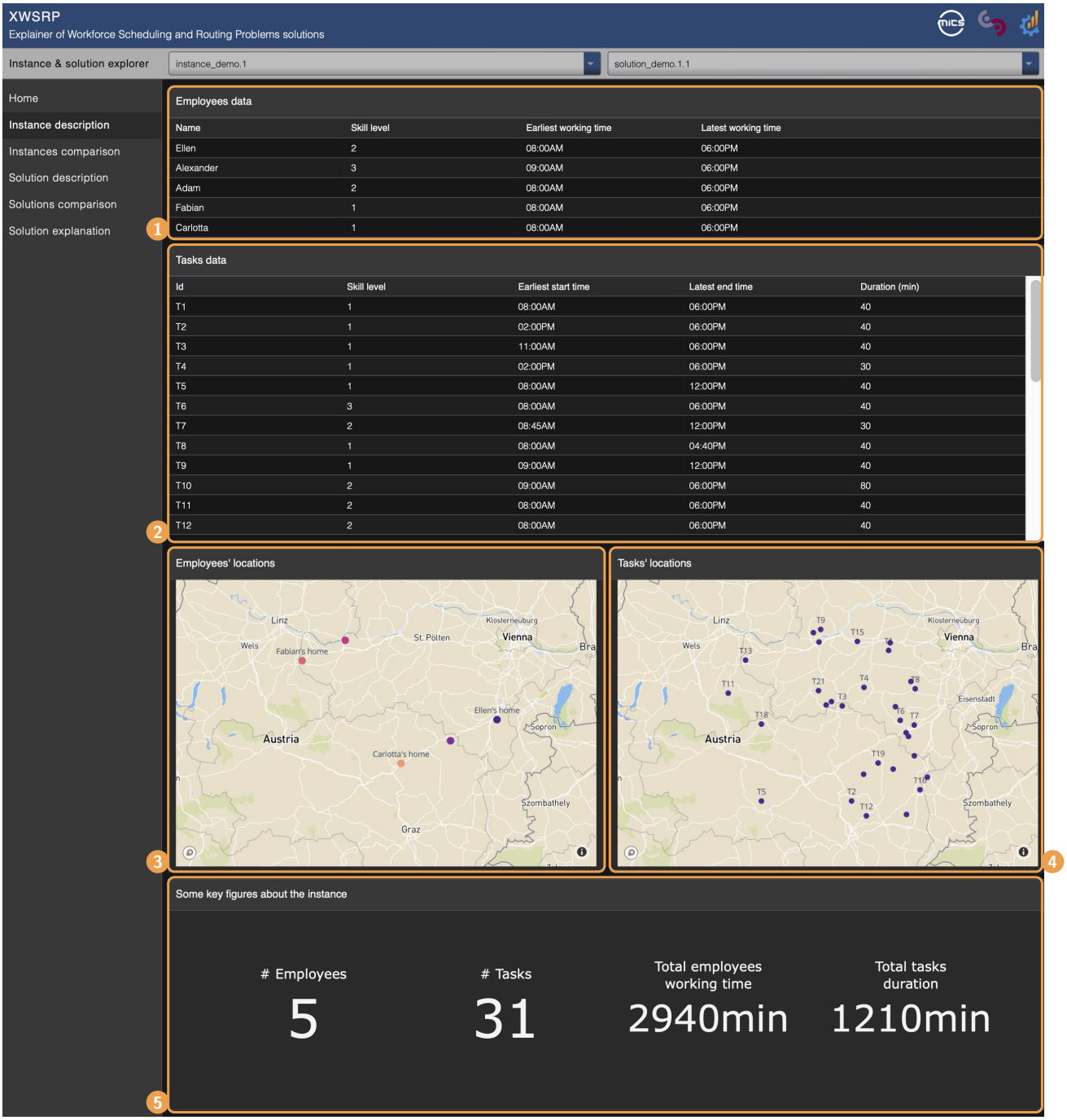


Figure 6.6: GUI "Instance description" view, which presents the data of the current instance.

Second, Figure 6.7 shows the "Solution description" view which can be used for inspecting the data of the current solution. Here, the main content contains three parts.

- In the top left part 1 is a map showing the employees' routes in the current solution, each employee being associated with a color used for drawing their route. It is based on the same geographic area as the maps showing employees' and tasks' locations in the "Instance description" view (see Figure 6.6). This map is actually the GUI equivalent of the representation of the routes in Figure 3.1. Besides, by hovering the mouse over some parts of the routes, information are displayed. By hovering it over a dot corresponding to a task, parameters of this task are shown (skill level, duration and availability time-window) so that one does not need to go back to the "Instance description" view to find this information.
- In the top right part 2 is a Gantt chart representing the employees' schedules in the current solution, each employee being associated with a color used for drawing their schedule which matches the color of their route in the map 1. This chart is the GUI equivalent of the representation of the schedules in Figure 3.1. Similarly to the routes, by hovering the mouse over some parts of the schedules, information are displayed. By hovering it over a colored rectangle corresponding to a task, information about the performance of this task are shown (including its start and end times). Over a gray rectangle corresponding to an employee travel between two tasks, information about the time needed for this travel is shown.
- At the bottom 3 is a panel with key figures about the current solution: the total working time and total traveling time, which are the two objectives that are optimized in line with the main model (see Model 1 in Subsection 3.2.2).

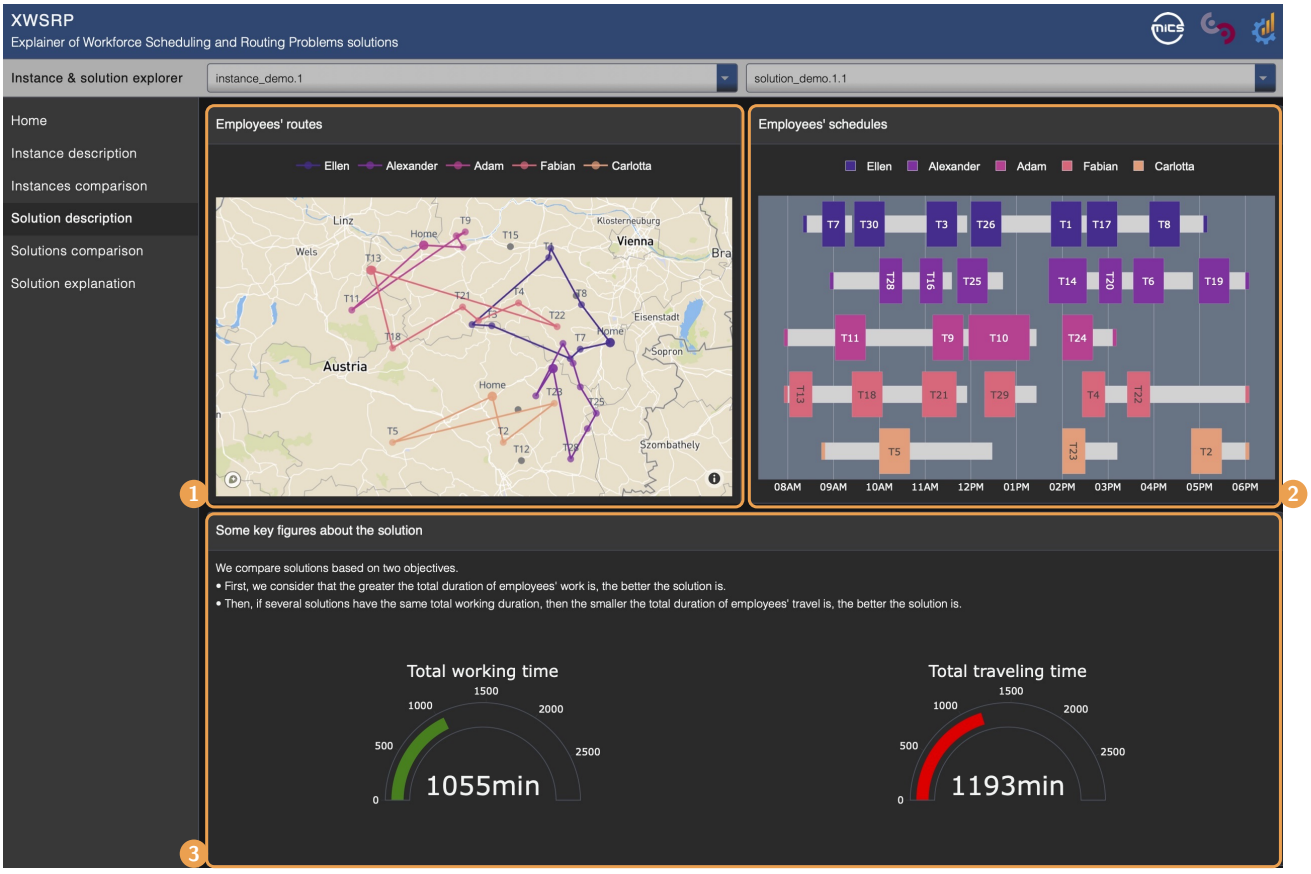


Figure 6.7: GUI "Solution description" view, which presents the data of the current solution.

Once end-users are familiar with the current instance and solution thanks to the views described in the previous sections, they may be interested in requesting explanations. This is the topic of the next subsection.

6.3.2 Requesting explanations about the current solution

In this subsection, we present how the "Solution explanation" view can be used in order to request explanations about the current solution.

