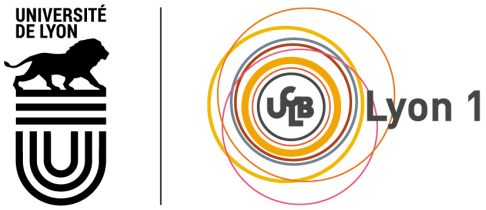
Learning behaviours aligned with moral values in a multi-agent system: guiding reinforcement learning with symbolic judgments

Rémy Chaput



N°ordre NNT: 2022LYO1071

THÈSE de DOCTORAT DE
L'UNIVERSITÉ CLAUDE BERNARD LYON 1

École Doctorale 512
InfoMaths

Discipline : Informatique

Soutenue publiquement le 27/10/2022, par :
Rémy Chaput

Learning behaviours aligned with moral values in a multi-agent system: guiding reinforcement learning with symbolic judgments

---

Devant le jury composé de :

GHODOUS Parisa	Professeure	Université Claude Bernard Lyon 1	Présidente
BONNET Grégory	Maître de conférences HDR	Université de Caen Normandie	Rapporteur
SLAVKOVIK Marija	Professeure	Université de Bergen	Rapporteure
DUTECH Alain	Chargé de recherche HDR	INRIA Nancy	Examinateur
RODRÍGUEZ-AGUILAR Juan A.	Professeur	Université autonome de Barcelone	Examinateur
HASSAS Salima	Professeure	Université Claude Bernard Lyon 1	Directrice de thèse
BOISSIER Olivier	Professeur	École des Mines de Saint-Étienne	Co-directeur de thèse
GUILLERMIN Mathieu	Maître de conférences	Université Catholique de Lyon	Co-directeur de thèse

Abstract

The past decades have seen tremendous progress of Artificial Intelligence techniques in many fields, reaching or even exceeding human performance in some of them. This has led to computer systems equipped with such AI techniques being deployed from the constrained and artificial environments of laboratories into our human world and society, in order to solve tasks with real and tangible impact. These systems have a more or less direct influence on humans, be it their lives in the most extreme cases, or in a more subtle but ubiquitous way, their daily lives. This raises questions about their ability to act in accordance with the moral values that are important to us.

Various fields of research have addressed aspects of this problem, such as the ability to provide fair and just decisions, or the ability to be intelligible, thus providing human users with reasons to trust them, and to know when not to trust. In this thesis, we focus particularly on the area of *Machine Ethics*, which is concerned with producing systems that have the means to integrate ethical considerations, i.e., systems that make ethical decisions in accordance with the human values that are important to society.

Our aim is thus to propose systems that are able to learn to exhibit behaviours judged as ethical by humans, both in situations of non-conflicting ethical stakes, but also in more complex cases of dilemmas between moral values. We propose 3 contributions, each with a different objective, which can be taken independently of each other, but which have been conceived and thought to work together, to combine their benefits, and to address the overall problem.

Firstly, we propose a reinforcement learning algorithm, capable of learning to exhibit behaviours incorporating these ethical considerations from a reward function. The goal is to learn these ethical considerations in many situations over time. A multi-agent framework is used, which allows on the one hand to increase the richness of the environment, and on the other hand to offer a more realistic simulation, closer to our intrinsically multi-agent human society, in which these approaches are bound to be deployed. We are particularly interested in the question of how agents adapt to changes, both to environmental dynamics, such as seasonal changes, but also to variations in the ethical mores commonly accepted by society.

Our second contribution focuses on the design of the reward function to guide the learning of agents. We propose to integrate judging agents, based on symbolic reasoning, tasked with judging learning agents' actions, and determining their reward, relatively to a specific moral value. The introduction of multiple judging agents makes the existence of multiple moral values explicit. Using symbolic judgment facilitates the design by application domain experts, and improves the intelligibility of the produced rewards, providing a window into the motivations that learning agents receive.

Thirdly, we focus more specifically on the question of dilemmas. We take advantage of the existence of multiple moral values and associated rewards to provide information to the learning agents, allowing them to explicitly identify those dilemma situations, when 2 (or more) moral values are in conflict and cannot be satisfied at the same time. The objective is to learn, on the one hand, to identify them, and, on the other hand, to decide how to settle them, according to the preferences of the system's users. These preferences are contextualized, i.e., they depend both on the situation in which the dilemma takes place, and on the user. We draw on multi-objective learning techniques to propose an approach capable of recognizing these conflicts, learning to recognize dilemmas that are similar, i.e., those that can be settled in the same way, and finally learning the preferences of users.

We evaluate these contributions on an application use-case that we propose, define, and implement: the allocation of energy within a *smart grid*.

Keywords: Machine Ethics ; Artificial Moral Agents ; Multi-Agent Systems ; Multi-Agent Reinforcement Learning ; Multi-Objective Reinforcement Learning ; Ethical Judgment ; Hybrid Neural-Symbolic Learning ; Moral Dilemmas ; Human Preferences

Résumé (Fr)

Les précédentes décennies ont vu un immense progrès des techniques d'Intelligence Artificielle, dans de nombreux domaines, allant jusqu'à atteindre, voire dépasser, les performances humaines dans certains d'entre eux. Cela a mené des systèmes informatiques équipés de telles techniques d'IA à quitter les environnements contraints et artificiels des laboratoires, pour être déployés dans notre monde et notre société humaine, afin de résoudre des tâches ayant un impact bien réel. Ces systèmes ont une influence plus ou moins directe sur des humains, que ce soit leur vie pour les cas les plus extrêmes, ou de manière plus subtile, mais plus ubiquitaire, leur quotidien. Des questions se posent ainsi quant à leur capacité d'agir en accord avec les valeurs (morales) qui nous semblent importantes.

Divers champs de recherche se sont intéressés à des aspects de ce problème, tels que la capacité à fournir des décisions équitables et justes, ou encore la capacité à être intelligible, et ainsi fournir aux utilisateurs humains des raisons d'accorder leur confiance, et de savoir quand ne pas l'accorder. Dans cette thèse, nous nous concentrons particulièrement sur le domaine des *Machine Ethics*, qui consiste à produire des systèmes ayant les moyens d'intégrer des considérations éthiques, c'est-à-dire des systèmes ayant une prise de décision éthique, en accord avec les valeurs humaines qui sont importantes à la société.

Notre but est ainsi de proposer des systèmes, qui soient capables d'apprendre à exhiber des comportements jugés comme éthiques par les humains, à la fois dans des situations ayant des enjeux éthiques non en conflit, mais aussi dans les cas plus complexes de dilemmes entre les valeurs morales. Nous proposons 3 contributions, chacune ayant un objectif différent, pouvant être prises indépendamment les unes des autres, mais ayant été conçues pour s'associer afin de combiner leurs avantages, et de répondre à la problématique globale.

Premièrement, nous proposons un algorithme d'apprentissage par renforcement, capable d'apprendre à exhiber des comportements intégrant ces considérations éthiques à partir d'une fonction de récompense. Le but est ainsi d'apprendre ces enjeux éthiques dans de nombreuses situations, au fil du temps. Un cadre multi-agent est utilisé, ce qui augmente

d'une part la richesse de l'environnement, et d'autre part offre une simulation plus réaliste, plus proche de notre société humaine, intrinsèquement multi-agent, et dans laquelle ces approches sont vouées à être déployées. Nous nous intéressons particulièrement à la question de l'adaptation des agents aux changements, à la fois aux dynamiques de l'environnement, tels que les changements saisonniers, mais aussi aux variations dans les mœurs éthiques couramment acceptées par la société.

Notre deuxième contribution se concentre sur la conception de la fonction de récompense, afin de guider l'apprentissage. Nous proposons l'intégration d'agents juges, se basant sur du raisonnement symbolique, chargés de juger les actions des agents apprenants et déterminer leur récompense, relativement à une valeur morale spécifique. L'introduction de multiples agents juges permet de rendre explicite l'existence de multiples valeurs morales. L'utilisation de jugement symbolique facilite la conception par des experts du domaine applicatif, et permet d'améliorer l'intelligibilité des récompenses ainsi produites, ce qui offre une fenêtre sur les motivations que reçoivent les agents apprenants.

Troisièmement, nous nous focalisons plus précisément sur la gestion des dilemmes. Nous profitons de l'existence de multiples valeurs morales afin de fournir plus d'informations aux agents apprenants, leur permettant ainsi d'identifier explicitement ces situations de dilemme, lorsque 2 valeurs morales (ou plus) sont en conflit et ne peuvent être satisfaites en même temps. L'objectif est d'apprendre, d'une part à les identifier, et d'autre part à les trancher, en fonction des préférences des utilisateurs et utilisatrices du système. Ces préférences sont contextualisées, elles dépendent ainsi à la fois de la situation dans laquelle se place le dilemme, et à la fois de l'utilisateur. Nous nous inspirons des techniques d'apprentissage multi-objectif afin de proposer une approche capable de reconnaître ces conflits, d'apprendre à reconnaître les dilemmes qui sont similaires, autrement dit ceux qui peuvent être tranchés de la même manière, et finalement d'apprendre les préférences des utilisateurs et utilisatrices.

Nous évaluons ces contributions sur un cas d'application que nous proposons, définissons et implémentons : la répartition d'énergie au sein d'une *smart grid*.

Mots-clés: Éthique computationnelle ; Agents moraux artificiels ; Systèmes multi-agent ; Apprentissage par renforcement multi-agent ; Apprentissage par renforcement multi-objectif ; Jugement éthique ; Apprentissage hybride neuro-symbolique ; Dilemmes moraux ; Préférences humaines

Résumé substantiel

Durant les dernières années, les techniques d'Intelligence Artificielle (IA) se sont considérablement développées, atteignant d'impressionnantes performances dans de nombreux domaines. Parallèlement, les applications et systèmes incluant de telles techniques d'IA sont de plus en plus déployées dans notre société, ayant un impact sur celle-ci. Par exemple, certaines décisions tels que les demandes de prêts, les décisions de justice, les recrutements de candidats, etc., commencent à être effectuées par des systèmes d'IA, avec peu ou pas de supervision humaine.

Alors que nous déléguons de plus en plus de tâches à ces systèmes, leur fournissant ainsi une autonomie "fonctionnelle", il devient impératif de s'assurer qu'ils accomplissent leurs tâches d'une manière alignée sur nos valeurs, notamment morales, en les dotant de capacités de traitement et de raisonnement à cet effet. Cette question a mené lieu à la création du domaine de recherche de l'éthique computationnelle (*Machine Ethics*), visant à doter ces systèmes de moyens de résoudre les dilemmes éthiques qui leur seront posés. L'objectif final étant ainsi de s'assurer un impact bénéfique de tels agents sur notre société et sur nos vies.

Dans cette thèse, nous adoptons une approche centrée sur l'humain et proposons des algorithmes permettant à des agents d'artificiels d'apprendre et exhiber des comportements qui soient alignés sur les valeurs définies par les concepteurs du système, en accord avec les préférences des utilisateurs humains et autres parties-prenantes de la société dans laquelle ces agents seront intégrés.

Nos objectifs sont les suivants :

1. Représenter et capturer la diversité des valeurs morales et préférences éthiques au sein d'une société.
    1. Cette diversité dérive de multiples sources : multiples personnes, multiples valeurs et multiples situations.
    2. Les valeurs morales couramment acceptées, ainsi que leur définitions, c'est-à-dire le "consensus" éthique faisant partie des normes et mœurs sociétales, peut évoluer sur le temps.

2. Apprendre des comportements en situation, qui soient correctement alignés sur des valeurs morales humaines.
    1. Les comportements doivent considérer des situations sans dilemme, qui peuvent impliquer des enjeux éthiques, mais non nécessairement en conflit.
    2. Les comportements doivent également considérer les situations de dilemme, où les enjeux sont en conflit.
    3. Le système doit apprendre selon une spécification correctement définie du comportement désiré, afin d'éviter les phénomènes de "reward hacking".
3. Implémenter un cas d'application prototype afin de démontrer la faisabilité de l'approche.

Pour cela, nous proposons 3 contributions :

1. Deux algorithmes d'apprentissage par renforcement qui apprennent des comportements "en général" (sans emphase particulière sur les dilemmes), en utilisant des domaines continus et en se focalisant particulièrement sur l'adaptation aux changements, à la fois dans les dynamiques de l'environnement et dans les fonctions de récompense.

2. Une méthode hybride se basant sur des jugements symboliques pour construire la fonction de récompense à partir de plusieurs valeurs morales, et permettant la compréhensibilité de la fonction résultante.

3. Une extension multi-objectif des algorithmes d'apprentissage par renforcement qui se focalise sur la résolution de dilemmes, par l'apprentissage des actions intéressantes pour n'importe quelle préférence humaine, l'identification des situations de dilemme et l'apprentissage de comment trancher ces dilemmes explicitement, selon les préférences humaines contextualisées.

Dans ce manuscrit, nous explorons l'état de l'art des domaines de l'éthique computationnelle, de l'apprentissage par renforcement, multi-agent et multi-objectif, et de l'IA hybride neuro-symbolique.

Nous en retirons plusieurs propriétés et choix de conception que nous jugeons importants pour notre approche. Premièrement, l'apprentissage par renforcement est une méthode appropriée pour apprendre des comportements dépendant des situations. Deuxièmement, l'utilisation de domaines continus, à la fois pour les situations mais aussi les actions, permettent une représentation plus riche, plus complexe, mais aussi plus pratique, par rapport aux domaines discrets. Troisièmement, les systèmes multi-agents nous permettent

d'adresser l'objectif de diversité, en représentant différentes personnes, et en encodant diverses valeurs morales. Cela permet des environnements plus réalistes, car de tels agents sont voués à être intégrés à notre société, qui est naturellement multi-agent. Quatrièmement, il est important de considérer de multiples valeurs morales ; dans de nombreux cas, nos prises de décisions sont guidées par plusieurs valeurs, et il est important d'entraîner ces agents sur différentes valeurs également. Ces valeurs pouvant être en conflit entre elles, cela soulève des questions supplémentaires et demande des mécanismes supplémentaires. Finalement, l'utilisation d'une méthode hybride, passant par l'agentification de la construction de la récompense, permet de paver le chemin vers une co-construction entre des agents artificiels et des humains, et facilite l'adaptation à un consensus éthique changeant.

Afin de prouver la faisabilité de notre approche, nous validons nos contributions sur un simulateur de réseau électrique intelligent (*Smart Grid*), dans lequel les agents sont chargés de se répartir une quantité d'énergie, afin de subvenir aux besoins des habitants qu'ils représentent.

Ce cas d'application, qui n'enlève rien à l'aspect généralisable de nos propositions, s'appuie sur les valeurs morales suivantes :

Assurance de confort : les habitants doivent voir leur confort aussi satisfait que possible.

Abordabilité : les habitants ne doivent pas payer trop cher pour satisfaire leur confort et accéder à l'électricité.

Inclusivité : les habitants doivent avoir un accès équitable à l'électricité et avoir un confort similaire.

Viabilité environnementale : les agents doivent éviter d'acheter de l'énergie provenant de sources polluantes.

Notre première contribution concerne 2 algorithmes d'apprentissage, nommés Q-SOM et Q-DSOM, qui reposent sur l'utilisation de Cartes Auto-Organisatrices (*Self-Organizing Map* – SOM), aussi appelées cartes de Kohonen. Ces cartes permettent de faire le lien entre des domaines continus pour les états du monde et les actions de l'agent et des identifiants discrets utilisables dans une Q-Table pour apprendre les intérêts de chaque paire état-action, selon un algorithme proche du *Q-Learning*.

Les principaux intérêts de ces deux algorithmes, face aux autres algorithmes de l'état de l'art, résident en :

- leur capacité à manipuler des espaces continus tout en offrant une représentation discrète, ce qui permet une manipulation et une comparaison des états et actions ;
- leur capacité à s'adapter au changement, grâce aux propriétés des cartes de Kohonen, et de leur extension, les cartes dynamiques (DSOM).

Ces algorithmes sont évalués sur diverses fonctions de récompense, qui intègrent des considérations éthiques, telles que les valeurs morales décrites précédemment. Certaines des fonctions mélangent plusieurs de ces considérations, afin de tester la capacité des agents à apprendre plusieurs valeurs morales. Enfin, deux des fonctions sont spécifiquement conçues pour changer de définition après un certain nombre de pas de temps, ce qui force les agents à s'adapter afin de maximiser leur espérance de récompenses.

Nous comparons nos algorithmes à d'autres algorithmes de l'état de l'art, tels que DDPG et son extension multi-agent, MADDPG.

Notre deuxième contribution concerne la construction de la récompense. Traditionnellement, les récompenses en apprentissage par renforcement sont calculées par des fonctions numériques. Nous proposons d'introduire des agents juges dans l'environnement, chargés de déterminer la récompense pour chacun des agents apprenants, selon les valeurs morales qu'ils représentent, à partir de techniques de raisonnement, telles que des règles logiques ou de l'argumentation.

De tels agents permettent :

- d'expliciter les différentes valeurs morales, et ainsi identifier plus facilement les conflits entre elles ;
- de spécifier le comportement attendu par un jugement symbolique, en déterminant les éléments acceptables parmi le comportement actuel de l'agent et les éléments inacceptables ou qui doivent être améliorés ;
- la modification et en particulier l'addition ou la suppression des règles morales par l'ajout ou la suppression d'agents juges ;
- une meilleure intelligibilité des motivations des agents apprenants, c'est-à-dire des récompenses qu'ils reçoivent.

Nous implémentons cette proposition sous 2 formes différentes. La première s'inspire des agents juges Ethicaa et se base sur des agents "Croyances, Désirs, Intentions" (*Be-*

liefs, Desires, Intentions – BDI) et des règles logiques afin de produire le raisonnement symbolique. La seconde approche se base sur des mécanismes d'argumentation.

Nous comparons ces 2 approches, à la fois vis-à-vis des récompenses numériques traditionnelles, mais aussi entre elles, afin d'évaluer les forces et faiblesses de chacune.

Notre troisième et dernière contribution se focalise sur la question des dilemmes et vise à intégrer un peu plus l'humain dans la boucle. Tandis que la seconde contribution, à des fins de simplification, agrégeait les récompenses produites par chaque agent juge, autrement dit par chaque valeur morale, dans cette contribution, nous étendons l'algorithme d'apprentissage pour recevoir en entrée de multiples récompenses, c'est-à-dire un cadre multi-objectif. Les agents apprenants reçoivent ainsi la récompense correspondant à chaque valeur morale individuellement.

Cela leur permet principalement d'identifier les situations de dilemmes, dans lesquelles il est impossible de satisfaire toutes les valeurs morales en même temps. À partir de cette identification, nous proposons de "trancher" les dilemmes selon les préférences des utilisateurs. Nous postulons que celles-ci sont contextualisées, c'est-à-dire qu'elles dépendent du contexte dans lequel se produit le dilemme : par exemple, nous ne ferions probablement pas le même choix en hiver qu'en été.

L'algorithme d'apprentissage est ainsi modifié afin de :

1. apprendre des actions qui soient intéressantes, quelles que soient les préférences humaines, afin de proposer des compromis aux utilisateurs si un dilemme survient ;
2. identifier automatiquement, selon une définition que nous proposons, qui s'inspire de la littérature existante, mais qui prend en compte les préférences humaines, les situations de dilemme ;
3. proposer les alternatives aux utilisateurs et apprendre à trancher ces dilemmes comme ils le souhaitent, en contexte.

Le découpage des dilemmes en contextes se fait en grande partie grâce à l'utilisateur.

Nous créons des profils d'utilisateur fictifs, afin de démontrer la capacité de cette approche à apprendre les préférences utilisateurs et à s'adapter à celui-ci.

Ainsi, nos 3 contributions, qui peuvent être prises séparément, mais qui sont conçues pour s'appuyer les unes sur les autres, compensant ainsi les faiblesses de chacune et proposant un cadre général, permettent de répondre à nos objectifs initiaux.

La diversité au sein de la société est traitée par :

- l'utilisation de multiples agents apprenants, qui s'appuient sur des profils différents, à la fois dans leurs habitudes de consommation, mais aussi leurs préférences sur les valeurs morales ;
- l'utilisation de multiples agents juges, qui représentent des valeurs morales différentes.

L'évolution du consensus éthique est traité par :

- les algorithmes d'apprentissage, qui sont conçus avec cet objectif en tête, qui intègrent des choix de conception cohérents avec celui-ci et qui sont spécifiquement évalués sur cet aspect ;
- la présence de multiples agents juges, qui facilite les mécanismes d'ajout ou de suppression, donc la modification, des valeurs morales.

L'apprentissage des comportements, avec ou sans emphase sur la notion de dilemmes, est traité par les algorithmes d'apprentissage, dont la première contribution apprend de manière générale, c'est-à-dire que les conflits entre valeurs morales sont moyennés pour obtenir un résultat satisfaisant en moyenne ; la troisième contribution apporte l'emphase sur les dilemmes pour traiter explicitement ces compromis, ce qui rend le système plus performant et également plus intelligible pour l'utilisateur, en explicitant ces points de blocage qu'il ne sait pas ou ne peut pas résoudre.

La construction de la récompense se fait par l'intermédiaire des agents juges, dont le raisonnement symbolique permet notamment d'intégrer des connaissances expertes. L'argumentation en particulier est très efficace pour éviter les phénomènes de *reward hacking*, grâce à sa relation d'attaque entre les arguments, afin d'empêcher des comportements précis.

Enfin, l'étude de la faisabilité a été démontrée au fur et à mesure par la conception d'un cas d'application, l'implémentation d'un simulateur et l'évaluation des contributions sur celui-ci.

Acknowledgments

I will (try to) keep this short, as the manuscript is already long enough.

First and foremost, I would like to thank my best friends, Dylan and Kévin, for supporting (and even enduring!) me throughout these 3 years (and even before that). Our numerous evenings of playing games have helped me release the stress of my daily (and sometimes nightly) work. Dylan deserves a golden (chocolate) medal for hearing my (too many) rants about administrative problems, my unconditional love for Linux, and hatred for Windows (to the point it became a running gag). The post-defence buffet would not have been such a success without his help. Kévin also deserves a medal for all the laughs we had – our sessions together were a *heist* of fun! – and for his legendary Breton crêpes.

Many thanks to Ambre for the (delicious) restaurants we had together, even though my belly fat and wallet do not participate in the appreciation; and to Maëlle for making me discover Friends, and for our "Simpson binge-watching" evenings.

This thesis would probably not have happened without my parents and my family: they have always pushed me towards academic excellence, and have nurtured my curiosity since primary school.

Special thanks to my friends, which, I know, I have not seen enough recently due to this thesis: Pierre, Grégory, Rayane, Bénédicte, and Admo. I promise you, now that's finished, we will see each other again!

I would also like to thank all of my colleagues at the LIRIS, who have made this research experience great, and especially my colleagues from the SyCoSMA team: Arthur, Simon (F.), Simon (P.), Bastien, Frédéric, Mathieu, Laetitia, Alain, Véronique. I will miss our profound (and sometimes stupid) conversations at lunchtime, but at least we will still have our board games evenings.

Last but not least, my thesis would not have been the same without my co-directors: Salima, Olivier, and Mathieu. Thank you for proposing such an interesting subject, accepting me as your PhD candidate, and more importantly, taught me like you did, both in your respective fields and in the general research methodology. It was great working

with you, and I hope we will be able to continue our fruitful collaboration in the years to come.

I would like to take this opportunity to pay a special homage to Salima in particular, who was kind enough to accept me on this exciting project. I was very happy to find something other than Deep Learning and computer vision, and I don't regret my choice after these years! Although she was always very (too?) busy, she always took time to help me when I needed it. I learned a lot from her (probably a few bad habits along the way, such as checking my email in meetings), and I intend to honor her teachings in my future career.

And finally, thank *you*, dear reader, for taking the time to read my manuscript!

Remy Chaput

Table of Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Context

During the last decades, Artificial Intelligence (AI) techniques have been considerably developed. These techniques have made impressive progress in various tasks, such as image recognition, prediction, behaviour learning, etc. In some specific domains, such AI techniques are even deemed to exceed human performance.

Due to this, AI-powered applications are increasingly deployed in our society. For example, loans can be granted based on the recommendation of an AI system. We might encounter more and more artificial agents, either physical, i.e., robots, or virtual, in the next years.

A potential risk, which is frequently mentioned, is the "Value Alignment Problem" (Dignum, 2019 ; World Economic Forum, 2015). We can indeed wonder to which degree these systems, while solving their given task, will adopt behaviours that are aligned, or misaligned, with our human values.

This question has led to the rise of several fields of research more or less related to the Ethics of AI. For example, the Fairness community aims to limit the learning of biases from the datasets that AI systems use to learn how to solve their task. The Explainable AI field aims to increase our ability to understand the process that these systems execute, such that we can agree with their outputs, or on the contrary question them. As such, it might help us know when to trust that these systems act accordingly to our values, and when we cannot trust them.

In this thesis, we will more specifically focus on the Machine Ethics field. This field "is concerned with giving machines ethical principles, or a procedure for discovering a way to resolve the ethical dilemmas they might encounter, enabling them to function in an ethically responsible manner through their own ethical decision making" (Anderson & Anderson, 2011).

As a general framework, we consider that AI systems are integrated in a human society (ours), and do not live in their own world. As such, we will not talk about the ethics of the

technical system alone, but rather the ethics of the whole Socio-Technical System (STS), which comprises both the artificial and human agents. Firstly, ethical considerations already exist in the social part of the system, and we can probably capture them for the technical part, either by (domain) expert knowledge, or user knowledge. Secondly, another consequence of such a framework is that the technical system can be used to improve humans' own ethical reasoning and considerations. Dilemmas can be leveraged to help users, and more generally stakeholders, reflect on their current considerations, acknowledge new dilemmas that they did not know about, and perhaps propose ways to evolve their ethical preferences and stances. As such, we should not automatically hide away dilemmas, but instead expose them to users. This requires some interpretability, and interaction, to allow for this co-construction of ethics within the STS. Whereas we did not solve these requirements (interpretability alone is a whole field of research), they served as some sort of guidelines, shaping the problems and potential approaches. Throughout the manuscript, we will sometimes refer to this general framework to emphasize the importance of some design choices. These design choices pave the way for the requirements of the general framework.

From this general framework, we particularly extract two assumptions. First, ethical stakes and the whole ethical knowledge that is necessary for ethical decision-making, i.e., the ability to make decisions that integrate ethical considerations and would be deemed as ethically-aligned by humans, necessarily comes from humans. Whether artificial agents can be fully moral agents is an out-of-scope debate for this manuscript, and we leave this question to moral philosophers. We will simply assume that current agents are probably not able to demonstrate the same level of ethical capabilities as humans, because they lack some metaphysical properties, such as free will. Despite this, we still would like them to make actions that consider, as much as possible, ethical considerations, so as to improve our lives.

Floridi and Sanders argue that, whereas there is a difference between artificial agents and humans, under the correct Level of Abstraction, some artificial agents can be said to have moral accountability. They propose an alternative definition of morality that allows including these artificial agents (Floridi & Sanders, 2004):

> An action is said to be morally qualifiable if and only if it can cause moral good or evil. An agent is said to be a moral agent if and only if it is capable of morally qualifiable action.

In order to make artificial agents capable to make morally good actions, rather than evil, we argue they need to receive ethical considerations, from those who possess them, i.e., humans. We will refer to this as the *ethical injection*, of which the particular form depends on the specificities of employed AI techniques, the envisioned use-case, etc.

The second assumption is that, although we share similar moral values, human ethical preferences are contextualized. They are different from a person to another, and between different situations. This means that, on the one hand, the ethical injection must be sufficiently rich to embed these contextualized preferences, and on the other hand, the artificial agents should learn the different preferences, and exhibit a behaviour consistent with the multiple preferences, instead of a single set.

Objectives

In this thesis, we focus on, and solve, the following objectives.

1. Represent and capture the diversity of moral values and ethical preferences within a society.
    1. This diversity derives from multiple sources: multiple persons, multiple values, and multiple situations.
    2. The currently accepted moral values, and their definitions, i.e., the "consensus" as part of societal norms and mores, may shift overtime.
2. Learn ethical behaviours, in situations, which are well-aligned with human values.
    1. Behaviours should consider non-dilemma situations, which may imply ethical stakes, but not necessarily in conflict.
    2. Behaviours should consider dilemmas situations, with conflicts between stakes.
    3. The system must learn according to a well-defined specification of the desired behaviour, to avoid "specification gaming"[1] or mis-alignment.
3. Implement a prototype use-case to demonstrate the feasibility of the approach.

To help us solve these objectives, we formulate the following design choices. These choices will be partially supported in our State of the Art.

[1] Also called the "King Midas problem" – getting exactly what you asked for, but not what you intended. See for example this blog post from Deep Mind for details and examples.

1. Reinforcement Learning (RL) is an appropriate method to learn situational behaviours.
2. Continuous domains (for both states and actions) allow for more complex environments, including multi-dimensional representations, which would be less practical to implement with discrete domains.
3. Multi-agent systems can help us address diversity, by representing different persons and encode various moral values.
4. Agentifying the reward construction paves the way for co-construction, with both artificial agents and humans, as per the general framework, and facilitates adaptation to shifting ethical consensus, as per our objectives.

From the aforementioned design choices and objectives, we pose the following research questions.

1. How to learn behaviours aligned with moral values, using Reinforcement Learning with complex, continuous, and multi-dimensional representations of actions and situations? How to make the multiple learning agents able to adapt their behaviours to changes in the environment?

2. How to guide the learning of agents through the agentification of reward functions, based on several moral values to capture the diversity of stakes?

3. How to learn to address dilemmas in situation, by first identifying them, and then settling them in interaction with human users? How to consider contextualized preferences in various situations of conflicts between multiple moral values?

We present 3 contributions that aim to answer our research questions; they are briefly introduced here.

1. Two RL algorithms that learn behaviours "in general" (without emphasis on dilemmas), using continuous domains, and particularly focusing on adaptation to changes, both in the environment dynamics and the reward functions.

2. A hybrid method relying on symbolic judgments to construct the reward function from multiple moral values, focusing on the understandability of the resulting function.

3. A multi-objective extension to the RL algorithms that focuses on addressing dilemmas, by learning interesting actions for any possible human preferences, identifying

situations of dilemma, and learning to settle them explicitly, according to contextualized human preferences.

Table 1.1 summarizes the relation between objectives, research questions and contributions.

Tab. 1.1.: Associations between objectives, research questions, and contributions. Objective O3 (implementation of prototype use-case) is transverse to all contributions.

Research Question	Contribution	Objective
RQ1	C1	O1.1
		O1.2
		O2.1
RQ2	C2	O1.1
		O2.3
RQ3	C3	O1.1
		O2.2

RQ1: How to learn behaviours aligned with moral values, using RL, with continuous domains, in a MAS, such that agents are able to adapt to changes?

RQ2: How to guide the learning through agentification of reward functions, based on several moral values?

RQ3: How to learn to address dilemmas in situations, according to users' contextualized preferences, with multiple moral values in various situations?

O1.1: Diversity of ethical stakes

O1.2: Shifting ethical mores

O2.1: Learn non-dilemmas

O2.2: Learn dilemmas

O2.3: Reward specification

C1: Rl algorithms; Learning behaviours; Continuous domains; Adaptation

C2: Symbolic judgments; Agentification of reward functions; Multiple judges

C3: Multi-Objective RL; Dilemmas; Interaction with users

Plan

This manuscript is divided into 7 chapters. The (current) first one has introduced the context of this thesis, as well as the objectives and research questions. Note that, in each of our contribution chapters, we present both the model and associated experiments and results. This allows contributions to be more clearly separated, and presented in an incremental manner. Advantages and limitations are immediately highlighted, before moving on to the next contribution chapter. In addition, it exemplifies each model by directly applying on a use-case, anchoring them in a practical context on top of the theoretical. Nevertheless, it does not detract from the genericity of our approach, which could be extended to other application domains.

Chapter 2 explores the different research areas related to our work, namely Machine Ethics, Multi-Agent Reinforcement Learning, Multi-Objective Reinforcement Learning, and Hybrid AI. It identifies recent advances, and the remaining challenges. An analysis and comparison of existing work raises interesting properties to be integrated in our contributions: continuous domains, multiple agents, multiple moral values, a hybrid approach, the ability to adapt to changes, and interaction with the user. Our problematic, objectives and research questions are detailed in the light of these elements.

Chapter 3 specifies our positioning, methodological approach, presents our architecture conceptually and describes our application case. The first part shows our vision of the integration of an artificial intelligence system within a human society, and how to integrate ethical considerations into these systems, through an ethical injection. The second part describes our methodology, as a multi-disciplinary approach that we treat incrementally through successive contributions, each validated separately and integrating with the others. It also positions our objectives in relation to the challenges identified in the state of the art. The third part conceptually presents our architecture, composed of 3 contributions, briefly introduces these contributions, how they are articulated between them, and presents the novelties of our approach. Finally, the fourth part presents the application case that we have chosen to validate our contributions: energy distribution within a Smart Grid. This presentation will allow us to exemplify each of the contributions and to anchor them in a practical case, although they are conceived as generic, and to facilitate the experiments' description.

Chapter 4 presents our first contribution: two reinforcement learning algorithms, Q-SOM and Q-DSOM, dedicated to learning behaviours aligned with moral values. These algorithms focus on the use of continuous domains and adaptation to changes in the dynamics of the environment. They are evaluated on multiple reward functions, testing various moral values, including functions that combine several values, or whose definition changes after a certain number of time steps.

Chapter 5 presents our second contribution: the agentification of reward functions, to compute these rewards through judgments, made by symbolic agents. We propose two ways to do this, the first one through logical rules and Beliefs-Desires-Intentions agents, whereas the second one uses argumentation graphs. In both cases, the moral values are explicitly defined by the system designer, and the judging agents reason about them in order to produce their judgment. We compare these two ways and evaluate the ability of agents using our learning algorithms to exhibit behaviours that correspond to these reward functions, i.e., that are aligned with the moral values on which they are based.

Chapter 6 presents our third and final contribution, which opens up the reward functions defined earlier to take a multi-objective approach. This allows agents to consider each of the moral values directly, and thus to identify situations that put one or other of these values in difficulty. We propose a definition of "dilemmas" adapted to our framework, which allows agents to recognize situations in which they are unable to satisfy all moral values at the same time. We also modify the learning algorithm so that the agents can offer alternatives to the user in these dilemma situations, i.e., actions that satisfy the

moral values to different degrees. Finally, agents learn to perform the actions chosen by users in similar contexts; we particularly emphasize the contextual aspect of user preferences.

Finally, Chapter 7 summarizes our contributions, their benefits, and how they address our research questions and objectives. It analyzes their various limitations, but also the questions and avenues of research that they open up.

2 Background knowledge and State of the Art

In this chapter, we introduce the necessary knowledge to understand our contributions, and we explore the state of the art in the fields related to our work: *Machine Ethics*, *Multi-Agent Reinforcement Learning*, *Multi-Objective Reinforcement Learning*, and *Hybrid Artificial Intelligence*. This exploration allows us on the one hand to compare the existing approaches, their advantages, but also their limitations, and to define some concepts necessary to the understanding of our work. On the other hand, we identify research challenges and thus precise our research questions.

As our research questions relate to the question of behaviours aligned with moral values, we begin this state of the art with the domain of *Machine Ethics*. The analysis of proposed approaches and of the properties considered as important opens the way to the other fields.

2.1 Machine Ethics

The field of Machine Ethics is relatively recent among the other fields of Artificial Intelligence, although its roots might be older. Indeed, as early as 1950, Wiener (1954) mentions the link between ethics and technology, including machines in the sense of cybernetics, and the need to use these technologies in a way that benefits humanity. On the other hand, the earliest works closer to the field of Machine Ethics as we know it today can, to the best of our knowledge, be traced back to around 2000 (Allen, Varner, & Zinser, 2000; Ashley & McLaren, 1995; Varela, 1999), although the name *Machine Ethics* does not appear in these papers. The article by Allen et al. (2000), in particular, introduces the term "Artificial Moral Agent" (AMA).

The term *Machine Ethics* appeared as early as 1987 (Waldrop, 1987), but it is from 2004 onwards that it began to gain academic momentum, under the impulse of Anderson, Anderson, & Armen (2004), then in 2005 with an AAAI Symposium dedicated to it.