As mentioned in Subsection 6.2.1, we consider that end-users start by requesting contrastive explanations. Therefore, the "Solution explanation" view first allows end-users to submit contrastive questions as shown in Figure 6.8.

- The top part of the main content (1) corresponds to employees' routes and schedules in the current solution. These two representations are the same as in the "Solution description" view. This allows end-users to see the solution while defining contrastive questions.
- Below (2) is the panel for defining a contrastive question and requesting an explanation in response. From left to right, top to bottom, it is composed of:
  - a drop-down list for selecting on which contrastive question template end-users want to base their question (*e.g.* "Why is employee _ not performing task _ just after activity _?");
  - one to three drop-down lists for selecting the values of the fields to fill in the question template (*e.g.* "Ellen", "T27" and "T17");
  - a text area for the question (*e.g.* "Why is employee Ellen not performing task T27 just after activity T17?");
  - a "Submit" button to request an explanation answering the defined question.

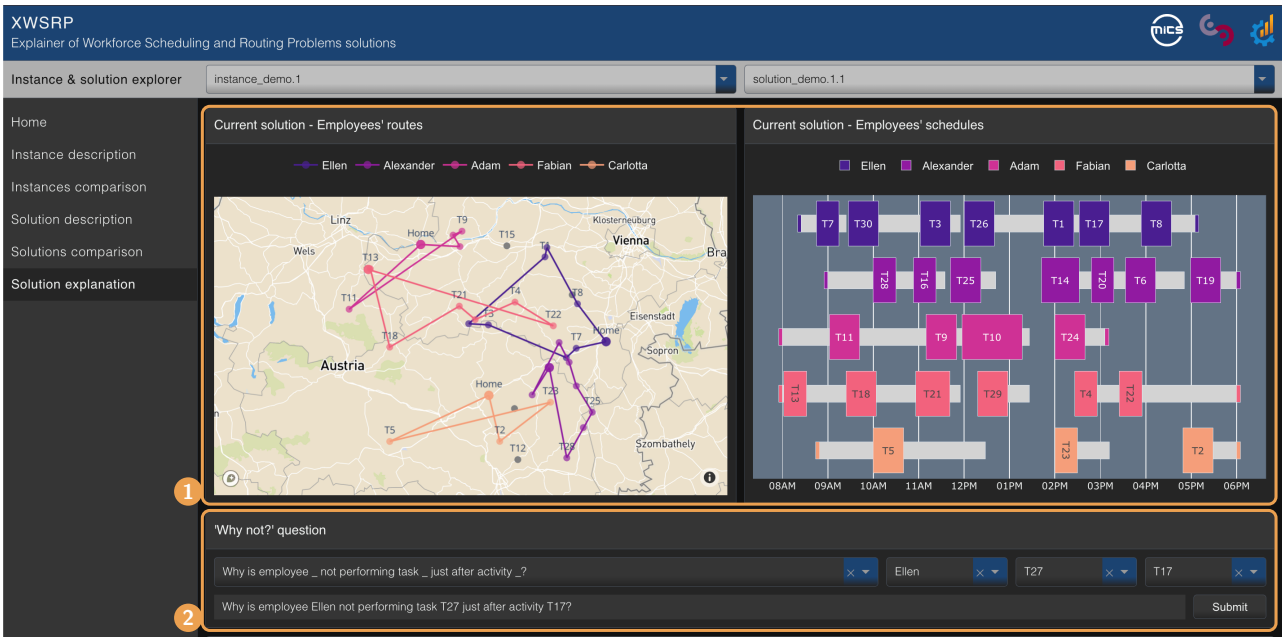


Figure 6.8: GUI "Solution explanation" view - part 1: defining contrastive questions.

After submitting a contrastive question, an explanation is generated according to the method detailed in Section 5.3. The view gets expanded below the panel related to the definition of the question in order to display the explanation as shown in Figure 6.9.

- The upper part (3) displays the routes and schedules in the support solution involved in the contrastive explanation. Regarding the map, as explanations focus on one employee, in order to put the emphasis on this employee, all the routes of the other employees are faded. Regarding the Gantt chart, if the support solution is feasible, then the chart is a regular Gantt chart; if it is infeasible relatively to time constraints, then the chart represents the time infeasibility in a similar way to what is depicted in Figure 6.2. For instance, in Figure 6.9, the support solution is time-infeasible due to the insertion of T27 in Ellen's schedule. Ellen's schedule is then represented on two rows, with T27 at the end of the first and T27', a copy of T27, at the beginning of the second. These two rows show two portions of Ellen's schedule which are respectively backward-feasible and forward-feasible. On the first row, a vertical red line on the left of T26 suggests that it is the backward-critical activity and shows why T27 cannot be shifted backward. On the second row, a vertical red line on the right of T27' indicates that it can not be shifted forward. By hovering the mouse on this line, we can read that this is due to the upper bound of its availability time-window. Therefore, the two schedule portions cannot be connected temporally.

- Below, is a horizontal panel 4 related to contrastive explanations. From left to right, top to bottom, it contains:
  - a text area displaying the contrastive explanation text ("In the current solution, ..."");
  - a "Return" button to go back to the definition of contrastive question;
  - a "Save" button for saving the support solution displayed in 3 that available only if it this solution is feasible.
  - a "What if?" button to start requesting scenario questions, which is available only if the support solution is infeasible;
  - a "How to?" button to start requesting counterfactual questions, which is also available only if the support solution is infeasible.

  Note that the "Save" button is darker than the other buttons. It means that the button is unavailable which makes sense since the support solution presented here is infeasible (relatively to the current instance).

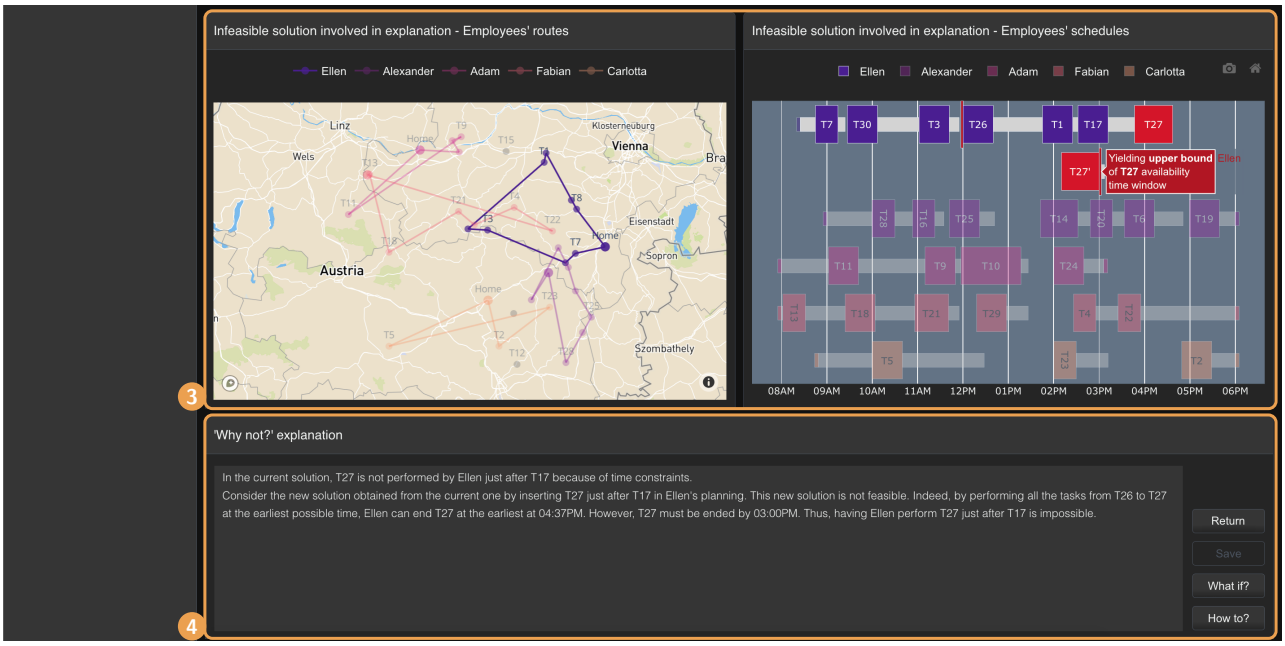


Figure 6.9: GUI "Solution explanation" view - part 2: presenting contrastive explanations.

Suppose that end-users decide to request scenario questions by clicking on "What if?". Then, the view expands again as shown in Figure 6.10 with three new panels.

- A first panel 5 can be used to edit the data about the employees.
- A second panel 6 can be used to edit the data about the tasks (*e.g.* changing the upper bound of the availability time-window of T27 from 03:00PM to 04:37PM as highlighted in blue).

  Thus, panels 5 and 6 allow end-users to define a relaxed instance of the current instance.
- A third panel 7 relates to the scenario question. It consists of:
  - a text area displaying the scenario question, which summarizes especially the changes applied to the current instance;
  - a "Submit" button to request a scenario explanation answering the question;
  - a "Reset" button to set any instance parameters back to its value in the current instance;
  - a "Return" button to come back to the contrastive explanation part (see Figure 6.9).

After submitting a scenario question, an explanation is generated according to the method detailed in Section 5.4. The view gets expanded again in order to display the explanation as shown in Figure 6.11. The structure of this scenario explanation part is similar to the one of contrastive explanation (see Figure 6.9): panels 8 and 9 play similar roles as panels 3 and 4. However, there are two buttons in 9, not four as in 4:

- a "Ok" button to to come back to the contrastive explanation part (see Figure 6.9);
- a "Save" button for saving the scenario support solution, displayed in 8, along with the relaxed instance; this button is available only if it this support solution is feasible.

Note that the "Save" button is again unavailable which makes sense since the (scenario) support solution presented here is infeasible (relatively to the relaxed instance).

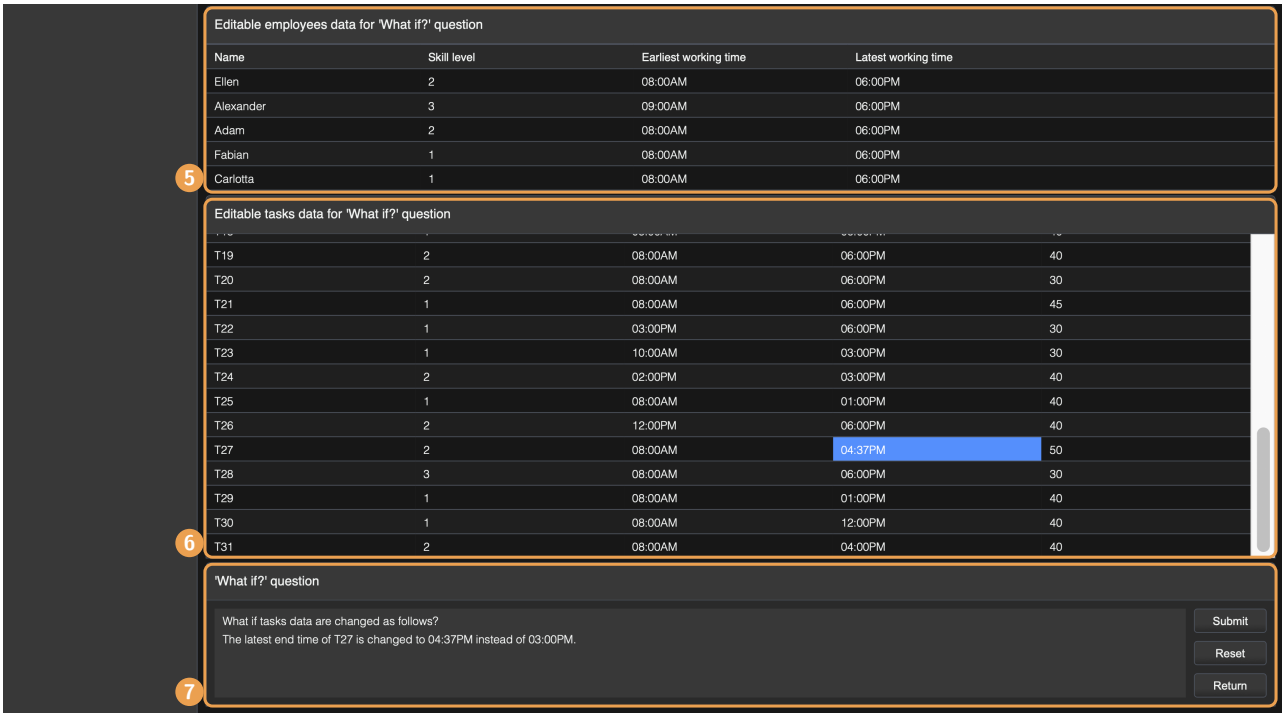


Figure 6.10: GUI "Solution explanation" view - part 3: defining scenario questions.

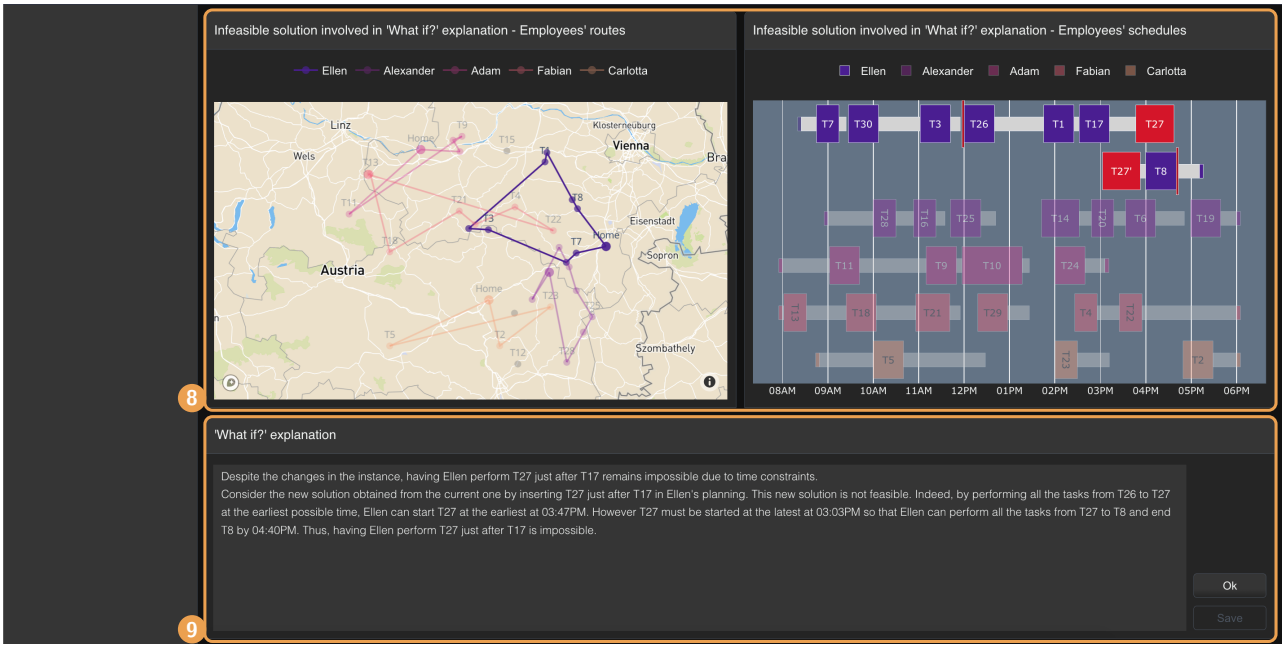


Figure 6.11: GUI "Solution explanation" view - part 4: presenting scenario explanations.

Suppose that end-users decide to come back to the contrastive explanation part shown in Figure 6.9 by clicking on "Ok". Suppose in addition that end-users decide this time to request counterfactual questions by clicking on "How to?". In the current state of the GUI prototype, there is no panels allowing end-users to define what instance parameters they allow to alter and by how much before asking their counterfactual questions. By default, the allowed alterations are in line with the assumptions presented in Subsection 5.5.1: alterations are exclusively allowed on task availability time-window bounds and task durations, and are limited to with "reasonable bounds".

As a consequence, after clicking on "How to?", the view expands as shown in Figure 6.12 with two panels related to the counterfactual explanation. The structure of this counterfactual explanation part is similar to the one of scenario explanation (see Figure 6.11): panels 10 and 11 play similar roles as panels 8 and 9. Especially, the two buttons "Ok" and "Save" in 11 play similar roles as the one in panel 9. Note that the "Save" button is here available which makes sense since the (counterfactual) support solution presented here is feasible (relatively to the relaxed instance).

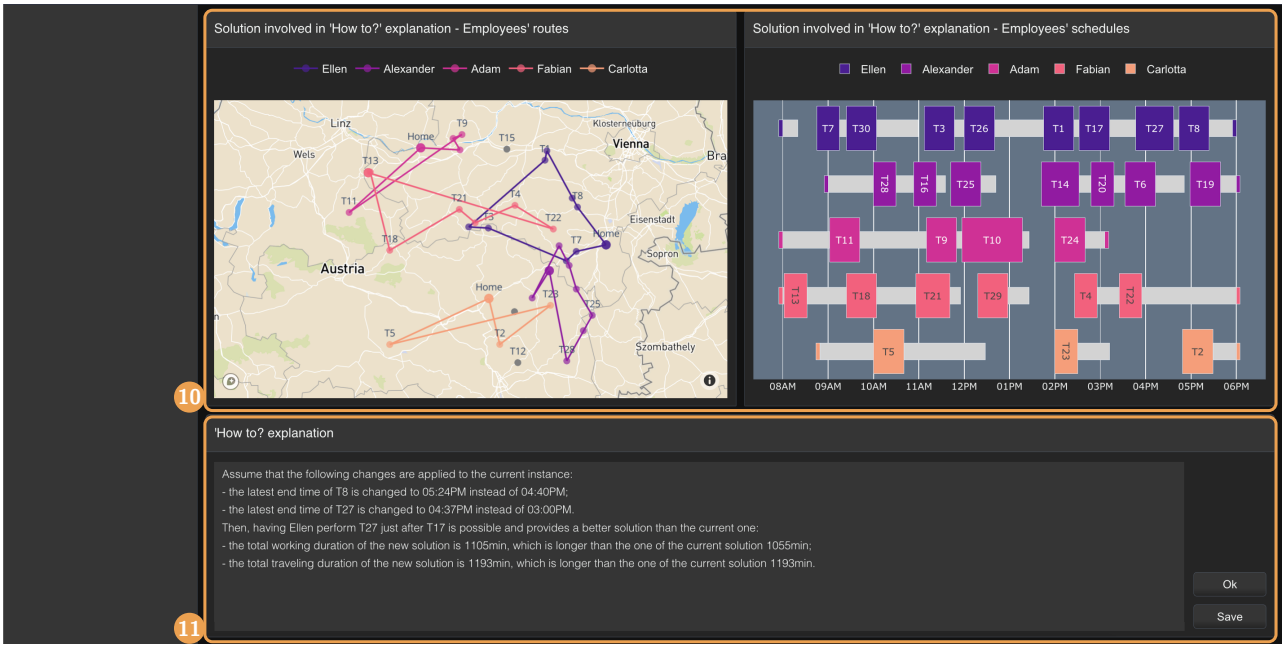


Figure 6.12: GUI "Solution explanation" view - part 5: presenting counterfactual explanations. In order to obtain the counterfactual explanation text displayed in the screen capture, the underlying counterfactual transformation model constraint has been constrained to alter at most two parameters among the task availability time-window bounds and the task durations.