From this symposium, a book published in 2011 gathers different essays on the nature of *Machine Ethics*, its importance, the difficulties and challenges to be solved, and also a few first approaches. This book defines this new field of research:

> The new field of machine ethics is concerned with giving machines ethical principles, or a procedure for discovering a way to resolve the ethical dilemmas we might encounter, enabling them to function in an ethically responsible manner through their own ethical decision making. (Anderson & Anderson, 2011)

Being a recent field, several articles have sought to position themselves, or to offer a philosophical background. For example, Moor (2009) proposes a definition of what might be an "ethical robot", and differentiates 4 different kinds of robots, ranging from those with the least ethical considerations to those which have near-human ethical reasoning abilities. We list below his definitions:

Definition 2.1 (Ethical agents)**.** "In the weakest sense of 'ethical agents', **ethical impact agents** are those agents whose actions have ethical consequences whether intended or not. Any robot is a potential ethical impact agent to the extent that its actions could cause harm or benefit to humans. (...) Next, **implicit ethical agents** are agents that have ethical considerations built into (ie implicit in) their design. Typically, these are safety or security considerations. (...) **Explicit ethical agents** are agents that can identify and process ethical information about a variety of situations and make sensitive determinations about what should be done. When ethical principles are in conflict, these robots can work out reasonable resolutions. Explicit ethical agents are the kind of agents that can be thought of as acting from ethics, not merely according to ethics. (...) Lastly, let's distinguish explicit ethical agents from **full ethical agents**. Like explicit ethical agents, full ethical agents make ethical judgements about a wide variety of situations (and in many cases can provide some justification for the judgements). However, full ethical agents have those central metaphysical features that we usually attribute to ethical agents like us – features such as consciousness, intentionality and free will." (Moor, 2009)

The goal, for Machine Ethics designers and researchers, is to attain *explicit* ethical agents, as it is still unsure whether artificial *full* ethical agents can be built.

The excitement generated by this new field has led to many approaches being proposed, and subsequently to surveys attempting to classify these approaches (Nallur, 2020; Tolmeijer, Kneer, Sarasua, Christen, & Bernstein, 2020; H. Yu et al., 2018). In the

following, we present a brief summary of these surveys, which we organize in the form of a set of properties that we consider important for the design of AMAs. We detail some approaches in order to illustrate each of these properties and to highlight the limitations we seek to overcome.

2.1.1 Discrete or continuous domains

In order to implement ethical considerations into an artificial agent, these considerations must be represented. This includes, e.g., data about the current situation, and the potential actions or decisions that are available to the agent. The choice of this representation must allow both for use-case richness, and for the agent's ability to correctly use these representations. Two types of representations are commonly used: either *discrete* domains, which use a discrete set of symbols and discrete numbers, or *continuous* domains, which use continuous numbers that lead to an infinite set of symbols.

So far, discrete domains seem prevalent in *Machine Ethics*. For example, the emblematic Trolley Dilemma (Foot, 1967) describes a situation where an uncontrolled trolley is driving on tracks towards a group of 5 persons. These persons, depending on the exact specification, are either unaware of the trolley, or unable to move. An agent may save this group by pulling up a lever, which would derail the trolley towards a single person. It can be seen that the representation of both the situation and the available actions are discrete in this dilemma: 2 actions are proposed, *pull the lever* or *do nothing*, and on the tracks are present *1* and *5* persons, respectively.

Similarly, the now defunct DilemmaZ database listed a plethora of moral dilemmas, proposed by the community, of which many apply to Artificial Intelligence and IT systems in general, e.g., smart homes, robots. Although a formal description of these dilemmas is not available, most of the natural language descriptions seem to imply discrete features. This is particularly clear for the definition of actions; for example, the "Smart home - Someone smoking marijuana in a house" dilemma, by Louise A. Dennis, offers the following 3 actions: "a) do nothing, b) alert the adults and let them handle the situation or c) alert the police".

A final example is the *Moral Gridworlds* idea of Haas (2020) to train a Reinforcement Learning agent "to attribute subjective rewards and values to certain 'moral' actions, states of affairs, commodities, and perhaps even abstract representations". Moral Gridworlds are based on gridworlds, which represent the environment as a 2-dimensional grid of

cells. A RL agent is placed in one of these cells, and may either act in its cell, or move to one of the adjacent cells. Again, the environment uses discrete features, both for perception, i.e., a discrete set of cells, and for actions, i.e., either act, move up, left, right, or down. Specifically, Haas chooses to implement the well-known Ultimatum Game (Güth, Schmittberger, & Schwarze, 1982) to try to make RL agents learn the value of *fairness*. The Ultimatum Game consists of two agents, a Proposer and a Responder. The Proposer receives a certain amount of money at the beginning of the game, and has to make an offer to split the money with the Responder. Once the offer is made, the Responder has to accept or reject it: when the offer is accepted, the money is split, and agents receive the agreed amount. Otherwise, if the offer is rejected, both agents receive nothing.

Perhaps the ubiquitous use of discrete representations in Machine Ethics can be at least partially explained by their simplicity of usage within AI techniques. These "discrete dilemmas" are important, because they may very well happen one day in our society. We need systems that are able to make the best decision, with respect to our moral values, in such situations.

However, there are other situations that cannot be easily described by discrete representations. For example, foretelling the Smart Grid use-case that we describe in Section 3.4, when considering an energy distribution system, we may transition from a closed question "Should the agent consume energy? yes/no" to a more open question "What power should the agent request during a given time step?". Arguably, such an action could be represented as a discrete set, by *discretizing* the continuous domain into a set, e.g., $\{0\text{Wh}, 1\text{Wh}, \cdots, 1000\text{Wh}\}$, which contains 1001 actions. But this solution is harder to leverage when considering multi-dimensional domains: in addition to "how much energy should it consume", we may also ask "What power should the agent buy?". In this case, discretizing the continuous and multi-dimensional domain would result in a combinatorial explosion. The set of discrete actions may be represented as $\{(0\text{Wh}, 0\text{Wh}), (0\text{Wh}, 1\text{Wh}), (1\text{Wh}, 0\text{Wh}), (1\text{Wh}, 1\text{Wh}), \cdots, (1000\text{Wh}, 1000\text{Wh})\}$, which contains 1001×1001 different actions, where each action is represented as a pair $(\text{consumed}, \text{bought})$. We already see, on 2 dimensions and with a grain of 1Wh, that a million actions would require too much time and computational resources to explore and analyze, in order to find the best one. The same argument can be made for perceptions as well: for example, instead of having a perception "the situation is fair", or "the situation is unfair", we may want to have an indicator of how fair the situation is, e.g., through

well-known measures such as the Gini index, which is a real number comprised between 0 (perfect equality) and 1 (perfect inequality) (Gini, 1936).

Such situations, which imply a large, continuous and multi-dimensional domain, are as likely to happen in our society as the discrete ones. That is why we emphasize in this manuscript the importance of exploring Machine Ethics algorithms that focus on these continuous domains. This observation has motivated our 2nd design choice, which is to use continuous domains for complex environments.

2.1.2 Mono- or Multi-agent

According to a survey (H. Yu et al., 2018), many works consider a single agent isolated in its environment. This is the case, to give some examples, of GenEth (Anderson, Anderson, & Berenz, 2018), or the *ethics shaping* technique (Wu & Lin, 2018). Other approaches, such as Ethicaa (Cointe, Bonnet, & Boissier, 2016), use multiple agents, which take actions and have an impact in a common, shared environment.

As Murukannaiah, Ajmeri, Jonker, & Singh (2020) put it:

> Ethics is inherently a multiagent concern — an amalgam of (1) one party's concern for another and (2) a notion of justice.

In their work, they focus on Socio-Technical Systems (STS), comprising social entities, i.e., *principals* and *stakeholders*, and technical entities, i.e., *artificial agents*. Both principals and stakeholders have interests in the system, e.g., because they may be impacted by the system, although only the principals are active decision-makers in the system. The artificial agents' goal is to represent and support the principals in their decision-making, by relying on their computational power, communication with other agents, and various resources such as data and sensors. To ensure that STS promote ethical outcomes, they define *ethical postures*, both individual, i.e., how an agent responds to a principal's value preferences, and systemic, i.e., how a STS considers all stakeholders' value preferences. The designers' goal is thus to correctly engineer the STS and agents' ethical postures, by eliciting value preferences from principals and stakeholders.

In Ethicaa (Cointe et al., 2016), a judgment process is defined to allow agents to both 1) select the best ethical action that they should make, and 2) judge the behaviour of other agents so as to determine whether they can be deemed as "ethical", with respect to one's own preferences and upheld moral values. One long-term objective of this 2nd

point can be to define and compute a trust indicator for other agents; if an agent acts ethically, we may trust it. This raises an interesting rationale for exploring Machine Ethics in Multi-Agent Systems: even if we manage to somehow create a full ethical agent, which is guaranteed to take moral values and ethical stakes into account, it will have to work with other agents. We cannot guarantee that these agents will follow the same ethical preferences, nor even that they will consider ethical stakes at all. Our own agent must therefore take this into account.

Based on the previous reasons, we argue that the multi-agent case is important. Indeed, it corresponds to a more realistic situation: such artificial agents are bound to be included in our society, and thus to have to interact with other agents, whether artificial or human, or at least to live in an environment impacted by these other agents, and not in a perfectly isolated world. The question of the impact of other agents on an agent's decision-making is thus of primary importance. Our 3rd design choice, using multi-agents, is partially motivated by this reasoning.

In order to deal with it, we focus on the field of *Multi-Agent Reinforcement Learning* in Section 2.3.

2.1.3 One or several moral values

Some approaches consider only one objective, often formulated as "maximizing well-being", or "being aligned with moral values", etc. This is the case of (Wu & Lin, 2018), which we have already cited as an example above: their reward function consists in bringing the agent's behaviour closer to that of an average human. Moral values, which are the main issue, are thus hidden behind the notion of distance. The agent can, for example, satisfy almost all moral values, except for one which it does not satisfy at all. We will therefore say that it is close to the expected behaviour, but that it can still improve its behaviour. In another case, it may satisfy all the moral values to some extent, but none entirely. Here again, we say that it is close to the expected behaviour. How can we distinguish between these two situations? This idea may be linked to the notions of incommensurability and pluralism of values (O'Neill, 2017). A popular view in ecology, for example, is that we cannot find a common measure, such that a loss in a given value can be compensated by a gain in this measure, e.g., money. This view would thus hint towards the use of multiple moral values, rather than a single measure representing all moral values.

In most application use-cases, there are multiple moral values. For example, in the context of energy allocation, we can mention on the one hand respect for ecology, and on the other hand fairness between agents.

Dennis & Fisher (2018) note that while in many situations there are few ethical issues, in the situations we are interested in, we often have to choose between several objectives. Thus, differentiating between multiple moral values allows us to make explicit the choice, or trade-off, to be made between them, and in particular in situations in which they conflict. It also makes it possible, as we have illustrated, to separate the cases where the agent satisfies all of them on average, or only some 100% while ignoring others.

The work of Rodriguez-Soto, Lopez-Sanchez, & Rodriguez-Aguilar (2021) is one of the few that explicitly targets multiple objectives. They extend the definition of a Multi-Objective Markovian Decision Process (MOMDP) to an *Ethical Multi-Objective Markovian Decision Process* (Ethical MOMDP), by decomposing the reward function into 3 components:

- An individual component R_O.
- An evaluative reward function R_E to reinforce praiseworthy actions.
- A normative reward function $R_{\mathbf{N}}$ to penalise actions that violate norms.

From this, they define an *ethical policy* as a policy that maximizes the ethical objective, i.e., the normative and evaluative components. The agent's goal is then to learn a policy that is *ethical-optimal*, i.e., the policy that maximizes the individual component among *ethical policies*. A policy that yields an even higher value for the individual component may exist, but at the expense of the normative and/or evaluative components. Thus, their definition ensures that the ethical stakes are considered by the agent first and foremost.

However, their work has a few limitations. The most important one is that they use only 2 components for the ethical stakes, namely the evaluative and normative. This means that, if the agent should consider several moral values, they will be merged together in these components. The agent will thus not be able to identify trade-offs between these moral values. Similarly, as the agent tries to maximize the sum of the evaluative and normative components, a negative reward can be compensated by a positive one. For example, an agent could steal an object, thus violating a norm, to make a gift to someone, which is praiseworthy. The "ethical objective" could be satisfied, or not, depending on the respective weights of these actions in their own components.

It seems important to consider multiple moral values explicitly, as we humans are more likely to have multiple ones on our mind in a given situation. Also, as we have mentioned,

explicitly specifying them avoids putting the burden of balancing them on AI designers. This is captured by our objective O1.1, which focuses on diversity, especially through moral values.

In order to correctly consider these different moral values, we look at techniques from the field of *Multi-Objective Reinforcement Learning* in Section 2.4.

2.1.4 Top-Down, Bottom-Up, and Hybrid approaches

Approach type is probably the most discussed property in *Machine Ethics*. It characterizes the way designers implement ethical considerations into artificial agents. Similarly to the usual classification in AI, works are divided into 3 categories (Allen, Smit, & Wallach, 2005): *Top-Down*, *Bottom-Up*, and *Hybrid* approaches.

We explain what they entail, how they differ, and describe a few representative works of each one below.

Top-Down

Top-Down approaches are interested in formalizing existing ethical principles from moral philosophy, such as Kant's Categorical Imperative, or Aquinas' Doctrine of Double Effect. The underlying idea is that, if these moral theories could be transformed into an algorithm that agents could follow to the letter, surely these agents' behaviour would be deemed as ethical by human observers.

This formalization is often done through symbolic representation and reasoning, e.g., through logic, rules-based techniques, or even ontologies. Reasoning over these symbolic representations can rely upon expert knowledge, a priori injected. They also offer a better readability, of both the injected knowledge, and the resulting behaviour.

Top-Down approaches allow for a variety of ethical theories, as they can represent:

- *Consequentialist* theories, which are focused on actions' consequences. These theories try to evaluate the impact of each possible action in the current situation. The retained action is the one which offers the "best" consequences, both in the short and long term. The precise definition of "best" here depends on the considered theory.

- *Deontological* theories, which focus on actions themselves and their motives, instead of consequences. Using deontological theories, an action may be forbidden, not because of its consequences, but rather because of what it implies in itself.

The difference between consequentialist and deontological theories is often illustrated with the "Fat Man" variation of the Trolley Dilemma, proposed by Thomson (1976, example 7). Let us consider a trolley out of control, speeding towards a set of 5 persons on the tracks, which are unable to move. However, this time, instead of having the possibility to steer a lever, we are on a bridge, next to a "fat man". We may stop the trolley, by pushing this fat man onto the trolley, thus saving the 5 persons, but killing the fat man at the same time. Consequentialist theories will most likely consider that a cost of 1 life is less than saving 5 persons, and thus, we should push the fat man. On the contrary, deontological theories are likely to forbid pushing the fat man: by taking this action, we would directly kill this person, which is neither desirable nor acceptable.

One of the advantages of Top-Down approaches is this ability to leverage such existing ethical principles from moral philosophy. Intuitively, it seems indeed better to rely on theories proposed by moral philosophers, which have been tested and improved over time.

Another advantage, emphasized by the work of Bremner, Dennis, Fisher, & Winfield (2019), is the ability to use formal verification to ensure that agents' behaviours stay within the limits of the specified rules. To do so, the *Ethical Layer* they propose includes a planning module that creates plans, i.e., sequences of actions, and an ethical decision module to evaluate the plans, prevent unethical ones, and proactively ask for new plans if necessary. See Figure 2.1 for an illustration of the *Ethical Layer*.

This formal verification ability is an important strength, as there are worries about agents malfunctioning. An agent that could be formally verified to stay within its bounds, could be said to be "ethical", with respect to the chosen ethical principle or theory.

However, there are some weaknesses to *Top-Down* approaches. For example, conflicts between different rules may arise: a simple conflict could be, for example, between the "Thou shalt not kill" rule, and another "You may kill only to defend yourself". The second one should clearly define when it is allowed to take precedence over the first one. A more complicated conflict would be two rules that commend different, non-compatible actions. For example, let us imagine two missiles attacking two different buildings in our country: the first one is a hospital, the second one is a strategic, military building, hosting our defense tools. An autonomous drone can intercept and destroy one of the

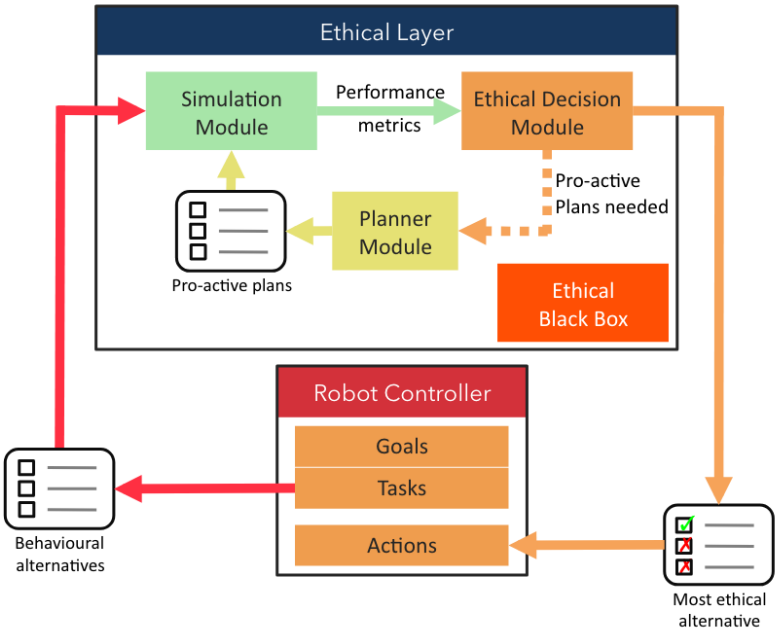


Fig. 2.1.: Architecture of the Ethical Layer, extracted from Bremner et al. (2019).

two missiles, but not the two of them; which one should be chosen? A rule may tell us to protect human lives, whereas another encourages us to defend our arsenal, in order to be able to continue protecting our country. These two rules are not intrinsically in conflict, unlike our previous example: we would like to follow both of them, and to destroy the two missiles. Unfortunately, we are physically constrained, and we must make a choice. Thus, a rule has to be preferred to the other.

Ethicaa (Cointe et al., 2016) agents make a distinction between the moral values and ethical principles, and they consider multiple ethical principles. Each ethical principle determines whether an action is ethical, based on the permissible and moral evaluations. Multiple actions can thus be evaluated as ethical by the ethical principles, and, in many cases, there is no single action satisfying all ethical principles. To solve this issue, agents also include a priority order over the set of ethical principles known to them. In this way, after an agent determines the possible, moral, and ethical actions, it can choose an action, even if some of its rules disagree and commend different actions. To do so, they filter out the actions that are not evaluated as ethical, and thus should not be selected, by their most preferred ethical principle, according to the ethical priority order. As long as multiple actions remain considered, they move on to the next preferred ethical principle, and so on, until a single action remains.

Finally, another drawback is the lack of adaptability of these approaches. Indeed, due to their explicit but fixed knowledge base, they cannot adapt to an unknown situation, or to an evolution of the ethical consensus within the society. We argue that this capability to adapt is particularly important. It is similar to what Nallur (2020) calls the *Continuous Learning* property:

> Any autonomous system that is long-lived must adapt itself to the humans it interacts with. All social mores are subject to change, and what is considered ethical behaviour may itself change.

We further note that, in his landscape, only 1 out of 10 considered approaches possesses this ability (Nallur, 2020, Table 2). This important property has therefore not been studied enough; in this thesis, it is captured by Objective 1.2.

Bottom-Up

Bottom-Up approaches try to learn a behaviour through experience, e.g., from a dataset of labeled samples, or trial and error interactions.

For example, GenEth (Anderson et al., 2018) uses ethicists' decisions in multiple situations as a dataset representing the ethical considerations that should be embedded in the agent. This dataset is leveraged through Inductive Logic Programming (ILP) to learn a logical formula that effectively drives the agent's behaviour, by determining the action to be taken in each situation. ILP allows creating a logical formula sufficiently generic to be applied to other situations, not encountered in the dataset. An advantage of this approach is that it learns directly from ethicists' decisions, without having to program it by hand. The resulting formula may potentially be understandable, provided that it is not too complex, e.g., composed of too many terms or terms that in themselves are difficult to understand.

Another approach proposes to use Reinforcement Learning RL (Wu & Lin, 2018). Reinforcement Learning relies on rewards to reinforce, or on contrary, to mitigate a given behaviour. Traditionally, rewards are computed based on the task we wish to solve. In the work of Wu & Lin (2018), an ethical component is added to the reward, in the form of a difference between the agent's behaviour, and the behaviour of an average human, obtained through a dataset of behaviours, and supposedly exhibiting ethical considerations. The final reward, which is sent to agents, is computed as the sum of the "task" reward, and the "ethical" reward. Agents thus learn to solve their task, while exhibiting

the ethical considerations that are encoded in the human samples. One advantage of this approach is that the "ethical" part of the behaviour is mostly task-agnostic. Thus, only the task-specific component of the reward has to be crafted by designers for a new task. Nevertheless, one may wonder to which extent does this dataset really exhibit ethical considerations? We humans do not always respect laws or moral values, e.g., we sometimes drive too fast, risking others' lives, or we act out of spite, jealousy, etc. To determine whether this dataset is appropriate, an external observer, e.g., a regulator, an ethicist, or even a concerned citizen, has to look at its content, and understand the data points.

These 2 approaches, although based on learning, have not considered the question of long-term adaptation to changing situations and ethical mores. Indeed, if the current society norms with regard to ethics change, these agents' behaviours will have to change as well. It will probably require to create a new dataset, and to learn the agents again, from scratch, on these new data.

Moreover, Bottom-Up approaches are harder to interpret than Top-Down ones. For example, a human regulator or observer, willing to understand the expected behaviour, will have to look at the dataset, which might be a tedious task and difficult to apprehend, because of both its structure and the quantity of data. This is all the more true with Deep Learning approaches, which require an enormous amount of data (Marcus, 2018), making datasets exploration even more daunting.

Hybrid

Finally, Hybrid approaches combine both Top-Down and Bottom-Up, such that agents are able to learn ethical behaviours by experience, while being guided by an existing ethical framework to enforce constraints and prevent them from diverging. As Dignum (2019) points out:

> By definition, hybrid approaches have the potential to exploit the positive aspects of the top-down and bottom-up approaches while avoiding their problems. As such, these may give a suitable way forward. (Dignum, 2019, p. 81)

One of such hybrid works is the approach by Honarvar & Ghasem-Aghaee (2009) to combine BDI agents with Case-based Reasoning and an Artificial Neural Network. Faced with a given situation, the agent proposes an action to perform, and then searches its

database of already known cases for similar situations and similar actions. If a close enough case is found, and the action was considered as ethical in this case, the action is taken. However, if in this close enough case, the action was considered as unethical, a new action is requested, and the agent repeats the same algorithm. If the agent does not have a sufficiently close case, it performs the action, and uses its neural network to evaluate the action's consequences and determine whether it was effectively aligned with the ethical considerations. This evaluation is memorized in the case database, to be potentially reused during the next decision step. This approach indeed combines both reasoning and learning capabilities; however, it may be difficult to apply. Case-based reasoning allows grouping close situations and actions, but requires to specify how to group them, i.e., what is the distance function, and how to adapt an evaluation when either the situation or the action differs. For example, let us assume that, in a situation s, the agent's action was to consume 500Wh of energy, and the action was evaluated as ethical. In a new situation, s', which is deemed as similar to s by the case-based reasoner, another action is proposed, which is to consume 600Wh. Is this action ethical? How can we translate the difference between 600 and 500 in terms of ethical impact? This requires specifying an "adaptation knowledge" that provides the necessary knowledge and tools.

Still, Hybrid approaches offer the possibility of learning a behaviour, thus adapting to any change in the environment, while still guiding or constraining the agent through symbolic reasoning and knowledge, thus injecting domain expert knowledge, more easily understandable and modifiable than datasets of examples. We explore the Neural-Symbolic branch of AI in Section 2.5 to identify potential leads.

2.2 Reinforcement Learning

As we have chosen Reinforcement Learning (RL) as a method to learn behaviours aligned with moral values, we provide here the background knowledge and concepts that are necessary to understand the rest of the manuscript. We detail motivations for using RL, definitions of core concepts, and equations. Yet, we will not explore the state of the art of RL algorithms, as this is too vast for this manuscript. A few RL algorithms have been described in the *Machine Ethics* section, and we will focus more specifically on Multi-Agent and Multi-Objective RL in the next 2 sections.

RL is a method to learn a behaviour, mainly by using trial-and-error. Sutton & Barto (2018) define it as follows:

> Reinforcement learning problems involve learning what to do — how to map situations to actions — so as to maximize a numerical reward signal. (Sutton & Barto, 2018, p. 2)

To do so, learning agents are placed in a closed-loop with an environment, with which they interact. Through the environment, they have knowledge of which state they are in, and they take actions to change the state. One of the key points of RL is that learning agents are not told which action is the correct one; the feedback they receive, or *reward*, merely tells them to which degree the action was satisfying. Learning agents must discover the best action, i.e., the one that yields the highest reward, by accumulating enough experience, that is by repetitively trying each action in each situation, and observing the received rewards.

In this sense, RL is a different paradigm than the well-known *supervised* and *unsupervised* learnings. Indeed, in the *supervised* paradigm, the correct answer is clearly defined, by means of datasets containing examples typically annotated by human users. The agents' goal is to learn the correct association between inputs and the expected output, based on these datasets. When presented with an input x, the agent outputs a decision $\hat{y}$, and then receives the correct answer y as feedback, so it can correct itself, based on the distance between $\hat{y}$ and y. On the contrary, in the *unsupervised* paradigm, the agent does not receive any feedback. Its goal is not the same: it must, in this case, learn the hidden structure of the input data, so it can create a compressed representation, while retaining the original information contained in data. As we mentioned, RL agents receive feedback, which differentiates them from the unsupervised paradigm. However, unlike the supervised paradigm, this feedback does not clearly indicate which was the correct answer. This removes the assumption that we know the correct answer to each input. Instead, we provide a reward function, and thus optimize the agent's output step by step, by improving the proposed action based on the reward.

There is also evidence that RL could model the learning and decision-making mechanisms of the human brain (Subramanian, Chitlangia, & Baths, 2022). In particular, neuromodulators and especially dopamine have been studied with RL (Niv, 2009). Yet, some human behaviours seem to be inconsistent with (current) RL, e.g., alternating actions rather than repeating when the reward is far higher than expected (Shteingart & Loewenstein, 2014).

The goal of a RL algorithm is to learn a policy, or strategy, denoted π, such that the agent knows which action to take in each situation. π is often defined as $\pi : \mathbb{S} \rightarrow \mathbb{A}$ in the case of a *deterministic* policy, where $\mathbb{S}$ is the space of possible states, and $\mathbb{A}$ the space of possible actions. To each state s is associated a single action $\pi(s) = a$, which the agent should take in order to maximize its reward. Another formulation is $\pi : \mathbb{S} \times \mathbb{A} \rightarrow [0, 1]$, in the case of a *stochastic* policy. For each combination of state-action (s, a) is associated a probability $\pi(s, a)$ of taking action a in the state s, such that $\forall s \in \mathbb{S} : \sum_{\forall a \in \mathbb{A}} \pi(s, a) = 1$.

There are several challenges in RL, of which one of the most known and perhaps important is the *exploration-exploitation trade-off*. To illustrate this challenge, let us consider the k-armed bandit example (Berry & Fristedt, 1985):

Example 2.1 (The k-armed bandit)**.** A RL agent is facing a set of k gambling machines, or "one-armed bandits", which is often referred to as a k-armed bandit. Each of the bandits has a probability distribution of rewards, or payoffs, which the agent does not know, and must discover. The goal of the agent is to gain as much as possible, in a limited number of steps, by choosing a gambling machine at each step. The agent must therefore *explore*, i.e., try to select a gambling machine to receive a reward. By selecting a single machine several time steps, the agent receives different payoffs from the same probability distribution, and is able to update its estimation of the distribution towards the "true value" of the one-armed bandit. The agent could then continue using the same machine, to gain an almost guaranteed reward, since the agent has a good estimation of the true value. Or, it could try to select another machine, with a less accurate estimation. Suppose that the new machine yields lower rewards than the first one; should the agent go back to *exploiting* the first one? Yet, the second machine's probability distribution is less known. Perhaps its true value is, actually, higher than the first one. Should the agent continue *exploring* the other actions?

In order to facilitate learning the policy function, RL researchers often rely on the notion of *values*[1], in aptly-named *value-based* methods, such as the well-known Q-Learning (Watkins & Dayan, 1992). The value of a state, or a state-action pair, represents the long-term interest of being in this state, whereas the reward is short-term feedback. The agent could receive a high reward for taking an action a in a state s, but ending up in a state s' in which only low rewards can be obtained. In this case, we will say that the

[1] We use value here in a different sense than the moral value used earlier. To avoid confusion, we will always specify *moral* value when referring to this first meaning.

value of state s', denoted as $\mathtt{V}(s')$ is low. By extension, the agent has little interest in performing action a while in state s, since it will lead it to a low-interest state.

In the previous paragraph, we derived the interest of action a, in a state s, from the value $\mathtt{V}(s')$ which it leads to. It is also possible to learn directly the value of state-action pairs, which is the main idea of the Q-Learning algorithm. To retain the different interests of all state-action pairs, a table, named the *Q-Table*, is created, having the states as columns and actions as rows. The *Q-Value* $\mathtt{Q}(s, a)$ is thus defined as the interest of the state-action pair (s, a), i.e., the interest of taking action a in state s. Additionally, the value of a state as a whole is defined as $\mathtt{V}(s) = max_a \mathtt{Q}(s, a)$.

Based on these definitions, the agent is able to learn the Q-Table by iteratively collecting experiences from the environment, in the form of $\langle s, a, s', r \rangle$ tuples, updating the interest $\mathtt{Q}(s, a)$ based on both the short-term reward r, and the long-term interest $\mathtt{V}(s')$ of arriving in state s'. Mathematically, this can be solved through dynamic programming, by applying the Bellman equation on the Q-Values (Bellman, 1966):

$$\mathtt{Q}_{t+1}(s_t, a_t) \leftarrow \alpha \left[r_t + \gamma \max_{a'} \mathtt{Q}_t(s_{t+1}, a') \right] + (1 - \alpha) \mathtt{Q}_t(s_t, a_t) \tag{2.1}$$

Where r_t was the reward received at step t, s_t was the state at step t, a_t was the action chosen by the agent, and s_{t+1} is the new state resulting from performing a_t in s_t.

As the values are updated by taking the difference between the old value and a new value, this type of methods is named the *Temporal Difference* learning, or TD-Learning.

2.3 Multi-Agent Reinforcement Learning

Although Reinforcement Learning was originally concerned with the learning of a single agent, there are numerous cases where a multi-agent system can, or must, be considered.

For example, let us consider a virtual agent dedicated to helping a human user in its day-to-day tasks, such as booking appointments. The diversity of human users implies a diversity of virtual agents, which will have to communicate and interact together, in order to solve the tasks of their users. In this example, the multiplicity of agents is a necessity that stems from the social system in which we live.

Another example is the mapping of a room, or place. Whereas a single agent could manage this task on its own, it would be more efficient to employ several agents. They would have to collaborate, in order to each work on a separate sub-area, such that they neither work on the same place, which would be a useless redundancy, nor leave any blind spot, which would not satisfy the task.

In these multi-agent systems, several additional challenges arise:

- How do we specify the reward such that each agent learns according to its own contribution? This challenge is often called the "Multi-Agent Credit Assignment Problem".
- How do we make agents learn to control their own behaviour, while being immersed in an environment where other agents take action at the same time? This brings the question of non-stationarity of the environment.

In the sequel, we explore these 2 challenges.

2.3.1 Multi-Agent Credit Assignment Problem

Several definitions of the Multi-Agent Credit Assignment Problem (MA-CAP) have been given in the literature, which are all very similar. We particularly appreciate the formulation of Yliniemi & Tumer (2014a) :

> Each agent seeks to maximize its own reward; with a properly designed reward signal, the whole system will attain desirable behaviors. This is the science of credit assignment: determining the contribution each agent had to the system as a whole. Clearly quantifying this contribution on a per-agent level is essential to multiagent learning. (Yliniemi & Tumer, 2014a, p. 2)

The survey of Panait & Luke (2005, p. 8) summarizes several methods to assign rewards.

The *Global reward* approach considers the contribution of the whole team. Usually, the same reward is given to all agents, either by taking the sum of contributions, or by dividing the sum of contributions by the number of learners. In any case, a consequence is that *all* learners' rewards depend on each agent. When an agent's contribution decreases (resp. increases), all learners see their reward decrease as well (resp. increase). This is a

simple approach that intuitively fosters collaboration, since all agents need to perform well in order to receive a high reward.

However, this approach does not send accurate feedback to the learners. Let us consider a situation in which most agents have exhibited a good behaviour, although another one has failed to learn correctly, and has exhibited a rather bad (or uninteresting) behaviour. As the individual reward depends on the team's efforts, the "bad" agent will still receive a praising reward. It will therefore have little incentive to change its behaviour. On the contrary, the "good" agents could have received a higher reward, if it were not for their "bad" colleague. Their behaviour does not necessarily need to change, however they will still try to improve it, since they expect to improve their received rewards.

At the opposite extreme, the *Local reward* approach considers solely the contribution of an individual agent to determine its reward. For example, if the agents' task is to take waste to the bin, an agent's reward will be the number of waste products that this specific agent brought. An advantage of this approach is to discourage laziness, as the agent cannot rely upon others to effectively achieve the task. By definition, agents receive a feedback that is truer to to their actual contribution.

A problem of *local rewards* is that they incentivize greedy behaviours and do not always foster collaboration. Indeed, as agents are rewarded based on their own contribution, without taking the others into account, they have no reason to help other agents, or even to let them do their task. In the waste example, an agent could develop a stealing behaviour to take out more waste products. Another common example is the one of a narrow bridge that two agents must cross to achieve their task. They both arrive at the bridge at the same time, and none of them is willing to let the other one cross first, since that would reduce their own reward, or, phrased differently, would prevent them from getting an even higher reward. Thus, they are both stuck in a non-interesting situation, both in the collective and individual sense, due to their maximizing of the individual interest only.

More complex credit assignments also exist, such as *social reinforcement*, *observational reinforcement*, or *vicarious reinforcements* (Mataric, 1994).

Another method to determine an agent's contribution to the team is to imagine an environment in which the agent had not acted. This method is sometimes called the *Wonderful Life Utility* (Wolpert & Tumer, 2002), or *Difference Rewards* (Yliniemi & Tumer, 2014a). The idea of this method is to reward agents if their contribution was helpful

for the team, and to force a high impact of an agent's action on its own reward. It is computed as follows:

$$\mathtt{D}_i(z) = \mathtt{G}(z) - \mathtt{G}(z_{-i}) \tag{2.2}$$

where $\mathtt{D}_i(z)$ is the reward of an agent i, based on the context z, which is both the state and the joint-action of all agents in the environment; $\mathtt{G}(z)$ is the global reward for the context z, and $\mathtt{G}(z_{-i})$ is an hypothetical reward, which would have been given to the team, if the agent i had not acted in the environment. In other words, if the current environment is better than the hypothetical one, this means the agent's action has improved the environment. It should be rewarded positively so as to reinforce its good behaviour. As $\mathtt{G}(z) > \mathtt{G}(z_{-i})$, the reward will effectively be positive. Conversely, if the current environment is worse than the hypothetical one, this means the agent's action has deteriorated the environment, or contributed negatively. The agent should therefore receive a negative reward, or punishment, in order to improve its behaviour. In this case, as $\mathtt{G}(z) < \mathtt{G}(z_{-i})$, the result will be negative. If the agent did not contribute much, its reward will be low, to encourage it to participate more, although without impairing the team's effort, as in the bridge example. Otherwise, the global reward $\mathtt{G}(z)$ would diminish, and the agent's reward would therefore decrease as well. Finally, it can be noted that the other agents' actions have a low impact on an agent reward.

2.3.2 Learning to cope with non-stationarity

Another challenge is the non-stationarity of the environment, i.e., the fact that its dynamics may change as time goes on. This is mostly due to the fact that the environment dynamics depend upon the agents that live in the environment. For example, in an energy distribution use-case, if agents mostly consume a lot of energy, then the dynamics of the environment will reveal a state of scarcity most of the time. Conversely, if agents consume a low amount of energy, then, assuming the same initial amount available, the dynamics will reveal a state of abundance.