6.3.3 Comparing instances and solutions in the history

While using the "Solution explanation" view, end-users may decide to save some of the feasible solutions presented within the explanations they obtain with the explanation system. In the case of scenario and counterfactual explanations, instances related to the saved solutions are also saved. After obtaining enough explanations about the current solution, end-users may be interested in comparing either instances or solutions they have saved. "Instances comparison" and "Solutions comparison" views are precisely designed for this purpose.

Figure 6.13 shows the "Instances comparison" view which allows to compare two instances stored in the history. The main content of this view is organized in two columns, each one corresponding to a different instance.

- The left column 1 is associated with the current instance, which is "instance_demo.1" in the screen capture.
- The right column 2 is associated with another instance, which is "instance_demo.2" in the screen capture. This other instance can be selected from the history thanks to a drop-down list 2a.

These two instances are compared: first the data related to the employees and then the ones related to the tasks. If the same parameter takes different values between the two instances, it is highlighted in blue, in the data tables of the other instance. For instance, in the screen capture, the upper bound of the availability time-window of task 8 is highlighted in blue because its value in "instance_demo.1" is 4:40PM while it is 5:24PM in "instance_demo.2".

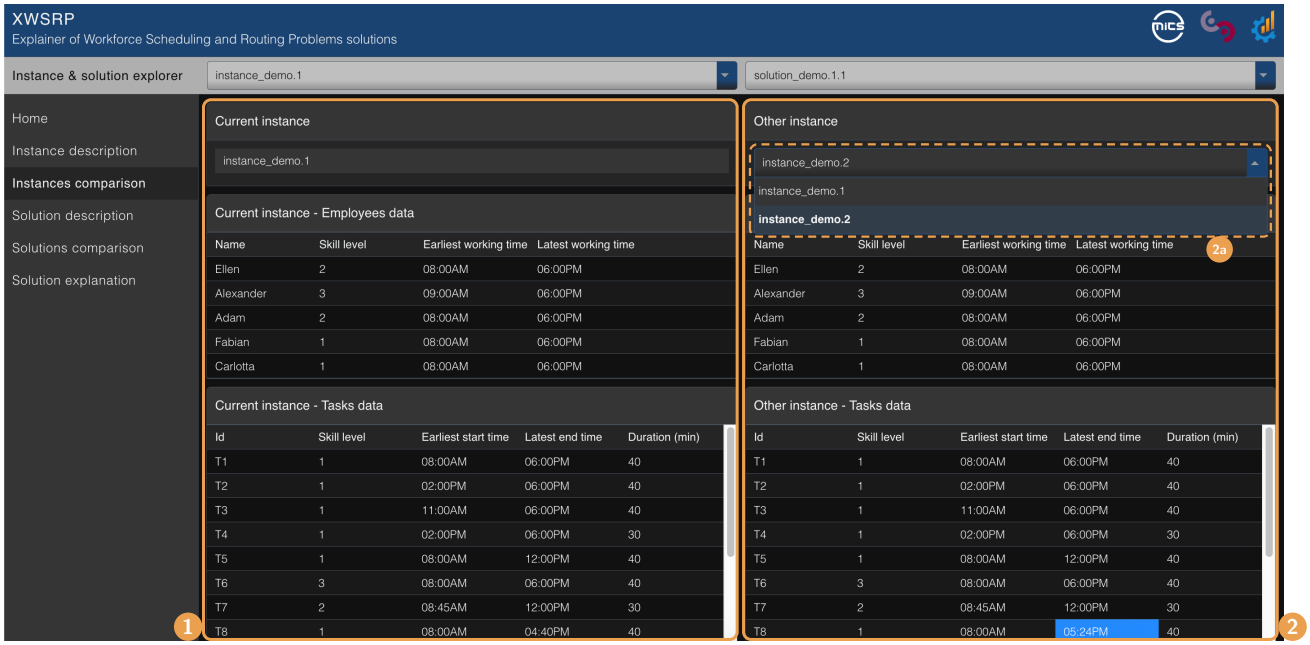


Figure 6.13: GUI "Instances comparison" view.

Figure 6.14 corresponds to the view for comparing two solutions of the history, both solutions being feasible with respect to the current instance. As in the view for comparing two instances, the main content is organized in two columns. Each column corresponds to a different solution.

- The left column (1) is associated with the current solution, which is "solution_demo.1.1" in the screen capture.
- The right column (2) is associated with another solution, which is "solution_demo.1.2" in the screen capture. This other solution can be selected from the history thanks to a drop-down list (2a).

Various aspects about the two solutions are presented in order to compare the two solutions, from top to bottom:

- the maps of employee routes (*e.g.* Ellen's route in "solution_demo.1.2" has an intersection that her route in "solution_demo.1.1" has not);
- the Gantt charts of the employee schedules (*e.g.* Ellen's schedule "solution_demo.1.2" contains all the tasks of her schedule "solution_demo.1.1", plus T27, and it is ordered in a different way);
- the values of the objective functions (*e.g.* the total working time and total traveling time of "solution_demo.1.2" are greater by respectively 50min and 28min than the ones of "solution_demo.1.1").

6.4 Conclusion

In this chapter, we described the design of an *explanation system i.e.* a system for structuring the interactions between on one hand end-users and on the ohter the methods for generating contrastive, scenario and counterfactual explanation texts, presented in Chapter 5. This system uses a *history i.e.* a set of instances, each instance being mapped with solutions that are feasible with respect to it. End-users provide the explanation system with an instance and a feasible solution that they obtained thanks a solving system applied to this instance. This instance-solution pair initializes the history. End-users can then request explanations about the solutions stored in this history. If the support solutions involved in the explanations they obtained from the system are feasible, they can decide to save these solutions, along with their corresponding instances, in the history and expand it. Thus, the explanation system can not only be employed by end-users to obtain explanations about solutions, but also to explore, step by step, the sets of relaxed instances and neighboring feasible solutions.

We also presented a *Graphical User Interface (GUI)* integrating this abstract explanation system. Such a GUI helps using the system in a user-friendly manner. It contains various *views*. Among others, one view consists essentially in interfacing the explanation system by allowing end-users to request explanations of various types. Two other views allow end-users to compare the parameters of two different instances stored in the history and to compare the parameters of two solutions that are feasible with respect to the same instance also stored in the history.

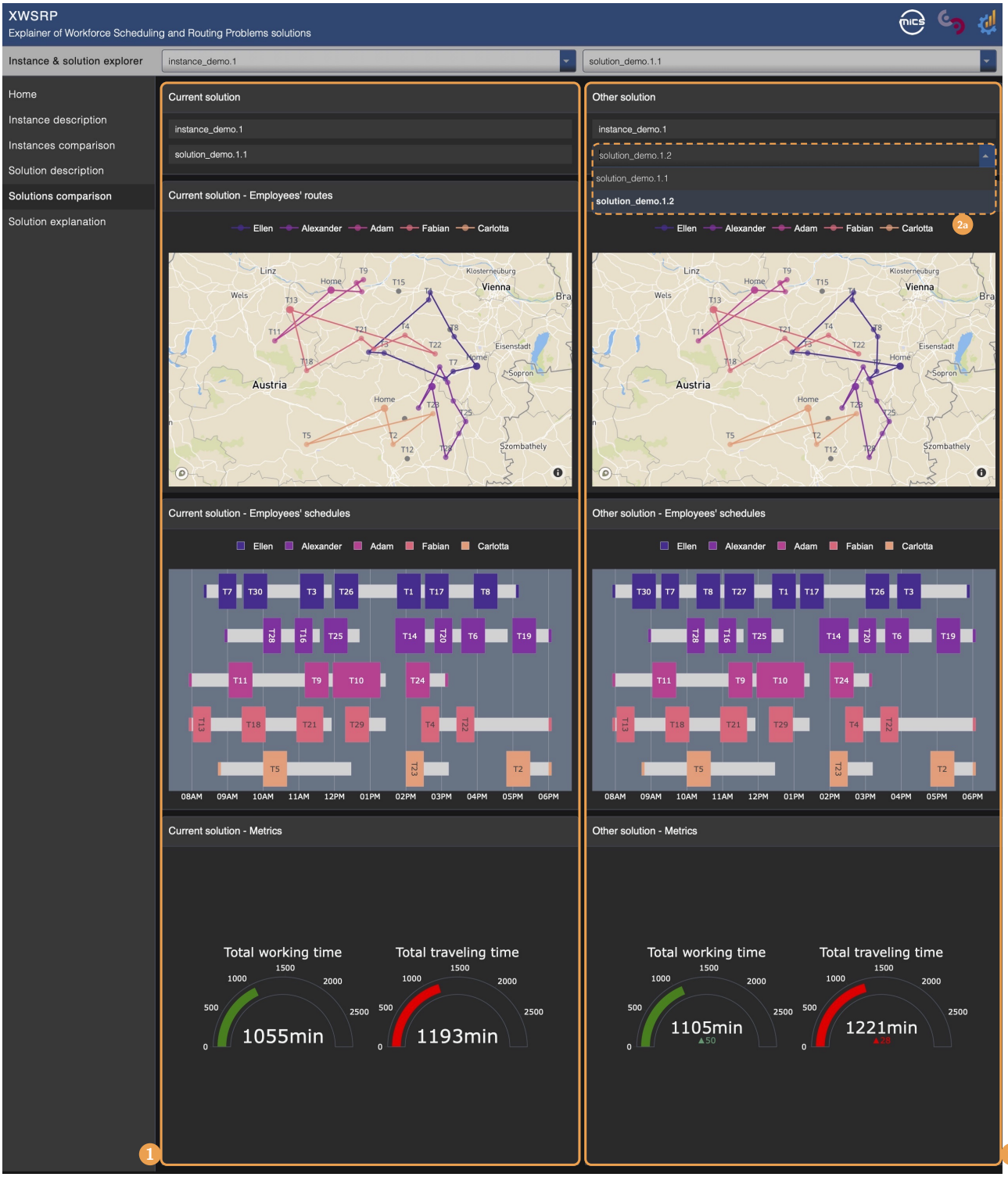


Figure 6.14: GUI "Solutions comparison" view.

Chapter 7 Conclusion and perspectives

7.1 Conclusion

This thesis focused on the development of an original framework to provide explanations in relation to a Combinatorial Optimization (CO) problem: the Workforce Scheduling and Routing Problem (WSRP). Given a set of mobile employees and a set of geographically dispersed tasks, this problem consists in building and assigning to each employee a planning, *i.e.* a route-schedule pair which defines the tasks that they should perform, in what order and at what times, over a certain horizon. The objective is to design a family of plannings accommodating as much work as possible, with minimum traveling cost, while satisfying a set of constraints.

Chapter 2 laid the groundwork of this thesis by emphasizing the interdisciplinary nature of the research related to explanations. We explored how explanations are a research topic not only in computer science but first and foremost in social sciences. Our exploration revealed the link between explanations and conversations, highlighting the importance that we, explanation designers, adhere to the cooperative principle by following the maxims of quality, quantity, relation, and manner when formulating the content of our explanations. Moreover, we noticed the inherent selectivity and the prevalent contrastive nature of explanations. This primary understanding of explanation from the perspective of social sciences served us as a basis for our subsequent work. Transitioning our review into the general field of artificial intelligence (AI), we inspected key characteristics of explanation methods in AI, which allowed us to position our work relative to this vast literature: the target audience of our explanations consists of the end-users of an optimization system solving our WSRP use case; our explanations adopt a local scope by focusing on a specific solution of a WSRP instance; they can be of three different types namely contrastive, scenario, or counterfactual; they are conveyed in the form of texts, constructed using templates; and they are triggered by questions formulated thanks to other template texts We also identified the need for methods to explain solutions within the domain of Operations Research (OR) and more specifically within CO.

Chapter 3 specified the characteristics of the WSRP use case for which we developed our explanation framework. We gave formal definitions and modeled this use case as a bi-objective Integer Linear Programming (ILP) problem. We also introduced the concept of solution transformations, along with their corresponding solution neighborhoods. We listed various families of solution transformations including transformations about inserting a task in an employee planning, exchanging a task in an employee planning with another task or reordering tasks within an employee planning. We classified transformations into three kinds depending on the structure of their corresponding neighborhoods, namely constant-size, polynomial-size and exponential-size transformations. Solution transformations and neighborhoods together play a crucial role in our endeavor to model, compute and generate explanations.

In Chapter 4, we turned our attention to the modeling of explanations. We detailed our original framework guiding end-user observations about WSRP solutions to questions and, ultimately, to explanations. Observations are assumed to be inherently linked to solution transformations and neighborhoods. The three types of explanations, contrastive, scenario and counterfactual, are handled by our framework with three corresponding types of questions, "why not", "what if" and "how to". Questions are interpreted in CO terms, possibly involving instance relaxations, and induce mathematical programs bringing into play solution neighborhoods. Explanations are then mathematically defined based on the feasibility of the induced programs. We ended this chapter by acknowledging the challenges posed by the mathematical nature of the defined explanations, which may not be intelligible for individuals who are not experts in combinatorial optimization, and the need to translate them into texts. So that the practical computation and narration of explanations remained a key step, setting the stage for Chapter 5.

Chapter 5 described our approach to generating explanation texts in response to end-user questions. Regardless of the type of the question given as input, our method comprises two main stages: identifying relevant mathematical content, so-called support content, and constructing explanation texts using templates filled with information extracted from this content. We detailed the algorithms employed to compute the support content, with an emphasis on efficiency. While numerical studies demonstrated the feasibility of our approach, we acknowledged potential computational challenges when utilizing ILP-based methods.

The achievement of our research is presented in Chapter 6 where we introduced a comprehensive explanation system designed to facilitate interactions between end-users and our explanation generation methods. This system leverages a history of instances and their associated solutions to not only provide end-users with explanations but also enable them to explore incrementally the spaces of neighboring feasible solutions and relaxed instances. A Graphical User Interface (GUI) integrates this system enabling end-users not only to request explanations but also compare different instances and solutions.

7.2 Perspectives

The work presented in this manuscript raises several issues and research perspectives. We describe in what follows some research directions that may be worth investigating.

- **Adapting our approach to other WSRP definitions.** In Chapter 3, we specified the characteristics of our WSRP use case, including the instance data, the lexicographic bi-objective function to optimize and the constraints to satisfy. However, the definition of the WSRP is not unique. Then, a first interesting research perspective to explore could be to investigate whether our methods for both modeling and generating explanations remain valid when applied to other variants of the WSRP. If they do not, it would be valuable to determine how they can be modified to accommodate these other definitions.
  In collaboration with Dr. Martin Aleksandrov, we actually started to explore this research perspective during an international research visit in June-July 2023, at Freie Universität, Berlin. The purpose of this visit was to study the integration of social considerations into our work on explanation methods for CO problems. Specifically, given Dr. Martin Aleksandrov's leadership in a research project titled "Fairness and Efficiency for Emerging Vehicle Routing Problems," and our focus on the CO problem of WSRP, we aimed at incorporating fairness considerations into our WSRP use case and at anticipating the consequences on our explanation methods. Drawing inspiration from research on fair Vehicle Routing Problems (VRP) (see *e.g.* [MHV18] for a comprehensive review and [SV13] for an example of fair VRP), we essentially integrated fairness into our WSRP formulation by replacing the first objective seeking to maximize the total working time by another one seeking to maximize the minimum individual working time. We concluded that such a modification in our WSRP definition would not affect the core concepts involved in our framework modeling explanations. However, our methods for generating explanations would require some adaptations or extensions. For instance, it would be relevant to develop new preliminary checks, including assessments anticipating whether the solution transformation associated with the end-user question would lead to a deterioration in solution quality.
  Moreover, various other complicating features have been considered in the literature when modeling the WSRP. For example, [GM13] considers task precedence constraints, [BR08] workforce synchronization constraints and [CTH16] experience-based service times in the context of a multi-day horizon. Analyzing how to adapt our methods in order to generate explanation for these other variants of the problem, which involve different instance data, non-lexicographic multi-objective function and new constraints, represents an intriguing research perspective.

- **Transposing our approach to other CO problems.** An essential research perspective to consider would be to assess the level of genericity 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
  Indeed, firstly, our approach for modeling explanations hinges on solution neighborhoods, a concept rooted in Local Search. Many CO problems have been extensively studied within the local search literature, leading to the creation of specific solution neighborhoods for each of these problems. These neighborhoods may be leveraged to build end-user observations. Moreover, while the solutions neighborhoods used as part of heuristic solving algorithms are usually small enough to be explored, expanding them into larger neighborhoods may also be relevant in the context of explaining solutions. As demonstrated in this thesis focusing on a WSRP use case, the exponential-size neighborhoods expanding polynomial-size ones also align with natural end-user observations. Thus, expansions of neighborhoods found in local search literature may also be leveraged to build end-user observations, resulting in a large diversity of observations and then questions.
  Secondly, the conceptual framework developed in Chapter 4 which guides the progression from an end-user observation to a question and ultimately to an explanation remains relatively abstract, even though this thesis focuses on a WSRP use case. The key concepts involved in this framework, such as observation-based questions, decision-problem interpreted questions, foil models, and others, are independent from the WSRP and could be applied to other problems.
  However, since the algorithms developed for generating explanation texts rely on neighborhoods, which depend on the nature of the CO problem, we expect that research work would be needed in order to adapt our explanation-generation techniques to another CO problem.