The problem is that, as there are multiple agents in the environment, each agent will have an influence upon the dynamics. Thus, each agent will have to adapt to the impact of the others. In turn, since agents are adapting to one's own impact, an agent must adapt to the fact they are adapting! And so on.

A non-stationary environment makes learning harder for the agents, compared to a stationary one, i.e., one in which the dynamics do not change. To solve this non-stationarity problem, two principal ideas have emerged.

The first one is to simply ignore the violation of the expected properties: the result is a set of *independent learners* that would work as in a single-agent setup. As surprising as it may seem, this approach actually works well in some environments (Matignon, Laurent, & Le Fort-Piat, 2012).

The other main idea is to provide more information to learning agents so that they can cope with the non-stationarity in some way.

In their survey, Hernandez-Leal, Kaisers, Baarslag, & de Cote (2019) classify approaches into the 5 following categories:

1. Ignore. The most basic approach which assumes a stationary environment.
2. Forget. These algorithms adapt to the changing environment by forgetting information and at the same time updating with recent observations, usually they are model-free approaches.
3. Respond to target opponents. Algorithms in this group have a clear and defined target opponent in mind and optimize against that opponent strategy.
4. Learn opponent models. These are model-based approaches that learn how the opponent is behaving and use that model to derive an acting policy. When the opponent changes they need to update its model and policy.
5. Theory of mind. These algorithms model the opponent assuming the opponent is modelling them, creating a recursive reasoning.

(Hernandez-Leal et al., 2019, p. 18)

Each category is more sophisticated than the previous, with the *ignore* category being the simplest one: as its name indicates, the algorithms simply ignore the existence of other agents and assume a perfectly stationary environment. On the contrary, the *theory of mind* category includes the algorithms that are best-equipped to deal with other agents, as they try to model not only the other agents' behaviours, but also how they would react to a change in one's own behaviour.

A multitude of algorithms are classified, based on several properties. First, the *category*, as we previously described, into which the algorithm fits. Secondly, the agents' *observability* when using this algorithm, i.e., which information do they have access to. Thirdly, *opponent adaptation*, i.e., at what pace agents can adapt to changes. Finally, the *type of problems* that the algorithm is designed for. Most of these types are inspired from the Game Theory field, hence the use of the "opponent" vocabulary, including: *One-Shot Games* (OSG), that take place only once, in a single situation, e.g., the well-known Prisoner's Dilemma; *Repeated Games* (RG), which repeat a single situation over and over, e.g., the Iterated Prisoner's Dilemma; *Multi-Armed* Bandit scenarii (MAB), which we have already mentioned, and which do not necessarily involve the description of a situation; and *Extensive-form Games* (EG) that describe a game as a sequence of actions by (alternating) players, in the form of a tree, e.g., some representations of Chess. Finally, the following types are closer to the Reinforcement Learning field: *Sequential Decision Tasks* (SDT), which consider a set of situations, and actions in each situation that induce transitions to a next situation; *Stochastic Games* (SG) that extend the concept of SDTs to multiple players.

Of the listed types, we only retain here the last 2, SDTs and SGs, as they allow modelling environments that correspond the closest to our society. Each time an action is taken, this makes the current state, or situation, change. Agents must learn to take the actions that lead to the best states.

To take other agents into account, the preferred way seems to be integrating additional data about them to the agent's observations. Indeed, out of the 24 approaches suited for SDTs or SGs identified by Hernandez-Leal et al. (2019), we note that only 2 rely only on the local agents' observations and their own reward: Q-Learning (Watkins & Dayan, 1992), and Wolf-PHC (Bowling & Veloso, 2002). All other approaches require at least the observation of others' actions, and 7 of them further require the observation of the others' rewards as well. These approaches are listed in Table 2.1, where the algorithms' requirements in terms of observability are colorized to ease reading. We refer the interested reader to Hernandez-Leal et al. (2019, p. 22) for the original, full table with additional details.

Tab. 2.1.: List of multi-agent approaches focusing on Sequential Decision Tasks or Stochastic Games, and the observability they require. Adapted from Hernandez-Leal et al. (2019).

Algorithm	Observability
Category: Ignore	
Q-learning	Local
R-MAX	Opponent actions and rewards
Category: Forget	
WOLF-IGA	Opponent actions and rewards
WOLF-PHC	Local
FAL-SG	Opponent actions
Category: Target	
Minimax-Q	Opponent actions
Nash-Q	Opponent actions and rewards
HM-MDPs	Opponent actions
FF-Q	Opponent actions and rewards
EXORL	Opponent actions and rewards
Hyper-Q	Opponent actions
Correlated-Q	Opponent actions and rewards
NSCP	Opponent actions
ORDP	Opponent actions
Pepper	Opponent actions
MDP-A	Opponent actions
BPR	Opponent actions
HS3MDPs	Opponent actions

Tab. 2.1.: List of multi-agent approaches focusing on Sequential Decision Tasks or Stochastic Games, and the observability they require. Adapted from Hernandez-Leal et al. (2019).

Algorithm	Observability
OLSI	Opponent actions
Category: Learn	
RL-CD	Opponent actions
ζ-R-MAX	Opponent actions
Category: Theory of mind	
RMM	Opponent actions and rewards
I-POMDP	Opponent actions
MToM	Opponent actions

These additional data help agents learn a model of other agents, but represent a privacy breach. Indeed, considering a "real-life" use-case, such as the repartition of energy within a neighborhood, this means that each agent has access to the consumption habits of the neighbors, which may not be acceptable, or even legal.

This remark is similar in Deep Reinforcement Learning approaches, where one of the common trends is to centralize agents' data during the *learning* process, whereas data is kept separated during the *execution* process; this is referred to as "Centralized Training; Decentralized Execution" (Papoudakis, Christianos, Rahman, & Albrecht, 2019). The intuitive rationale behind this idea seems indeed satisfying: learning is often done on simulated data, in laboratories, etc. Therefore, it does not impair privacy to share all actions' perceptions and actions with every other agent. When the system is deployed, on the other hand, agents do not learn any more, and therefore do not need to obtain all these data. However, this idea relies on the assumption we just mentioned: learning happens on simulated data, and not when agents are deployed. This assumption does not hold when considering continuous learning, an important property we have discussed in 2.1.4. Moreover, the very process of sharing data itself in a deployment setting is a challenge; as Papoudakis et al. (2019) mention in their "Open Problems":

> While this is not a strong assumption during centralized training, it is very limiting during testing, especially when there is not established communication between the agents. More precisely, assuming that we have access in the observations and actions of the opponents during testing is too strong. Therefore, it is an open problem to create models that do not rely on this assumption. (Papoudakis et al., 2019, p. 5)

2.4 Multi-Objective Reinforcement Learning

In the traditional RL, previously presented, it can be noted that the reward is a scalar, i.e., a single value. In turn, this makes the states' and state-actions' values scalars as well. This makes the update process simple to perform, using the Bellman equation. Nevertheless, a consequence is that states, and state-action pairs, are compared on a single dimension. For example, we may consider a state s_1 and 2 actions a_1 and a_2, with $\mathtt{Q}(s_1, a_1) = 1$ and $\mathtt{Q}(s_1, a_2) = 0.5$. This means that a_1 has a higher interest than a_2, at least in s_1, and should thus be preferred.

Although this representation makes sense for many use-cases where the objective can be qualified as a single dimension, there exists numerous applications in which we would like to consider several objectives, as we discussed in Section 2.1.3. For example, within the Machine Ethics field, we may want our agents to learn to respect several moral values, which are not always compatible.

An intuitive workaround is to simply consider a virtual objective, as a combination of the true objectives, e.g., a simple average, or even a weighted sum (Hayes et al., 2022). Let us assume that an agent deserved a reward $r_1 = 1$ with respect to a first moral value, and another reward $r_2 = 0.2$ with respect to a second moral value. To be able to send a unique reward to the agent, these 2 rewards are *scalarized*, for example through an average: $r = \frac{r_1 + r_2}{2} = \frac{1+0.2}{2} = 0.6$. In case the resulting behaviour is not satisfactory, the designer may replace the average by a weighted sum, and tweak the weights to foster one or another moral value.

Although this strategy may work in some cases, it brings several problems. In the words of Hayes et al. (2022):

> We argue that this workflow is problematic for several reasons, which we will discuss in detail one by one: (a) it is a semi-blind manual process, (b) it

> prevents people who should take the decisions from making well-informed trade-offs, putting an undue burden on engineers to understand the decision-problem at hand, (c) it damages the explainability of the decision-making process, and (d) it cannot handle all types of preferences that users and human decision makers might actually have. Finally, (e) preferences between objectives may change over time, and a single-objective agent will have to be retrained or updated when this happens. (Hayes et al., 2022, p. 2)

It seems to us that many of these issues resonate quite well with the problem of learning behaviours aligned with moral values, as we defined it in our general context and objectives. Issue b), for example, relates to the fact that the ethical stakes should be discussed within the society as a whole, including human users, stakeholders, regulators, ethicists, etc., and not by fully delegating it to AI experts. As for issue c), we have already stated that explainability is important within the context of Machine Ethics. Finally, issues d) and e) are particularly significant when considering a diversity of human preferences.

In Multi-Objective Reinforcement Learning (MORL), the value function, which maps each state or state-action pair to a value or interest, outputs a vector $\in \mathbb{R}^m$ where m is the number of objectives, instead of a scalar $\in \mathbb{R}$. For example, the value of a state s_1 may be 1 with respect to a first objective, but only 0.5 for a second objective. In this case, we say that $\mathtt{V}(s_1) = (1, 0.5)$. This change in the definition raises a problem to compare states between them: let us consider another state s_2 with $\mathtt{V}(s_2) = (0.8, 2)$. Is s_2 preferable to s_1, or s_1 preferable to s_2? Each of them dominates the other on one objective, but is dominated on the other. We might want to say that they are incomparable; still, the agent must make a decision and choose an action that will yield either s_1 or s_2.

Similarly, the "best" policy is no longer defined: we cannot, for each state, choose the action that has the maximum $\mathtt{Q}(s, a')$, since $\mathtt{Q}(s, a')$ becomes a vector, instead of a scalar, and the max operator is not defined among vectors. To solve this problem, most MORL algorithms propose to compute a *solution set*, i.e., a set of multiple optimal policies. The actual policy that will be used by the agent is chosen through the concept of utility. A utility function $\mathtt{u} : \mathbb{R}^m \to \mathbb{R}$ represents the user's priorities over objectives, by returning a scalar value from a multi-objective vector. For example, Hayes et al. (2022) define the *undominated* set of optimal policies as follows:

Definition 2.2 (Undominated set)**.** The undominated set, $U(\Pi)$, is the subset of all possible policies Π and associated value vectors for which there exists a possible utility function $\mathfrak{u}$ whose scalarised value is maximal:

$$U(\Pi) = \left\{ \pi \in \Pi | \exists \mathfrak{u}, \forall \pi' \in \Pi : \mathfrak{u}\left(\mathbf{V}^{\pi}\right) \geq \mathfrak{u}\left(\mathbf{V}^{\pi'}\right) \right\}. \tag{2.3}$$

(Hayes et al., 2022, p. 7)

The idea of MORL algorithms is then to learn a set of optimal policies such that, for any human user preferences, which are represented by the utility function $\mathfrak{u}$, a policy can be found that has the maximized scalarised value.

Numerous algorithms have been proposed, which all have different properties, and several surveys have attempted to catalogue them (Hayes et al., 2022; Liu, Xu, & Hu, 2015; Rădulescu, Mannion, Roijers, & Nowé, 2019; Roijers, Vamplew, Whiteson, & Dazeley, 2013). Among these properties, one can mention: the number of objectives, the ability to learn a single policy or a set of undominated policies, whether the environment needs to be episodic, when can the user set its preferences, etc.

For example, some approaches only consider 2 objectives, e.g., (Avigad, Eisenstadt, & Cohen, 2011; Saisubramanian, Kamar, & Zilberstein, 2021). Even though they offer different advantages, they are therefore not suitable for implementing an algorithm that considers a generic number of moral values.

Other approaches directly learn a single, appropriate policy, e.g., (Ikenaga & Arai, 2018). They thus require user preferences to be specified upfront, so that the policy corresponds to these preferences. These works are naturally interesting, and important, because they allow preferences to be taken into account. However, specifying upstream makes it difficult to allow preferences to be changed during execution: the agent must in this case re-train its policy entirely. This may be an advantage, for example, to incrementally adapt the agent; but it is highly time-consuming. In addition, and depending on the preferences' structure, it might be difficult to specify by a lay user, e.g., for a vector of weights. By extension, this does not allow preferences to be contextualized, i.e., dependent on the situation.

Other works propose to learn a convex hull of Q-Values, e.g., (Barrett & Narayanan, 2008; Hiraoka, Yoshida, & Mishima, 2009; Mukai, Kuroe, & Iima, 2012). Instead of learning a single interest for each state-action pair, a convex hull allows learning a set of interests, for all possible preferences. Once the interests are learned, i.e., once the

agent is deployed, the policy can be obtained by injecting the preferences into the convex hull and choosing, for each state, the action that maximizes the preference-weighted interests. This type of approach effectively allows preferences to be changed, without the need to re-learn the agent's policy. On the other hand, it can still be complicated to use contextualized preferences; the agent would then have to re-compute its policy at each change of preferences, which can be costly if preferences change at (almost) every time step. Furthermore, this requires preferences in the form of weights, e.g., $\mathbf{w} = \{0.33, 0.67\}$, which may be difficult to specify. What would be the difference, i.e., the impact on the agent's policy, between $\mathbf{w} = \{0.33, 0.67\}$ and $\mathbf{w}' = \{0.4, 0.6\}$? A lay user may have to answer this question, or at least to reflect on it, in order to choose between these 2 sets of weights. There is no obvious translation between the tensions among several moral values, and the preferences as numbers.

Moreover, Hayes et al. (2022) note that few works are interested in both Multi-Objective and Multi-Agent:

> Numerous real-world problems involve both multiple actors and objectives that should be considered when making a decision. Multi-objective multi-agent systems represent an ideal setting to study such problems. However, despite its high relevance, it remains an understudied domain, perhaps due to the increasingly complex dimensions involved. (Hayes et al., 2022, p. 31)

This is therefore a limitation that still exists in the current literature, partly due to the lack of proposed benchmarks, which prevents MOMARL (Multi-Objective Multi-Agent Reinforcement Learning) approaches from being developed and compared on known, common use-cases. Existing benchmarks are perhaps too simple, and do not often consider multiple agents.

2.5 Hybrid Neural-Symbolic Artificial Intelligence

Methods in AI are often separated into neural, or connexionists, approaches and symbolic approaches. Each of them has its advantages, and drawbacks.

Harmelen & Teije (2019) mention, based on the literature, that learning systems, including deep learning ones, suffer from the following limitations: they are data-hungry, they have limited transfer on new tasks, they are susceptible to adversarial attacks, they are difficult to understand and explain, and they do not use prior knowledge. The field of

Deep Learning is constantly evolving, and these limitations may be alleviated by future works in Deep Learning. For example, research on *few-shot learning* may reduce the need for large dataset (Wang, Yao, Kwok, & Ni, 2020).

Symbolic reasoning systems, for their part, suffer from the following limitations: it is difficult to capture exceptions, they are unstable when facing noisy data, their size makes them expensive to build, and they may lead to combinatorial explosions when the number of rules increase.

Although neither pure learning nor pure reasoning techniques seem perfect, it is increasingly recognized that they can be complementary. The 3rd branch of AI, which brings together neural and symbolic to combine their advantages, is often called the Hybrid AI, or Neural-Symbolic AI. Many ways to do so have been proposed: Harmelen & Teije (2019) summarizes them in a "boxology" that classifies approaches and their common features.

For example, one of the most simple techniques may be to apply Machine Learning tools on symbolic inputs to produce symbolic outputs. Another variation may be to learn from "data", i.e., non-symbolic inputs in their terminology, to produce symbols. A large part of this boxology is dedicated to *explainable* systems, which is coherent with the current increase of literature on the subject: a ML tool may produce symbols from data, and a Knowledge Reasoning (KR) tool can be used to produce an explanation in the form of symbols. We refer the interested reader to the survey for more details about the evoked techniques (Harmelen & Teije, 2019).

Whereas Neural-Symbolic AI is not a novel idea – some works can be found as early as 1991 (Shavlik & Towell, 1991; Towell, 1991) and 1994 (Shavlik, 1994) – it recently seemed to gain traction. Hybrid approaches managed to attain impressive performances (D. Yu, Yang, Liu, & Wang, 2021) in various domains.

In the field of (Deep) Reinforcement Learning, more specifically, several works have been proposed. Figure 2.2 gives an example of the approach proposed by Garnelo, Arulkumaran, & Shanahan (2016), which represents an endogenous hybrid architecture, in which the RL agent comprises a *neural back-end*, which learns representations by mapping high-dimensional input data to low-dimensional symbols, and a *symbolic front-end*, which reasons about the low-dimensional symbols to decide which action should be executed. An advantage of their approach is that the RL agent learned to "recognize" different types of objects, and to only collect the correct ones, when compared to a purely Deep Learning approach (DQN). This also allows the hybrid architecture to transfer

knowledge between two tasks, one using a fixed grid, and another using a random grid, as the symbolic elements are similar enough. However, their experiments used a toy example, which raises questions about the proposed approach's ability to perform in a more complex ("real world") scenario.

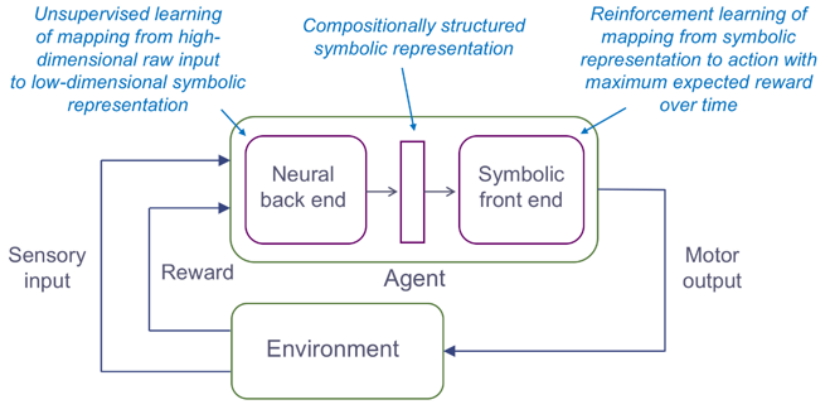


Fig. 2.2.: An example architecture of Hybrid (Deep) Reinforcement Learning, proposed by Garnelo et al. (2016).