- **Considering end-user questions that do not satisfy the assumptions underlying our approaches.** In Chapter 4, we introduced the assumptions on which we underpin our approach for modeling explanations in our CO context. Especially, we assume that end-user questions are based on observations which are inherently linked to solution transformations and neighborhoods. However, it is worth noting that not all interesting questions about WSRP

solutions align with these assumptions. For example, consider the question "Why is Ellen working more than Carlotta?" which is a legitimate question that a planner might ask. Its corresponding observation is "Ellen is working more than Carlotta". It is challenging to us to identify a transformation associated with this observation. In any case, the transformation linked to this question could potentially involve a very large number of elementary transformations, as it would require modifying the schedules of two or more employees and considering a substantial set, if not all, of the tasks. Consequently, applying it would demand a significant computational effort. Furthermore, we will have to identify the relevant explanatory information to provide in the explanation text provided to the end-users.

- **Evaluating the benefits of our approach for the end-users.** As outlined in Chapter 1, one of the key motivations behind this thesis is leveraging explanations as a way to hold the trust of decision-makers in the optimization-based systems they use and the solutions they obtained from them. Thus, in our opinion, a crucial research topic to study is evaluating the actual benefits of our explanations for these end-users.
  In the last phase of this thesis, we undertook an effort to investigate this matter. In [VL21], the authors identify two main ways to evaluate explanation methods: *i)* the "objective evaluations" which correspond to research studies employing objective metrics and automated approaches, and *ii)* the "human-centered evaluations" which involve end-users and exploit their feedback. Drawing inspiration from the field of Explainable Artificial Intelligence (XAI), exemplified by works such as [YJ95, KKM+21], we developed a human-centered evaluation with the objective of assessing whether having access to explanations about the solutions obtained from an optimization system would influence the trust that end-users have in these solutions.
  Practically, akin to the approach adopted by [KKM+21] in the context of AI Planning, we designed a questionnaire and harnessed our GUI, presented in Chapter 6, to conduct our evaluation. Our GUI was populated with three different instance-solution pairs - where the three solutions had been obtained by solving heuristically the three instances. Participants were tasked with adopting the role of planners, assuming the responsibility for gauging the relevance of WSRP solutions and for deciding the application of these solutions to define the plannings of their hypothetical mobile employee colleagues. Without knowing it, the participants were divided into three groups. All groups were given access to the GUI in order to inspect the solutions. Two groups also had the option to request explanations if desired. Within these two groups, one could receive only textual explanations while the other was provided with both textual explanations and accompanying graphics. Following each solution inspection, participants were asked to assess their confidence in the relevance of the presented solution. For the assessment, participants were asked to rate their confidence on a Likert scale, from 1 "very unconfident" to 5 "very confident", similarly to the approach in [YJ95].
  In order to test our survey, we formed a panel of participants made of 38 M.Sc. students. We imagined measuring significant variations between rates given by participants of the different groups: participants who were given access to explanations were expected to show a higher level of confidence than the ones who were not. However, the results of this survey did not yield conclusive findings: distribution of rates were overlapping. From these results, we deduced two main weaknesses in our evaluation approach. First, this alpha test survey lacked rigor: more relevant results could likely be obtained with a panel comprising actual users of optimization systems who could take the time to respond sincerely to the survey questions. Second, trust is a multifaceted sentiment influenced by knowledge, beliefs, emotions, and other aspects of experience. Therefore, attempting to measure it through a single Likert scale may prove futile. We could maybe draw inspirations from works such as [HMKL18] which propose measuring trust using multiple Likert scales covering different dimensions of trust.
  In a nutshell, there is still work to be done in the design and implementation of a relevant and effective approach for evaluating the benefits of our explanation methods for end-users.

- **Leveraging Natural Language Generation techniques to produce explanation texts.** In Chapter 5, we presented our method for generating explanation texts. After identifying relevant mathematical content, referred to as support content, explanations texts are built thanks to templates populated with information extracted from this support content. While we endeavored to streamline the process by minimizing the number of predefined template texts used for generating a diverse range of potential explanations, the preparatory work remains time-consuming. Moreover, because we concatenate predefined template texts pieces to compose final explanation texts, the resulting explanation texts may occasionally come across as unnatural and cumbersome.
  To address these issues, we believe that Natural Language Generation (NLG) techniques could be leveraged. There exist works aiming at rationalizing explanation text generation, in the broader context of XAI, incorporating NLG techniques, among others [BPO19]. For the authors, the explanation generation process can be conceptualized as a sequence of three main phases: *i)* content extraction from an instantiated AI model, *ii)* semantic representation of this content, and finally, *iii)* text generation using NLG techniques. While content extraction is specific to each AI model, the two other components are common to all AI models, enabling a shared framework. Consequently, the

authors' ambition is to build a semantic representation that is independent of the AI model, allowing any specialist in an XAI model to represent their explanations without needing to delve into the textual aspects. [Baa22] represents an initial endeavor towards achieving this goal, but there remains work to be done in order to devise a comprehensive proposal. Nevertheless, once such a framework has matured, it could potentially relieve us of the need to manage the textual aspect of our explanations.

To conclude this manuscript, our work represents a step forward in the development and practical application of explanation methods in CO. It opens doors to further research, innovation, and real-world applications in the domain of decision support systems.

Bibliography

[Baa22] Ismaïl Baaj. *Explainability of possibilistic and fuzzy rule-based systems*. Theses, Sorbonne Université, January 2022.

[BADRDS+20] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, 58:82–115, 2020.

[BGG21] Bart Bogaerts, Emilio Gamba, and Tias Guns. A framework for step-wise explaining how to solve Constraint Satisfaction Problems. *Artificial Intelligence*, 300:103550, 2021.

[BPO19] Ismaïl Baaj, Jean-Philippe Poli, and Wassila Ouerdane. Some insights towards a unified semantic representation of explanation for eXplainable artificial intelligence. In *Proceedings of the 1st Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence (NL4XAI 2019)*, pages 14–19. Association for Computational Linguistics, 2019.

[BR08] David Bredström and Mikael Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. *European Journal of Operational Research*, 191(1):19–31, 2008.

[CCK+19] Michael Cashmore, Anna Collins, Benjamin Krarup, Senka Krivic, Daniele Magazzeni, and David Smith. Towards Explainable AI Planning as a service. In *International Conference on Automated Planning and Scheduling second workshop on Explainable Planning*, 2019.

[CJ06] Hadrien Cambazard and Narendra Jussien. Identifying and exploiting problem structures using explanation-based constraint programming. *Constraints*, 11:295–313, 2006.

[ČKL+20] Kristijonas Čyras, Amin Karamlou, Myles Lee, Dimitrios Letsios, Ruth Misener, and Francesca Toni. AI-assisted schedule explainer for nurse rostering. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, page 2101–2103, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[ČLMT19] Kristijonas Čyras, Dimitrios Letsios, Ruth Misener, and Francesca Toni. Argumentation for Explainable Scheduling. In *Proceedings of the thirty-third Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, pages 2752–2759. AAAI Press, 2019.

[CSK20] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. The emerging landscape of Explainable AI Planning and Decision Making. In *Proceedings of the twenty-ninth International Joint Conference on Artificial Intelligence*, pages 4803–4811. IJCAI Organization, 2020.

[CSLSQ16] José Arturo Castillo-Salazar, Dario Landa-Silva, and Rong Qu. Workforce Scheduling and Routing Problems: literature survey and computational study. *Annals of Operations Research*, 239:39–67, 2016.

[CTH16] Xi Chen, Barrett W. Thomas, and Mike Hewitt. The Technician Routing Problem with experience-based service times. *Omega*, 61:49–61, 2016.

[CTJ89] Balakrishnan Chandrasekaran, Michael Tanner, and John Josephson. Explaining control strategies in problem solving. *IEEE Expert*, 4:9–15, 1989.

[dK86] Johan de Kleer. Problem solving with the ATMS. *Artificial Intelligence*, 28:197–224, 1986.

[Dun95] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[DVK17] Finale Doshi-Velez and Been Kim. Towards a rigorous science of Interpretable Machine Learning. *arXiv: Machine Learning*, 2017.

[FSPC18] James Forrest, Somayajulu Sripada, Wei Pang, and George Coghill. Towards making NLG a voice for interpretable machine learning. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 177–182, Tilburg University, The Netherlands, nov 2018. Association for Computational Linguistics.

[GA19] David Gunning and David Aha. DARPA's Explainable Artificial Intelligence (XAI) program. *AI Magazine*, 40:44–58, 2019.

[GDP16] 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), 2016.

[Gin93] Matthew Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[GM13] Asvin Goel and Frank Meisel. Workforce routing and scheduling for electricity network maintenance with downtime minimization. *European Journal of Operational Research*, 231(1):210–228, 2013.

[GMR+18] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51:1–42, 2018.

[Gri75] H. P. Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41–58. Academic Press, New York, 1975.

[Hil90] Denis Hilton. Conversational processes and causal explanation. *Psychological Bulletin*, 107:65–81, 01 1990.

[HMKL18] Robert Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. *ArXiv*, abs/1812.04608, 2018.

[JO01] Narendra Jussien and Samir Ouis. User-friendly explanations for Constraint Programming. In *Proceedings of the eleventh Workshop on Logic Programming Environments*, 2001.

[Jun04] Ulrich Junker. 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*, page 167–172. AAAI Press, 2004.

[KKM+21] Benjamin Krarup, Senka Krivic, Daniele Magazzeni, Derek Long, Michael Cashmore, and David E. Smith. Contrastive explanations of plans through model restrictions, 2021.

[KSB21] Anton Korikov, Alexander Shleyfman, and Christopher Beck. Counterfactual explanations for optimization-based decisions in the context of the GDPR. In *Proceedings of the thirtieth International Joint Conference on Artificial Intelligence*, pages 4097–4103. IJCAI Organization, 2021.

[Lew73] David Lewis. *Counterfactuals*. Cambridge, MA, USA, Blackwell, 1973.

[LGM20] Qingzi Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the AI: Informing design practices for Explainable AI User Experiences. In *Proceedings of the 2020 Conference on Human Factors in Computing Systems*, page 1 – 15. Association for Computing Machinery, 2020.

[LGMO22] Mathieu Lerouge, Céline Gicquel, Vincent Mousseau, and Wassila Ouerdane. Generating user-centered contrastive explanations for the Workforce Scheduling and Routing Problem. Working paper https://hal.archives-ouvertes.fr/hal-03795653, 2022.

[LGMO23] Mathieu Lerouge, Céline Gicquel, Vincent Mousseau, and Wassila Ouerdane. Counterfactual explanations for workforce scheduling and routing problems. In *Proceedings of the 12th International Conference on Operations Research and Enterprise Systems (ICORES)*, pages 50–61. SCITEPRESS, 2023.

[Lin20] Alan Lindsay. 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*, 2020.

[Lip90] Peter Lipton. Contrastive explanation. *Royal Institute of Philosophy Supplement*, 27:247–266, 1990.

[LKS18] Jeremy Ludwig, Annaka Kalton, and Richard Stottler. Explaining complex scheduling decisions. In *Proceedings of the 2018 Association of Computing Machinery conference on Intelligent User Interfaces*, volume 2068, 2018.

[MHV18] P. Matl, R. F. Hartl, and T. Vidal. Workload equity in vehicle routing problems: A survey and analysis. *Transportation Science*, 52(2):239–260, 2018.

[Mil19] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[Mil21] Tim Miller. Contrastive explanation: a structural-model approach. *The Knowledge Engineering Review*, 36:e14, 2021.

[MSV19] Federico Mosquera, Pieter Smet, and Greet Vanden Berghe. Flexible home care scheduling. *Omega*, 83:80–95, 2019.

[MZR21] Sina Mohseni, Niloofar Zarei, and Eric D. Ragan. A multidisciplinary survey and framework for design and evaluation of Explainable AI systems. *Associtation for Computing Machinery Transactions on Interactive Intellent Systems*, 11(3–4), 2021.

[POP21] Jean-Philippe Poli, Wassila Ouerdane, and Régis Pierrard. 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)*, pages 1–6, 2021.

[Sav92] Martin Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, 4:146–154, 1992.

[SF96] Mohammed Sqalli and Eugene Freuder. Inference-based Constraint Satisfaction supports explanation. In *Proceedings of the Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, volume 1, pages 318–325. AAAI Press, 1996.

[SS87] William Swartout and Stephen Smoliar. On making expert systems more like experts. *Expert Systems*, 4(3):196–208, 1987.

[SSK18] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In *Proceedings of the twenty-seventh International Joint Conference on Artificial Intelligence*, pages 4829–4836. IJCAI Organization, 2018.

[SV13] Silvia Schwarze and Stefan Voß. Improved load balancing and resource utilization for the skill vehicle routing problem. *Optimization Letters*, 7(8):1805–1823, 2013.

[SVZ14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[VDG19] Gabriel Volte, Chloé Desdouits, and Rodolphe Giroudeau. The Workforce Routing and Scheduling Problem: solving real-world instances. In *Proceedings of the ninth International Network Optimization Conference*, pages 60–65. OpenProceedings, 2019.

[VL21] Giulia Vilone and Luca Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106, 2021.

[WMR18] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31, 2018.

[WT92] Michael Wick and William Thompson. Reconstructive Expert System explanation. *Artificial Intelligence*, 54:33–70, 1992.

[YJ95] L. Richard Ye and Paul E. Johnson. The impact of explanation facilities on user acceptance of expert systems advice. *MIS Quarterly*, 19(2):157–172, 1995.

[ZF14] Matthew Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833, 2014.

Appendix A Illustrative example

A.1 Instance

Employee i in $\mathcal{E}$	**Name**	**Skill level** ske_i in $\mathbb{N}^\star$	**Location** as (lat., long.) in deg.	**Time-window** $[lbe_i,\ ube_i]$ as time range	as integer range
1	Ellen	2	(47.773, 16.193)	[8:00AM, 6:00PM]	[480, 1080]
2	Alex	3	(47.598, 15.785)	[9:00AM, 6:00PM]	[540, 1080]
3	Adam	2	(48.188, 14.862)	[8:00AM, 6:00PM]	[480, 1080]
4	Fabian	1	(48.069, 14.485)	[8:00AM, 6:00PM]	[480, 1080]
5	Carlotta	1	(47.463, 15.351)	[8:00AM, 6:00PM]	[480, 1080]

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

Table A.1: Description of the WSRP instance $\mathcal{I}$ on which is based the solution represented in Figure 3.1. 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 ske_i, a departure and return location, and a time-window $[lbe_i,\ ube_i]$. The second table describes the data about the tasks. Each task j is described by a skill level skt_j, a location, a duration dt_j and a time-window $[lbt_j,\ ubt_j]$. It is assumed that all the employees have the same traveling speed of 50km/h. Considering that the earth radius is 6731km, the traveling time, in minutes, between two locations $(lat_1, long_1)$ and $(lat_2, long_2)$ are 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 3.2.1 for the definition of a WSRP instance.

A.2 Solution

Employee i	Planning $(\mathcal{R}_i, \mathcal{C}_i)$								
1	$\mathcal{R}_1 = (d_1,$	$7,$	$30,$	$3,$	$26,$	$1,$	$17,$	$8,$	$r_1)$
	$\mathcal{C}_1 =$ (504,	525,	567,	662,	720,	840,	900,	1009,	1080)
	$\equiv$ (08:05AM,	08:45AM,	09:27AM,	11:02AM,	12:00PM,	02:00PM,	03:00PM,	04:49PM,	06:00PM)
2	$\mathcal{R}_2 = (d_2,$	$28,$	$16,$	$25,$	$14,$	$20,$	$6,$	$19,$	$r_2)$
	$\mathcal{C}_2 =$ (540,	600,	653,	702,	822,	888,	933,	1019,	1080)
	$\equiv$ (09:00AM,	10:00AM,	10:53AM,	11:42AM,	01:42PM,	02:48PM,	03:33PM,	04:59PM,	6:00PM)
3	$\mathcal{R}_3 = (d_3,$	$11,$	$9,$	$10,$	$24,$	$r_3)$			
	$\mathcal{C}_3 =$ (480,	542,	670,	717,	840,	906)			
	$\equiv$ (08:00AM,	09:02AM,	11:10AM,	11:57AM,	02:00PM,	03:06PM)			
4	$\mathcal{R}_4 = (d_4,$	$13,$	$18,$	$21,$	$29,$	$4,$	$22,$	$r_4)$	
	$\mathcal{C}_4 =$ (480,	482,	564,	656,	738,	866,	925,	1080)	
	$\equiv$ (08:00AM,	08:02AM,	09:24AM,	10:56AM,	12:18PM,	02:26PM,	03:25PM,	06:00PM)	
5	$\mathcal{R}_5 = (d_5,$	$5,$	$23,$	$2,$	$r_5)$				
	$\mathcal{C}_5 =$ (529,	600,	840,	1009,	1080)				
	$\equiv$ (08:49AM,	10:00AM,	02:00PM,	04:49PM,	06:00PM)				

Table A.2: Description of the solution $\mathcal{S} = ((\mathcal{R}_i, \mathcal{C}_i))_{i \in \mathcal{E}}$ represented in Figure 3.1. $\mathcal{S}$ is a feasible solution of the WSRP instance given in Table A.1. Each employee $i \in \mathcal{E}$ is associated with a planning $(\mathcal{R}_i, \mathcal{C}_i)$ made of: firstly, a route $\mathcal{R}_i$, which is a sequence of activities starting with the departure d_i of i (from their personal location), followed with the tasks $j \in \mathcal{T}$ that i performs, and ending with the return r_i of i (to their personal location); secondly, 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 3.2.1 for the definition of a solution to a WSRP instance.

A.3 Time slacks

Employee i	Sequence of BTS	Sequence of FTS
1	(5, 0, 3, 2, 0, 15, 28, 55, 55)	(76, 56, 53, 53, 48, 33, 20, 0, 0)
2	(0, 0, 0, 7, 67, 67, 67, 75, 75)	(45, 45, 45, 38, 8, 8, 8, 0, 0)
3	(0, 0, 0, 0, 0, 0)	(10, 10, 10, 54, 20, 174)
4	(0, 0, 0, 0, 23, 26, 25, 25)	(25, 25, 25, 25, 2, 0, 0, 0)
5	(49, 49, 141, 169, 169)	(80, 80, 30, 0, 0)

Table A.3: Description of the sequences of Backward Time Slacks (BTS) and Forward Time Slacks (FTS), in minutes, corresponding to the feasible solution $\mathcal{S}$ given in Table A.2. The mathematical formula used for computing BTS and FTS are given in Subsection 3.3.1.