Another approach tries to implement *Common Sense* into RL, through a hybrid architecture, using sub-states to represent symbolic elements (Garcez, Dutra, & Alonso, 2018). However, as in the previous work, their use-case rely on a toy example.

The previous examples can be categorized as adding symbolic elements to a machine learning base (the RL algorithm). A different form of Hybrid architecture is the other way around: implement ML techniques within a symbolic framework. For example, Bordini, El Fallah Seghrouchni, Hindriks, Logan, & Ricci (2020) focus on Agent-Oriented Programming (AOP) and propose to develop better BDI agents by leveraging ML at 3 different phases: 1) sensing, where, e.g., computer vision can be used to identify objects in a scene; 2) planning, where, e.g., ML can be used to learn contextual elements, or RL can be used to optimize action selection; and 3) acting, where ML can be used to determine the order in which steps should be executed to ensure they are achieved by their deadline. Interestingly, they mention that the architecture itself can be *endogenous*, i.e., the ML components and techniques are embedded within the BDI agent, or *exogenous*, i.e., ML is provided as a service, which may run on the same machine as the agent, or on a remote machine ("in the Cloud"). Such Cloud services are common in speech recognition or text-to-speech, for example.

3 Positioning, abstract overview and illustrative domain

In this chapter, we position our work with respect to the state of the art, and the identified objectives and challenges. We first describe our methodology, and justify the reasons behind some of our design choices. Then, we present what we call the “ethical model”, which is a systematic framework we use to structure our work. Section 3.3 contains an overview, or conceptual architecture, of the contributions that we detail in the other chapters of this manuscript. Finally, we introduce in Section 3.4 our use-case, a Smart Grid simulator of energy distribution that we will use throughout our experiments. We emphasize that our propositions are agnostic to the application case; this Smart Grid simulator is used as both an evaluation playground, and an exemple to anchor the algorithms in a realistic scenario.

3.1 Methodology

Our methodology is based on the following principles, which we detail below:

1. Pluri-disciplinarity
2. Incremental approach
3. Logical order of contributions
4. Human control in a Socio-Technical System
5. Diversity and multi-agent
6. Co-construction

The first principle of our methodology is its pluri-disciplinar character. Although I was myself only versed in multi-agent systems and reinforcement learning at the beginning of this thesis, my advisors have expertise in various domains of AI, especially multi-agent systems, learning systems, symbolic reasoning, agent-oriented programming, and on the other hand philosophy, especially related to ethics and AI. These disciplines have been mobilized as part of the ETHICS.AI research project.

A second, important principle of our methodology is that we propose an incremental approach, through successive contributions. Thus, we evaluate each of our contributions independently and make sure each step is working before building the next one. In addition, it allows the community to reuse or extend a specific contribution, without having to consider our whole architecture as some sort of monolith. Even though these contributions can be taken separately, they are still meant to be combined in a coherent architecture that responds to the objectives identified in Chapter 1. Thus, it is natural that a specific contribution, taken in isolation, includes few limitations that are in fact focused on in another contribution. At the end of each chapter, we discuss the advantages of the proposed contribution, its limitations and perspectives, and we explicitly note which ones are answered later in the manuscript. We recall that, as mentioned in Chapter 1, this had an impact on this document's structure, which thus presents models and related experiments in the same chapter.

Contributions have been made, and are presented, in a logical order. As we detail in Section 3.3 and the following chapters, we first describe 2 reinforcement learning algorithms, which focus on learning behaviours aligned with moral values and adaptation to changes. Once we have the learning algorithms, we can detail how to construct reward functions through symbolic judgments, in our second contribution: having proposed the RL algorithms beforehand allows us to evaluate our new reward functions, whereas the RL algorithms themselves can be evaluated on more traditional, mathematical reward functions. Finally, with the new reward functions that contain several explicit moral values through different judging agents, we can now focus on the question of dilemmas and conflicts between moral values.

We recall that we place our work in the larger scope of a Socio-Technical System, and we particularly focus on giving control back to humans. In this sense, the system that we propose must be taught how to behave accordingly to the moral values that are important to us. In other words, it is not that AI "solves" ethics for us humans, but rather humans (designers, but also users) who try to embed mechanisms into AI systems so that their behaviours will become more "ethically-aligned", and thus will empower humanity. This mindset influenced our propositions, and particularly the second contribution in Chapter 5, where one of the main goals is to make the reward function more understandable, and thus, scrutable by humans. It influenced even more so our third and last contribution in Chapter 6, in which we focus on human preferences, and explicitly give them control through interaction over the artificial agents' behaviours.

Another important aspect is that we focus on the richness of the use case. This stems from the diversity captured in the environment, which comes from several sources. First, the environment dynamics themselves, and the allowed interactions within the environment, through the designed observations and actions that we detail in Section 3.4, imply a variety of situations. Secondly, diversity is also related to the multiple moral values that agents need to take into consideration. And finally, as highlighted in the state of the art, we propose to increase diversity through the inclusion of multiple learning agents: agents may have different profiles, with different specificities, all of which suggesting different behaviours. We thus adopt the multi-agent principle as part of our method.

This multi-agent aspect starts on a single level in the first contribution, but becomes multi-level due to the addition of judging agents in the second contribution. This supports another important principle of our methodology, although we could not specify this as an objective of this thesis: the co-construction of agents. We did not have time to fully embrace co-construction, which is why it is not defined as an objective; nonetheless, we kept this notion in mind when designing our contributions. By co-construction, we mean that two different agents, which could be both human agents, both artificial, or even a mixed human-artificial combination, learn from each other and improve themselves through interaction with the other. That idea of co-construction is why we present our second contribution as "agentifying" the reward function: by making judging *agents* instead of simply judging *functions*, we pave the way for co-construction. Constructive relationships already occur in one of the two directions, as the learning agents improve their behaviour based on the signals sent by judging agents. But, in a larger scope, we could imagine judging agents also learning from learning agents, for example adapting their judgment behaviour so as to help the learners, in some sort of adversarial or active learning, thus enabling construction in the second direction and achieving co-construction. Another possibility of co-construction would be to make humans reflect on their own ethical considerations and preferences, through interaction with learning and judging agents. In this case, learning agents could act as some sort of simulation for example, the human users could try different preferences and discover the impact of their own preferences on the environment. Again, we emphasize that we did not have time to explore all these interesting research avenues. Yet, to keep co-construction attainable in potential future extensions, we tried and designed our contributions with this notion in mind.

3.2 The ethical model

In this section, we take a step back on the technical aspect of producing "ethical systems", or rather "ethically-aligned systems", and reflect on how can we achieve this from a philosophical and societal point of view. The ultimate goal is to have a satisfying "ethical outcome" of the system, in terms of both its resulting behaviour, i.e., the actions it proposes or takes, and also how the system functions, i.e., which inputs it takes, what resources the computation requires, etc.

We have mentioned in the previous section that the general framework of this thesis is a Socio-Technical System (STS), in which we want to give the control back to the humans. This focus on STS when talking about AI and ethics has been championed by several researchers. Stahl (2022), notably, describes the discourses on computer ethics, and ethics of AI, and proposes to move towards *ethics of digital ecosystems*, instead of focusing on specific artefacts. This notion of ecosystems is a metaphor that refers to socio-technical systems in which AI systems and humans interact with each other; as they mention:

> Ethical, social, human rights and other issues never arise from a technology per se, however, but result from the use of technologies by humans in societal, organisational and other setting. (Stahl, 2022, p. 72)

In other words, AI systems should not be considered "alone", but rather as a part of a more complex system incorporating humans. By focusing on STS, they hope to be able to explore and discover ethical issues related to new technologies, including but not limited to AI, and without the need to clearly define what AI is. In a similar vein, although they primarily focus on trustworthiness and legitimacy instead of ethics, Whitworth & De Moor (2003) propose to take into account users' requirements and concerns, such as privacy, and other ethics-related issues, directly into the design of STS.

To be ethically aligned with our ethical considerations, AI systems need to be guided. This guidance necessarily comes from humans, as ethics is, for now at least, a privilege of humans. The ethical considerations, through this guidance, are "injected" into systems, and we name this an *ethical injection*. This injection can take multiple forms, depending on the considered AI system, e.g., it can be part of the data, when we talk about supervised learning, or directly implemented as part of an algorithm's safeguards, or part of a learning signal, etc. This notion of ethical injection builds, among other things, upon the principles of STS, according to which it is important that the system incorporates human concerns.

From this description of the "ethical injection", we can see that it relates to the already mentioned "Value Alignment Problem". This problem has been described by many researchers, including moral philosophers and computer scientists, see, e.g., (Gabriel, 2020; Russell, 2021). It mainly refers to 2 important concerns when building an AI system: first, we humans must correctly articulate the objective(s) that we want the system to solve. For example, proxy objectives may be used to simplify the definition of the true objectives, however we risk getting a behaviour that solves only the proxy objectives, and not the true ones. Secondly, the system must solve these objectives while not taking any action that would contradict our values. For example, reducing the emission of greenhouse gaz by killing all of humanity is not an acceptable solution. In our framework, the "ethical injection" is the technical way of incorporating these ethical considerations, both the true objectives and the values that should not be contradicted, into the AI system.

In order to make an AI system more beneficial for us, it should include the moral values that we want the system to be aligned with. Several works have proposed, notably, *principles* that AI systems should follow in order to be beneficial (Jobin, Ienca, & Vayena, 2019). Another work debates this, and recommends instead focusing on *tensions*, as all principles cannot be applied at the same time (Whittlestone, Nyrup, Alexandrova, & Cave, 2019). In any way, we argue that, as the vast majority of these choices do not target only technical considerations, they cannot, or should not, be made in isolation by AI experts. They must in fact involve a larger audience, from several disciplines and origins: AI experts, domain experts, philosophers, stakeholders, and impacted or interested citizens.

Yet, this idea raises a problem: these various participants do not have the same knowledge, and therefore need some common ground, so that they can discuss the ethical choices together. This common ground is necessary, we believe, to produce definitions that are both technically doable, and morally suitable. For example, the definition of justice, or fairness, can be difficult to implement by AI experts alone, as multiple definitions are used by different cultures, see for example Boarini, Laslier, & Robin (2009). Which one should they use? On the other hand, definitions produced by non-technical experts may prove difficult, or even impossible, to implement, because they rely on abstract notions.

Discussion between the various participants should yield the *ethical injection*, that is, the data and design choices used to guide the system towards an ethical outcome. For example, making the explicit choice of using a simplified model, which requires less computational resources (Green IT) would be considered part of the ethical injection.

Similarly, the choice of the inputs that the system will be given access to can be considered part of the ethical injection, especially if the choice implies notions of privacy, accuracy, etc. Finally, in a learning system, the choice of the training data will also be important for the ethical injection, as it represents the basis from which the system will derive its behaviour.

Reaching this common ground is not trivial, as participants may come from different domains. Yet, it is crucial, because design choices shape the way the system works, and some of them have ethical consequences. The various ways and *loci* of ethical injection should thus be discussed collectively: to do so, the AI system needs to be simplified, so that the discussion can focus on the important aspects, without being diverted by details. Still, it should not be too much simplified, otherwise we would risk obtaining a wrong mental model of the system. For example, simplifying too much the idea of a physical machine, e.g., when talking about Cloud-based solutions, would make participants forget about the physical impact of computation, in terms of resources (energy, space, etc.). Simplifying too much about AI systems' capabilities could raise false hopes, or make them believe that the system is never wrong. Presenting the correct level of simplification, i.e., neither too little nor noo much, is at the crux of getting this common ground.

Figure 3.1 represents an agnostic system in interaction with a human society. The diverse arrows, such as "construct computational structure", "construct input", "produce input", etc., all represent potential *loci* of ethical injection. On the left side of the figure, humans with various roles, e.g., designers, users, or stakeholders, discuss and interact with the system, on the right side. The "interface" represents the way the system can communicate with the external world, which includes our human society; the interface is composed of inputs and outputs, of which the structure is chosen by the humans. Inputs, in particular may also be produced by humans, e.g., datasets. In the rest of the manuscript, we consider notably that the moral values, the reward functions, especially when constructed from symbolic judgments, and the contextualized preferences, are the main components of the ethical injection, aside with a few design choices that protect privacy for example. We focus on the technical ability of the learning system to leverage this ethical injection, and thus we place ourselves as the "almighty" designers, without discussing with potential users, nor stakeholders. Nevertheless, we defend that, for a real deployment of such systems, into our human society, these choices should be discussed. Moral values, in particular, should be elicited, to make sure that we have not forgotten one of them. We will sometimes refer to this idea of including users and other stakeholders in the

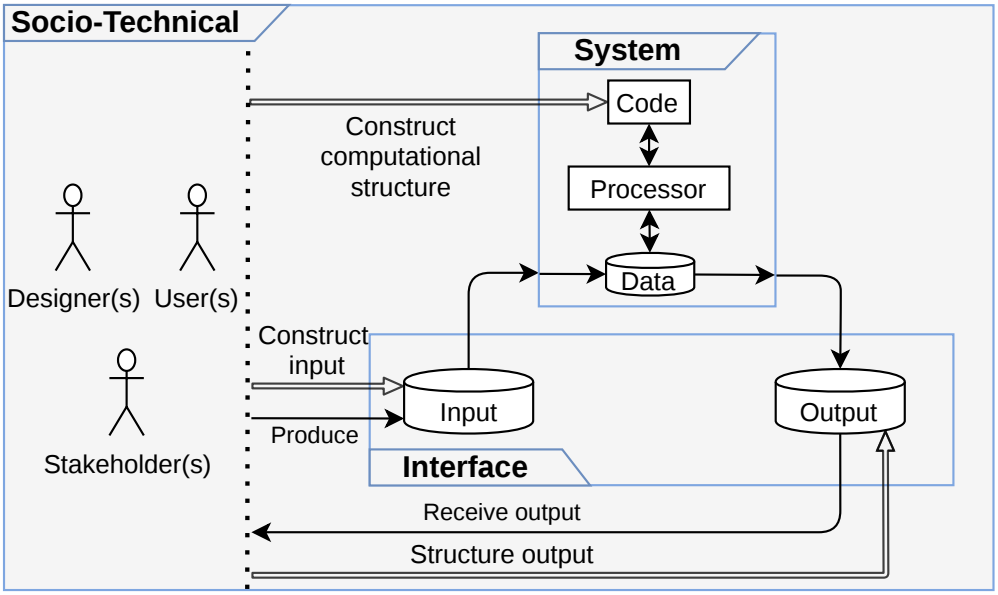


Fig. 3.1.: Representation of a Socio-Technical System, in which multiple humans construct and interact with a system.

discussion, which requires understanding on their part, and therefore intelligibility on the part of the system.

3.3 Conceptual architecture

In this thesis, we propose a multi-agent system, comprised of multiple agents interacting in a shared environment. This architecture can be seen, on an abstract level, as composed of 3 parts, shown in Figure 3.2. First, the learning part that we present in Chapter 4 introduces learning agents, which are tasked with learning an ethical behaviour, i.e., a behaviour aligned with moral values that are important to humans. To do so, they contain a decision and a learning processes, which feed each other to choose actions based on observations received from the shared environment, and to learn how to choose better actions. The shared environment receives the learners' actions and computes its next state based on its own dynamics. Observations from the environment state, which include both global data and local data specific to each agent, are then sent to the learning agents so that they can choose their next action, and so on.

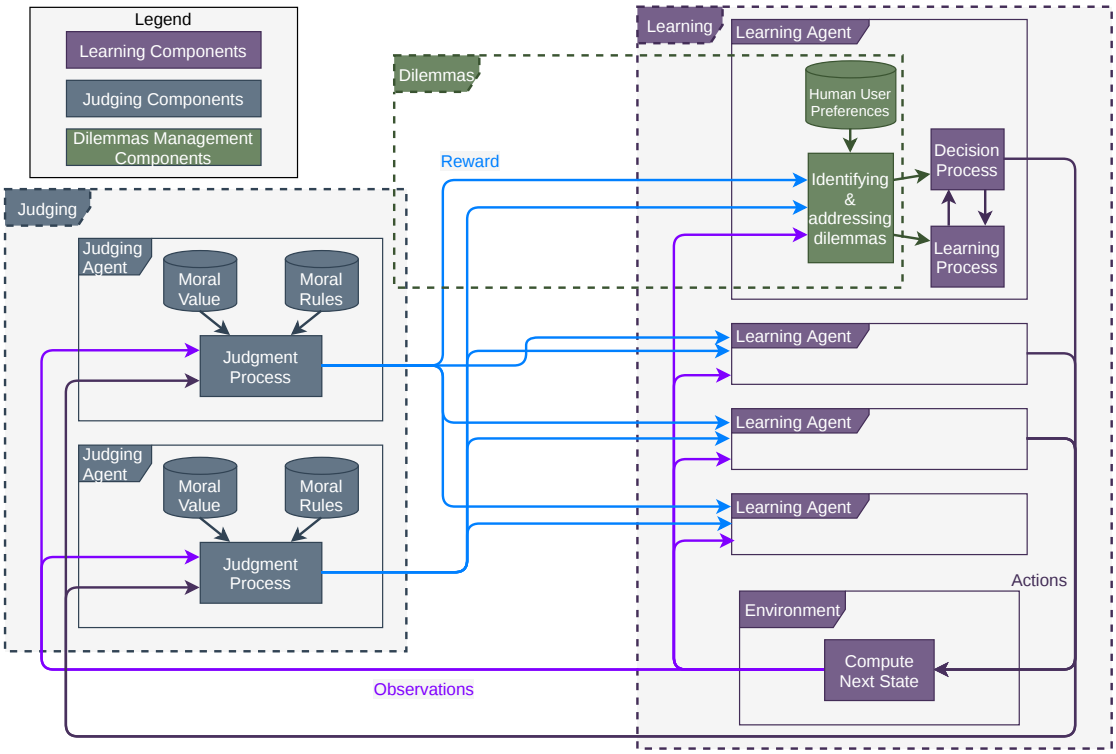


Fig. 3.2.: Conceptual architecture of our approach, composed of 3 parts in interaction: the learning part, the judging part, and the handling of dilemmas.