Appendix B ILP solution

Employee i	Binary decision variables U_{ijk}
1	$U_{1,d_1,7} = U_{1,7,30} = U_{1,30,3} = U_{1,3,26} = U_{1,26,1}$ $= U_{1,1,17} = U_{1,17,8} = U_{1,8,r_1} = 1$ $U_{1jk} = 0$ for all other couples of activities (j,k)
2	$U_{2,d_2,28} = U_{2,28,16} = U_{2,16,25} = U_{2,25,14} = U_{2,14,20}$ $= U_{2,20,6} = U_{2,6,19} = U_{2,19,r_2} = 1$ $U_{2jk} = 0$ for all other couples of activities (j,k)
3	$U_{3,d_3,11} = U_{3,11,9} = U_{3,9,10} = U_{3,10,24} = U_{3,24,r_3} = 1$ $U_{3jk} = 0$ for all other couples of activities (j,k)
4	$U_{4,d_4,13} = U_{4,13,18} = U_{4,18,21} = U_{4,21,29} = U_{4,29,14}$ $= U_{4,14,22} = U_{4,22,r_4} = 1$ $U_{4jk} = 0$ for all other couples of activities (j,k)
5	$U_{5,d_5,5} = U_{5,5,23} = U_{5,23,2} = U_{5,2,r_5} = 1$ $U_{5jk} = 0$ for all other couples of activities (j,k)

Task j	Int. dec. var. T_j	Task j	Int. dec. var. T_j
1	840	17	900
2	1009	18	564
3	662	19	1019
4	866	20	888
5	600	21	656
6	933	22	925
7	525	23	840
8	1009	24	840
9	670	25	702
10	717	26	720
11	542	27	0
12	0	28	600
13	482	29	738
14	822	30	567
15	0	31	0
16	653		

Table B.1: Description of the (feasible) ILP-solution $\mathcal{X} = \varphi^{-1}(\mathcal{S})$ associated with the (feasible) solution $\mathcal{S}$ given in Table A.2. The first table gives the values of the binary decision variables (U_{ijk}) and the second table gives the values of the integer decision variables (T_j). $\mathcal{X}$ can be obtained from $\mathcal{S}$ by applying φ^{-1} described in Algorithm C.2. Conversely, $\mathcal{S}$ can be obtained from $\mathcal{X}$ by applying φ described in Algorithm C.1. See Subsection 3.2.2 for the definition of the ILP model (including the decision variables) and the bijection φ.

Appendix C Bijection

Algorithm C.1: Bijection from a feasible to ILP-solution to a feasible solution

Input :
$\mathcal{X} = ((T_j), (U_{ijk}))$ a feasible ILP-solution (w.r.t. an instance $\mathcal{I}$)

$\mathcal{S} \leftarrow ()$
for $i \in \mathcal{E}$ **do**
  $\mathcal{R}_i \leftarrow ()$
  add d_i at the end of $\mathcal{R}_i$
  $j \leftarrow d_i$
  $k \leftarrow d_i$
  while $k \neq r_i$ **do**
    $k \leftarrow$ activity after k in $\mathcal{A}_i$
    if $k = j$ **then**
      $k \leftarrow$ activity after k in $\mathcal{A}_i$
    while $U_{ijk} \neq 1$ **do**
      $k \leftarrow$ activity after k in $\mathcal{A}_i$
    $j \leftarrow k$
    add j at the end of $\mathcal{R}_i$
  $\mathcal{C}_i \leftarrow ()$
  add lbe_i at the end of $\mathcal{C}_i$
  $j \leftarrow$ activity after d_i in $\mathcal{R}_i$
  while $j \neq r_i$ **do**
    add $st_j \leftarrow T_j$ at the end of $\mathcal{C}_i$
    $j \leftarrow$ activity after j in $\mathcal{R}_i$
  add ube_i at the end of $\mathcal{C}_i$
  add $(\mathcal{R}_i, \mathcal{C}_i)$ to $\mathcal{S}$

Output :
$\mathcal{S}$ a feasible solution

Algorithm C.2: Bijection from a feasible solution to a feasible ILP-solution

Input :
$\mathcal{S}$ a feasible solution (w.r.t. an instance $\mathcal{I}$)

$\mathcal{X} \leftarrow ()$
for $j \in \mathcal{T}$ **do**
  add $T_j \leftarrow 0$ to $\mathcal{X}$
for $i \in \mathcal{E}$ **do**
  for $j \in \mathcal{A}_i \setminus \{r_i\}$ **do**
    for $k \in \mathcal{A}_i \setminus \{d_i, j\}$ **do**
      add $U_{ijk} \leftarrow 0$ to $\mathcal{X}$
for $i \in \mathcal{E}$ **do**
  $j \leftarrow$ activity after d_i in $\mathcal{R}_i$
  $U_{i,d_i,j} \leftarrow 1$ in $\mathcal{X}$
  $k \leftarrow$ activity after j in $\mathcal{R}_i$
  while $k \neq r_i$ **do**
    $T_k \leftarrow st_k$ in $\mathcal{X}$
    $U_{ijk} \leftarrow 1$ in $\mathcal{X}$
    $j \leftarrow k$
    $k \leftarrow$ index of the activity after k in $\mathcal{R}_i$
  $U_{i,j,r_i} \leftarrow 1$ in $\mathcal{X}$

Output :
$\mathcal{X}$ a feasible ILP-solution

Appendix D Synthèse en français

Cette thèse cherche à répondre à un défi lié à l'utilisation de systèmes d'aide à la décision basés sur de l'optimisation combinatoire, à savoir, expliquer à des utilisateurs les solutions obtenues en sortie de ces systèmes. L'approche proposée dans cette thèse pour répondre à ce défi est appliquéee au cas du problème de planification d'employés mobiles, en anglais *Workforce Scheduling and Routing Problem* (*WSRP*). Le choix du WSRP est, entre autres, motivé par les besoins de notre partenaire industriel, DecisionBrain, spécialisé dans le développement de systèmes d'aide à la décision basés sur de l'optimisation combinatoire. En cherchant à expliquer les résultats obtenus en sortie de tels systèmes d'aide à la décision, ce travail s'inscrit dans le domaine de l'Intelligence Artificielle explicable, *Explainable Artificial Intelligence* (*XAI*) et pose des questions de recherche telles que la modélisation mathématique des explications en contexte d'optimisation combinatoire, le calcul efficace des explications, et la communication des explications aux décideurs.

Le chapitre d'introduction de la thèse expose les difficultés rencontrées par les décideurs utilisant les systèmes d'aide à la décision basés sur de l'optimisation combinatoire, notamment le manque de compréhension des concepts mathématiques et des principes algorithmiques sous-jacents à ces systèmes, pouvant conduire à une perte de confiance et à un besoin en explication de leurs parts.

Le chapitre 2 souligne la nature interdisciplinaire de la recherche sur les explications, qui se développe notamment dans le cadre des sciences sociales et des sciences informatiques.

Le chapitre 3 formalise le WSRP en tant que programme linéaire en nombres entiers avec une fonction bi-objectif lexicographique. Il introduit également différentes familles de transformations de solutions ainsi que leurs voisinages associés.

Le chapitre 4 se penche sur la modélisation mathématique du processus d'explication en contexte d'optimisation combinatoire, en introduisant un cadre original reliant les observations faites par les utilisateurs à propos de solutions de WSRP à des questions, puis aux explications en réponse à ces questions. Cette modélisation exploite notamment les familles de transformations de solutions et les voisinages associés, détaillés dans le chapitre 3. Trois types d'explications, contrastives, scénarios et contrefactuelles, sont définis mathématiquement en fonction de la faisabilité de programmes mathématiques, qui s'appuient sur le programme linéaire en nombres entiers bi-objectif lexicographique modélisant le WSRP.

Le chapitre 5 présente une méthode de génération de textes explicatifs en réponse aux questions des utilisateurs. Indépendamment du type de question posée et fournie en entrée, la méthode identifie un contenu mathématique pertinent, appelé contenu support, pour construire des textes explicatifs à l'aide de modèles de texte préétablis. L'efficacité des algorithmes utilisés est mise en évidence par des expériences numériques, bien que des défis computationnels potentiels soient identifiés pour les méthodes basées sur la résolution de programmes linéaires en nombres entiers.

Le chapitre 6 introduit un système visant à structurer les interactions entre les utilisateurs finaux et les méthodes de génération d'explications. Ce système utilise un historique d'instances et de solutions pour fournir des explications aux utilisateurs et leur permet également d'explorer de manière incrémentielle les espaces de solutions et d'instances voisines. Une interface graphique intègre ce système, permettant aux utilisateurs de demander des explications et de comparer différentes instances et solutions de WSRP.

Les contributions principales de cette thèse résident dans la création d'un cadre pour le processus d'explication en optimisation combinatoire, la génération efficace de textes explicatifs centrés sur l'utilisateur, et la conception d'un système interactif permettant aux utilisateurs de poser des questions, obtenir des textes explicatifs et d'explorer des solutions et instances du WSRP. La thèse souligne l'importance de l'explicabilité dans les systèmes d'aide à la décision basés sur de l'optimisation combinatoire et ouvre la voie à de nouvelles recherches visant à étendre l'approche développée à d'autres problèmes que le cas de WSRP étudié.