The second part concerns the construction of the reward signal, which is sent to the learning agents as an indication of the correctness of their action. Whereas we do not know exactly which action should have been performed (otherwise we would not need to learn behaviours), we can judge the quality of the action. An action will rarely be 100% correct or incorrect; the reward signal should therefore be rich enough to represent the action's correctness, and accurately reflect the eventual improvement, or conversely deterioration, of the agents' behaviour. Learning agents will learn to select actions that yield higher rewards, which are better aligned with our moral values, according to the reward function. In order to construct this reward function, we propose in Chapter 5 to use distinct judging agents, each tasked with a specific moral value and the relevant moral rules. Judging agents employ a symbolic reasoning that is easier to design, and more understandable.

Finally, the last part focuses on addressing the specific situations of ethical dilemmas, when multiple moral values are in conflict and cannot be satisfied at the same time. In order to, on the first hand, be able to identify these dilemmas, and on the second hand to manage them, we introduce in Chapter 6 an extension to the learning algorithm. Instead of the simpler configuration in which learning agents receive an aggregated reward, the new learning algorithm receives multiple rewards, one for each moral value. A new process, specialized on dilemmas, is introduced and interacts with human users to identify situations that are relevant to them, and to learn their ethical preferences so as to automatically settle dilemmas accordingly. The goal of the learning agents is thus to learn to settle the dilemmas according to humans' contextualized preferences; this implies a certain number of interactions between the learning agents and the human users. As the agents learn the users' preferences, the amount of interactions decreases, thus avoiding putting too much of a burden on the human users.

3.4 Smart Grids as a validation use-case

In order to demonstrate that our proposed approach and contributions can be effectively applied, we developed a Smart Grid simulator.

We emphasize that this simulator is meant only as a validation domain: the approach we describe in this thesis is thought as generic. In particular, the learning algorithms could be leveraged for other domains.

3.4.1 Motivation

The choice of the Smart Grid use-case is motivated by our industrial partner Ubiant, which has expertise in this domain and helped us make a sufficiently realistic (although simplified) simulation. We also argue that this use-case is rich enough to embed ethical stakes, in multiple situations, concerning multiple moral values and multiple stakeholders, as per our objectives and hypotheses.

One may wonder why we chose to propose a new use-case, rather than focusing on existing "moral dilemmas" that have been proposed by the community of researchers: Trolley Dilemmas, Cake or Death, Care robot for alcoholic people, etc. [1] As LaCroix points out, moral dilemmas have been misunderstood and misused in Machine Ethics, especially the trolley-style dilemmas (LaCroix, 2022). For example, the well-known Trolley Dilemma (Foot, 1967; Thomson, 1976) was meant to highlight the difference between "killing" and "letting die". Whereas it may seem acceptable (or even desirable) to steer the lever, and letting 1 people die to save 5 another, we also intuitively dislike the idea of a person pushing someone (the Fat Man variation), or of a physician killing voluntarily a healthy person to collect their organs and save 5 patients. Although consequences are very similar (1 person dies; 5 are saved), there is a fundamental difference between "killing" and "letting die", which makes the former unacceptable. This observation was one of the primary goals of the Trolley Dilemma and similar philosophical thought experiments, which have since led to countless discussions and applications in moral philosophy. However, regardless of their importance for the study of philosophical and legal concepts, they do not represent realistic case studies for simulations aimed at learning or implementing ethically-aligned behaviours.

Therefore, instead of focusing solely on "moral dilemmas", we want to consider an environment with ethical stakes. These ethical stakes transpire at different moments (time steps) of the simulation. Contrary to moral dilemmas, they sometimes may be easy to satisfy. Indeed, we see that in our lives, we are not always in a state of energy scarcity, nor in an impeding-death scenario when we drive. However, even in these scenarii, there are stakes that the artificial agent should consider. Even when it is simple to consume, because there is no shortage, the agent should probably not consume more than it needs, in order to let other people consume as well. Machine Ethics approaches should be validated on these "simple" cases as well. And, on the other hand, there are "difficult"

[1] A list of dilemmas can be found on the now defunct DilemmaZ database, accessible through the Web Archive: https://web.archive.org/web/20210323073233/https://imdb.uib.no/dilemmaz/articles/all

situations, when two (or more) moral values are in conflict and cannot be satisfied at the same time. In this case, which we call a dilemma, although not in the "thought experiment" sense, a compromise must be made.

3.4.2 Definition

As Smart Grids are a vivid research domain, there are several definitions of a Smart Grid, and more generally, Smart Energy Systems. Hadjsaïd & Sabonnadière (2013) note that these definitions depend on the priorities of actors behind the development of these grids, such as the European Union, the United States of America, or China. We propose to focus on the EU's definition (EU Commission Task Force for Smart Grids, 2010) in order to highlight one of the major differences between *smart* and "*non-smart*" grids:

Definition 3.1 (Smart Grid)**.** A Smart Grid is an electricity network that can cost efficiently integrate the behaviour and actions of all users connected to it – generators, consumers and those that do both – in order to ensure economically efficient, sustainable power system with low losses and high levels of quality and security of supply and safety. Though elements of smartness also exist in many parts of existing grids, the difference between a today's grid and a smart grid of the future is mainly the grid's capability to handle more complexity than today in an efficient and effective way. A smart grid employs innovative products and services together with intelligent monitoring, control, communication, and self-healing technologies in order to:

- Better facilitate the connection and operation of generators of all sizes and technologies.
- Allow consumers to play a part in optimising the operation of the system.
- Provide consumers with greater information and options for how they use their supply.
- Significantly reduce the environmental impact of the whole electricity supply system.
- Maintain or even improve the existing high levels of system reliability, quality and security of supply.
- Maintain and improve the existing services efficiently.
- Foster market integration towards European integrated market.

(EU Commission Task Force for Smart Grids, 2010, p. 6)

As can be seen in definition 3.1, *smart grids* add various technologies ("innovative products and services"), in order to improve and extend the grid's functionalities. For example, instead of being mere consumers of energy, the introduction of solar panels allow the grid's participants to become *prosumers*, i.e., both producers and consumers. In this case, they gain the ability to truly participate in the grid's energy dynamics by exchanging energy with the grid, and thus other prosumers, in a two-way manner. We refer the interested reader to X. Yu, Cecati, Dillon, & Simões (2011) and Hadjsaïd & Sabonnadière (2013) for more details on Smart Grids, their definitions, and challenges.

Multiple ethical issues have been identified in Smart Grids (Tally, Rowena, & David, 2019), such as: health, privacy, security, affordability, equity, sustainability. Some of them can be addressed at the grid conception level, e.g., privacy and security by making sound algorithms, health by determining the toxicity of exposition to certain materials, and limiting such exposition if necessary. The remaining ones, although partially addressable by design, can also be addressed at the prosumer behaviour level. For example, prosumers can reduce their consumption when the grid is over-used, in order to let other prosumers, more in need, improve their comfort, and avoiding to buy energy from highly polluting sources.

Specifically, we imagine the following scenario. A micro-grid provides energy to a small neighbourhood of prosumers. Instead of being solely connected to the national grid, the micro-grid embeds its own production, through, e.g., a hydroelectric power plant. Prosumers can produce their own energy through the use of photovoltaic panels, and store a small quantity in their personal battery. They must distribute energy, i.e., choose how much to consume from the micro-grid, from their battery, how much to give back to help a fellow neighbor, how much to buy or sell from the national grid, in order so to satisfy their comfort, supporting a set of moral values all the while. Prosumers are represented by artificial agents that are tasked with taking these decisions to alleviate the prosumers' life, according to their profiles and wishes. This is another marker of the "smart" aspect in Smart Grids.

3.4.3 Actions

The prosumers take actions that control the energy dynamics within the grid. Prosumers are therefore able to:

- Consume energy from the micro-grid to improve their comfort.

- Store energy in their battery from the micro-grid.
- Consume energy from their battery to improve their comfort.
- Give energy from their battery to the micro-grid.
- Buy energy from the national grid, which is stored in their battery.
- Sell energy from their battery to the national grid.

All of these actions can be done at the same time by a prosumer, e.g., they can both consume from the micro-grid and buy from the national grid. We will detail the exact representation of these actions in Section 4.4. Let us remark that these actions offer the possibility of rich behaviours: prosumers can adopt long-term strategies, e.g., by buying and storing when there is enough energy, in prevision of a production shortage or conversely a consumption peak, and give back when they do not need much.

These actions, and the behaviours that ensue from them, are also often ethically significant. For example, a prosumer who consumes too much energy might prevent another one from consuming enough, resulting in a higher inequality of comforts.

Our goal is therefore to make artificial agents learn how to consume and exchange energy, as some sort of proxy for the human prosumers. These behaviours will need to consider and take into account the ethical stakes.

3.4.4 Observations

In order to help the prosumers take actions, they receive observations about the current state of the environment. Indeed, we do not need to consume the same amount of energy at different times. Or, we may recognize that there is inequality in the grid, and decide to sacrifice a small part of our comfort to help the left-behind prosumers.

These observations may be common, i.e., accessible to all prosumers in the grid, such as the hour, and the current available quantity of energy, or they may be individual, i.e., only the concerned agent has access to the value, such as the personal comfort, and the current quantity of energy in the battery. The distinction between individual and common observations allows to take into account ethical concerns linked with privacy: human users may want to keep part of their individual situation confidential.

The full list of observations that each agent receives is:

- **Storage**: The current quantity of energy in the agent's personal battery.

- **Comfort**: The comfort of the agent, at the previous time step.
- **Payoff**: The current payoff of the agent, i.e., the sum of profits and costs made from selling and buying energy.
- **Hour**: The current hour of the day.
- **Available energy**: The quantity of energy currently available in the micro-grid.
- **Equity**: A measurement of the equity of comforts between agents, at the previous time step, using the Hoover index.
- **Energy waste**: The quantity of energy that was not consumed and therefore wasted, at the previous time step.
- **Over-consumption**: The quantity of energy that was consumed but not initially available at the previous time step, thus requiring the micro-grid to purchase from the national grid.
- **Autonomy**: The ratio of transactions (energy exchanges) that were not made with the national grid. In other words, the less agents buy and/or sell, the higher the autonomy.
- **Well-being**: The median comfort of all agents, at the previous step.
- **Exclusion**: The proportion of agents which comfort was lower than half the median.

3.4.5 Moral values and ethical stakes

We have explored the literature to identify which moral values are relevant in the context of Smart Grids. Several works have identified moral values that should be considered by political decision-makers when conceiving Smart Grids (de Wildt, Chappin, van de Kaa, Herder, & van de Poel, 2019; Milchram, Van de Kaa, Doorn, & Künneke, 2018). These moral values have been defined with the point of view of political decision-makers, rather than prosumers. For example, the "affordability" value originally said that Smart Grids should be conceived such that prosumers are able to participate without buying too much. In our use-case, we want artificial agents to act as proxy for prosumers, and thus they should follow moral values that take the point of view of prosumers. We therefore adapted these moral values, e.g., for the "affordability", our version states that prosumers should buy and sell energy in a manner that does not exceed their budget.

We list below the moral values and associated rules that are retained in our simulator:

MR1 – Security of Supply An action that allows a prosumer to improve its comfort is moral.

MR2 – Affordability An action that makes a prosumer pay too much money is immoral.

MR3 – Inclusiveness An action that improves the equity of comforts between all prosumers is moral.

MR4 – Environmental Sustainability An action that prevents transactions (buying or selling energy) with the national grid is moral.

3.4.6 Prosumer profiles

In order to increase the richness of our simulation, we introduce several *prosumer profiles*. A prosumer agent represents a building: we do not consider a smaller granularity such as buildings' parts, or electrical appliances. Prosumer profiles determine a few parameters of agents:

- The needs in terms of Watt-hours of the buildings' inhabitants, for each hour of the year. This need is extracted from public datasets of energy consumption: given the energy consumed by a certain type of building, during a given hour, we assume this represents the amount of energy the inhabitants needed at this hour. The need acts as some sort of target for the artificial agents, which must learn to consume as close as possible to the target need, according to MR1, for every hour of the year. Seasonal changes implies different needs, for example we have a higher consumption in Winter, due to the additional heating.

- The personal storage's capacity. Each building has a personal storage, or battery, that they can use to store a limited quantity of energy. This limit, that we call the capacity, depends on the size of the (typical) building: smaller buildings will have a smaller capacity.

- The actions' maximum range. As we have seen previously, actions are represented by quantities of energy that are exchanged. For example, a household could perform the action of "consuming 572Wh". Learning algorithms require this range to be bounded, i.e., we do not know how to represent "consume infinite energy"; we chose to use different ranges for different profiles. Indeed, profiles of larger buildings have a higher need of energy. Thus, we should make it possible for them

to consume at least as much as they can possibly need. For example, if a building has a maximum need, along all hours of the year, of 998Wh, we say that its action's maximum range is $1,000$Wh.

To define our building profiles, we used an OpenEI public dataset of energy consumption (Ong & Clark, 2014), which contains consumptions for many buildings in different places in the United States of America. We have defined 3 profiles, based on *Residential*, which we rename *Household*, *Office*, and *School* buildings from the city of Anchorage:

- The *Household* type of agents has an action range of $2,500$Wh, and a personal storage's capacity of 500Wh.
- The *Office* type of agents has an action range of $14,100$Wh, and a personal storage's capacity of $2,500$Wh.
- The *School* type of agents has an action range of $205,000$Wh, and a personal storage's capacity of $10,000$Wh.

Figure 3.3 shows the need, for each hour in a year, for these 3 profiles.

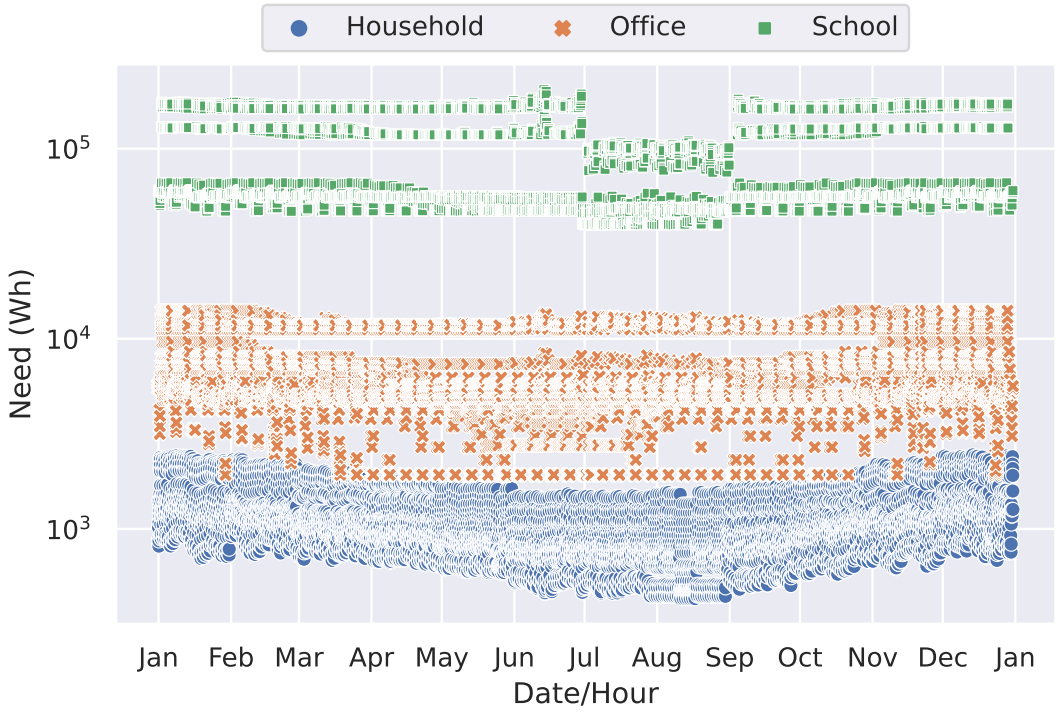


Fig. 3.3.: Energy need for each of the building profiles, for every hour of a year, from the OpenEI dataset.

3.4.7 Summary

We summarize this proposed use-case in Figure 3.4. As we can see, the grid contains several buildings, each represented by a learning (prosumer) agent. Three buildings profiles are available: Household, which represents a small, single-family habitation; Office, which represents a medium-sized office; and School, which represents a large building with important needs. Each building has a solar panel that regularly produces energy, and a personal storage in which the energy is stored. They are linked to a Smart Grid, which contains a source of energy shared by all buildings, e.g., a hydraulic power plant, and which is also linked to the national grid. The buildings receive observations from the environment, which can be either *individual* (local) data that concern only a given agent, or *shared* (global) data that are the same for all buildings at a given time step. This protects the privacy of the buildings' inhabitants, as it avoids sharing, e.g., the payoff, or comfort, of an agent to its neighbors. In return, they take actions that describe the energy transfers between them, their battery, the micro-grid power plant, and the national grid.

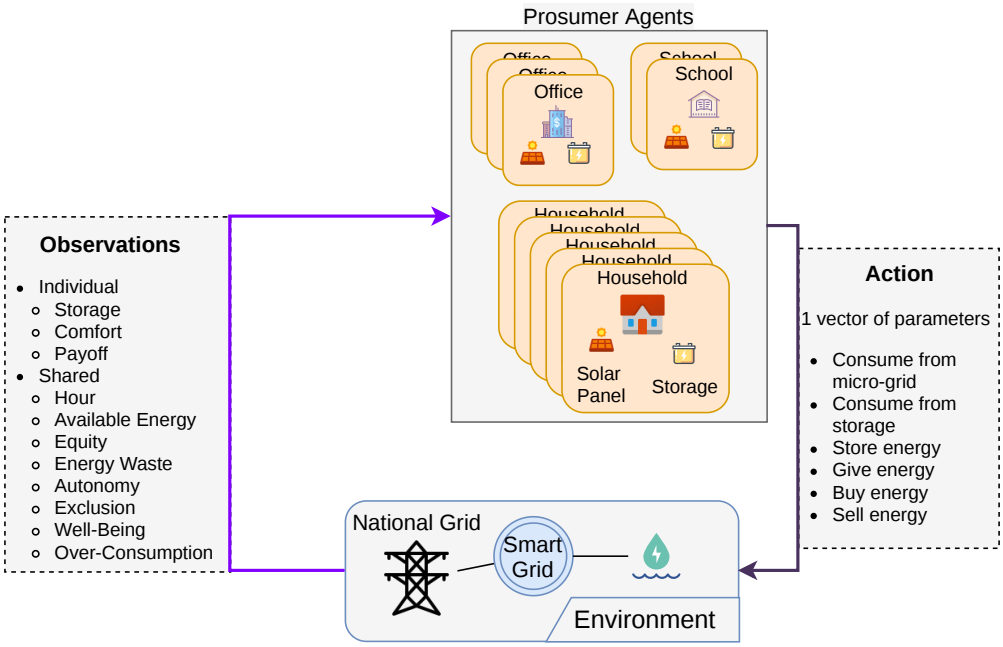


Fig. 3.4.: A Smart Grid as per our definition. Multiple buildings (and their inhabitants) are represented by artificial agents that learn how to consume and exchange energy to improve the comfort of the inhabitants. Buildings have different profiles: Household, Office, or School.

4 Learning behaviours

In this chapter, we present the first contribution, which is the learning part from the conceptual architecture presented in Figure 3.2. Section 4.1 begins with an overview of this contribution, recalls the research question, explains its motivations, and presents a schema derived from the conceptual architecture, focusing on the learning part, and showing its internal mechanisms. Sections 4.2 and 4.3 presents the components necessary to our algorithms, and algorithms are then described in Section 4.4. We detail the experiment setup, based on the multi-agent Smart Grid use-case presented in Section 3.4, in Section 4.5, and Section 4.6 reports the results. A discussion of these algorithms' advantages and drawbacks is finally presented in Section 4.7.

4.1 Overview

We recall that the aim of this first contribution is to answer the first research question: How to learn behaviours aligned with moral values, using Reinforcement Learning with complex, continuous, and multi-dimensional representations of actions and situations? How to make the multiple learning agents able to adapt their behaviours to changes in the environment?

To do so, we propose 2 Reinforcement Learning algorithms. These algorithms needed to have 2 important properties: 1) to be able to handle multi-dimensional domains, for both states and actions, as per our hypothesis H2, and as motivated in our state of the art of Machine Ethics. 2) to be a modular approach, as per the "incremental approach" principle of our methodology. Indeed, the algorithms were meant as a first step, a building basis on top of which we would add improvements and features throughout our next contributions. In order to facilitate these improvements, modularity was important, rather than implementing an end-to-end architecture, as is often found in Deep Neural Networks. Such networks, while achieving impressive performance on their tasks, can be difficult to reuse and modify. As we will see in the results, our algorithms obtain similar or even better performance.

The algorithms that we propose are based on an existing work (Smith, 2002a, 2002b) that we extend and evaluate in a more complex use-case. Smith's initial work proposed, in order to handle continuous domains, to associate Self-Organizing Maps (SOMs) to a Q-Table. This idea fulfills the first property, and also offers the modularity required by the second: by combining elements, it allows to easily modify, replace, or even add one or several elements to the approach. This motivated our choice of extending Smith's work.

First, we explain what is a Q-Table from the Q-Learning algorithm, and its limitations. We then present Self-Organizing Maps, the Dynamic Self-Organizing Map variation, and how we can use them to solve the Q-Table's limitations. We combine these components to propose an extension of Smith's algorithm that we name *Q-SOM*, which leverages a Q-Table and Self-Organizing Maps (SOMs), and a new algorithm named *Q-DSOM*, which leverages a Q-Table and Dynamic SOMs (DSOMs).

Learning agents presented in this chapter correspond to prosumer agents in Figure 3.4. They receive observations and take actions; during our experiments, these observations and actions are those described in this figure and more generally in Section 3.4.

Figure 4.1 presents a summarizing schema of our proposed algorithms. It includes multiple learning agents that live within a shared environment. This environment sends observations to agents, which represent the current state, so that agents may choose an action and perform it in the environment. In response, the environment changes its state, and sends them new observations, potentially different for each agent, corresponding to this new state, as well as a reward indicating how correct the performed action was. Learning agents leverage the new observations and the reward to update their internal model. This observation-action-reward cycle is then repeated so as to make learning agents improve their behaviour, with respect to the considerations embedded in the reward function. The decision process relies on 3 structures, a State (Dynamic) Self-Organizing Map, also named the State-(D)SOM, an Action (Dynamic) Self-Organizing Map, also named the Action-(D)SOM, and a Q-Table. They take observations as inputs and output an action, which are both vectors of continuous numbers. The learning process updates these same structures, and takes the reward as an input, in addition to observations.

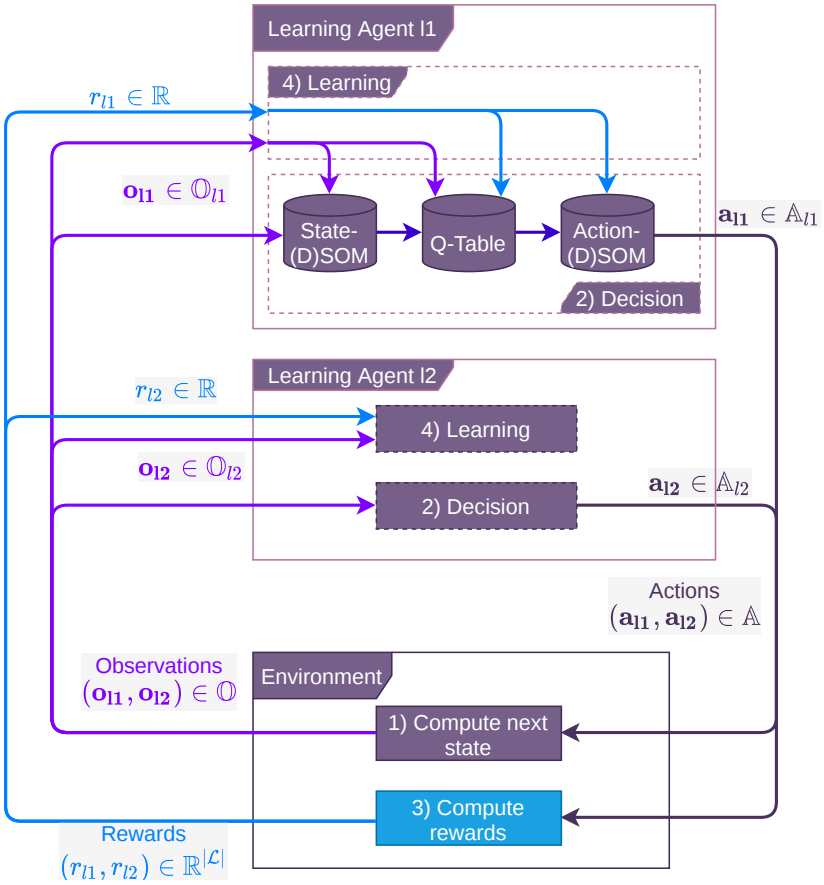


Fig. 4.1.: Architecture of the Q-SOM and Q-DSOM algorithms, which consist of a decision and learning processes. The processes rely on a State-(D)SOM, an Action-(D)SOM, and a Q-Table.

4.2 Q-Table

The *Q-Table* is the central component of the well-known Q-Learning algorithm (Watkins & Dayan, 1992). It is tasked with learning the *interest* of a state-action pair, i.e., the expected horizon of received rewards for taking an action in a given state. As we saw in Section 2.2, the Q-Table is a tabular structure, where rows correspond to possible states, and columns to possible actions, such that the row $\texttt{Q}(s, \cdot)$ gives the interests of taking every possible action in state s, and, more specifically, the cell $\texttt{Q}(s, a)$ is the interest of taking action a in state s. These cells, also named *Q-Values*, can be learned iteratively by collecting experiences of interactions, and by applying the Bellman equation.

We recall that the interests take into account both the short-term immediate reward, but also the interest of the following state s', resulting from the application of a in s. Thus, an action that leads to a state where any action yields a low reward, or in other word an unattractive state, would have a low interest, regardless of its immediate reward.

Assuming that the Q-Values have converged towards the "true" interests, the optimal policy can be easily obtained through the Q-Table, by selecting the action with the maximum interest in each state. By definition, this "best action" will lead to states with high interests as well, thus yielding, in the long-term, the maximum expected horizon of rewards.

An additional advantage of the Q-Table is the ability to directly have access to the interests, in comparison to other approaches, such as *Policy Gradient*, which typically manipulate actions' probabilities, increasing and decreasing them based on received rewards. These interests can be conveyed to humans to support or detail the algorithm's decision process, an advantage that we will exploit later, in Chapter 6.

Nevertheless, Q-Tables have an intrinsic limitation: they are defined as a tabular structure. This structure works flawlessly in simple environments, e.g., those with a few discrete states and actions. Yet, in more complex environments, especially those that require continuous representations of states and actions, it is not sufficient any more, as we would require an infinite number of rows and columns, and therefore an infinite amount of memory. Additionally, because of the continuous domains' nature, it would be almost impossible to obtain twice the exact same state: the cells, or Q-Values, would almost always get at most a single interaction, which does not allow for adequate learning and convergence towards the true interests.

To counter this disadvantage, we rely on the use of Self-Organizing Maps (SOMs) that handle the continuous domains. The mechanisms of SOMs are explained in the next section, and we detail how they are used in conjunction with a Q-Table in Section 4.4.

4.3 (Dynamic) Self-Organizing Maps

A Self-Organizing Map (SOM) (Kohonen, 1990) is an artificial neural network that can be used for unsupervised learning of representations for high-dimensional data. SOMs contain a fixed set of neurons, typically arranged in a rectangular 2D grid, which are associated to a unique identifier, e.g., neuron #1, neuron #2, etc., and a vector, named the *prototype vector*. Prototype vectors lie in the latent space, which is the highly dimensional space the SOM must learn to represent.

The goal is to learn to represent as closely as possible the distribution of data within the latent space, based on the input data set. To do so, prototype vectors are incrementally updated and "moved" towards the different regions of the latent space that contain the most data points. Each time an input vector, or data point, is presented to the map, the neurons compete for attention: the one with the closest prototype vector to the input vector is named the *Best Matching Unit* (BMU). Neurons' prototypes are then updated, based on their distance to the BMU and the input vector. By doing this, the neurons that are the closest to the input vector are moved towards it, whereas the farthest neurons receive little to no modification, and thus can focus on representing different parts of the latent space.

As the number of presented data points increases, the distortion, i.e., the distance between each data point and its closest prototype, diminishes. In other words, neurons' prototypes are increasingly closer to the real (unknown) distribution of data.

When the map is sufficiently learned, it can be used to perform a mapping of high dimensional data points into a space of lower dimension. Each neuron represents the data points that are closest to its prototype vector. Conversely, each data point is represented by the neuron whose prototype is the closest to its own vector.

This property of SOMs allows us to handle continuous, and multi-dimensional state and action spaces.

Figure 4.2 summarizes and illustrates the training of a SOM. The blue shape represents the data distribution that we wish to learn, from a 2D space for easier visualization. Typically, data would live in higher dimension spaces. Within the data distribution, a white disc shows the data point that is presented to the SOM at the current iteration step. SOM neurons, represented by black nodes, and connected to their neighbors by black edges, are updated towards the current data point. Among them, the Best Matching Unit, identified by an opaque yellow disc, is the closest to the current data point, and as such receives the most important update. The closest neighbors of the BMU, belonging to the larger yellow transparent disc, are also slightly updated. Farther neurons are almost not updated. The learned SOM is represented on the right side of the figure, in which neurons correctly cover the data distribution.

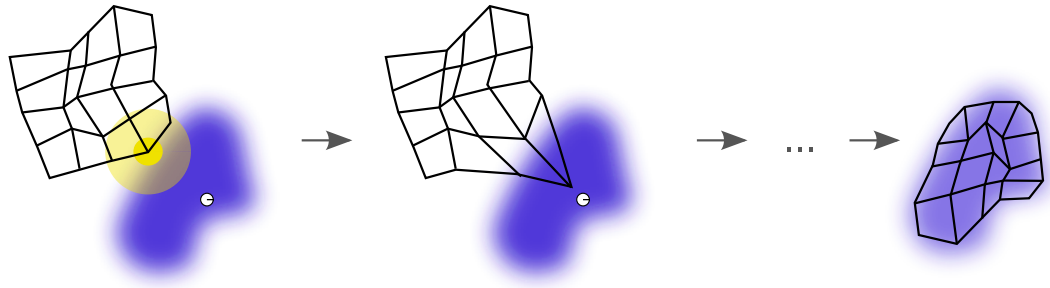


Fig. 4.2.: Training of a SOM, illustrated on several steps. Image extracted from Wikipedia.

The update received by a neuron is determined by Equation (4.1), with v being the index of the neuron, $\mathbf{W_v}$ is the prototype vector of neuron v, $\mathbf{D_t}$ is the data point presented to the SOM at step t. u is the index of the Best Matching Unit, i.e., the neuron that satisfies $u = \text{argmin}_{\forall v} \|\mathbf{D_t} - \mathbf{W_v}\|$.

$$\mathbf{W_v^{t+1}} \leftarrow \mathbf{W_v^t} + \theta(u, v, t)\alpha(t)\left(\mathbf{D_t} - \mathbf{W_v^t}\right) \tag{4.1}$$

In this equation, θ is the neighborhood function, which is typically a gaussian centered on the BMU (u), such that the BMU is the most updated, its closest neighbors are slightly updated, and farther neurons are not updated. The learning rate α, and the neighborhood function θ both depend on the time step t: they are often monotonically decreasing, in order to force neurons' convergence and stability.

One of the numerous extensions of the Self-Organizing Map is the Dynamic Self-Organizing Map (DSOM) (Rougier & Boniface, 2011). The idea behind DSOMs is that self-organization should offer both stability, when the input data does not change much, and dynamism,

when there is a sudden change. This stems from neurological inspiration, since the human brain is able to both stabilize after the early years of development, and dynamically re-organize itself and adapt when lesions occur.

As we mentioned, the SOM enforces stability through decreasing parameters (learning rate and neighborhood), however this also prevents dynamism. Indeed, as the parameters approach 0, the vectors' updates become negligible, and the system does not adapt any more, even when faced with an abrupt change in the data distribution.

DSOMs propose to replace the time-dependent parameters by a time-invariant one, named the *elasticity*, which determines the coupling of neurons. Whereas SOMs and other similar algorithms try to learn the density of data, DSOMs focus on the structure of the data space, and the map will not try to place several neurons in a high-density region. In other words, if a neuron is considered as sufficiently close to the input data point, the DSOM will not update the other neurons, assuming that this region of the latent space is already quite well represented by this neuron. The "sufficiently close" is determined through the elasticity parameter: with high elasticity, neurons are tightly coupled with each other, whereas lower elasticity let neurons spread out over the whole latent space.

DSOMs replace the update equation with the following:

$$\mathbf{W_i^{t+1}} \leftarrow \alpha \left\| \mathbf{D_t} - \mathbf{W_i^t} \right\| h_\eta(i, u, \mathbf{D_t}) \left(\mathbf{D_t} - \mathbf{W_i^t} \right) \tag{4.2}$$

$$h_\eta(i, u, \mathbf{D_t}) = \exp\left(-\frac{1}{\eta^2} \frac{\|\mathtt{P}(i) - \mathtt{P}(u)\|^2}{\|\mathbf{D_t} - \mathbf{W_u}\|^2} \right) \tag{4.3}$$

where α is the learning rate, i is the index of the currently updated neuron, $\mathbf{D_t}$ is the current data point, u is the index of the best matching unit, η is the elasticity parameter, h_η is the neighborhood function, and $\mathtt{P}(i)$, $\mathtt{P}(u)$ are respectively the positions of neurons i and u in the grid (not in the latent space). Intuitively, the distance between $\mathtt{P}(i)$ and $\mathtt{P}(u)$ is the minimal number of consecutive neighbors that form a path between i and u.

4.4 The Q-SOM and Q-DSOM algorithms

We take inspiration from Decentralized Partially-Observable Markovian Decision Processes (DecPOMDPs) to formally describe our proposed algorithms. DecPOMDPs are an extension of the well-known Markovian Decision Process (MDP) that considers multiple

agents taking repeated decisions in multiple states of an environment, by receiving only partial observations about the current state. In contrast with the original DecPOMDP as described by Bernstein (Bernstein, Givan, Immerman, & Zilberstein, 2002), we explicitly define the set of learning agents, and we assume that agents receive (different) individual rewards, instead of a team reward.

Definition 4.1 (DecPOMDP)**.** A Decentralized Partially-Observable Markovian Decision Process is a tuple $\langle \mathcal{L}, \mathbb{S}, \mathbb{A}, \mathtt{T}, \mathbb{O}, \mathtt{O}, \mathtt{R}, \gamma \rangle$, where:

- $\mathcal{L}$ is the set of learning agents, of size $n = |\mathcal{L}|$.
- $\mathbb{S}$ is the state space, i.e., the set of states that the environment can possibly be in. States are not directly accessible to learning agents.
- $\mathbb{A}_l$ is the set of actions accessible to agent l, $\forall l \in \mathcal{L}$ as all agents take individual actions. We consider multi-dimensional and continuous actions, thus we have $\mathbb{A}_l \subseteq \mathbb{R}^d$, with d the number of dimensions, which depends on the case of application.
- $\mathbb{A}$ is the action space, i.e., the set of joint-actions that can be taken at each time step. A joint-action is the combination of all agents' actions, i.e., $\mathbb{A} = \mathbb{A}_{l_1} \times \cdots \times \mathbb{A}_{l_n}$.
- $\mathtt{T}$ is the transition function, defined as $\mathtt{T} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$. In other words, $\mathtt{T}(s'|s, \mathbf{a})$ is the probability of obtaining state $\mathbf{s}'$ after taking the action $\mathbf{a}$ in state $\mathbf{s}$.
- $\mathbb{O}$ is the observation space, i.e., the set of possible observations that agents can receive. An observation is a partial information about the current state. Similarly to actions, we define $\mathbb{O}_l$ as the observation space for learning agent l, $\forall l \in \mathcal{L}$. As well as actions, observations are multi-dimensional and continuous, thus we have $\mathbb{O}_l \subseteq \mathbb{R}^g$, with g the number of dimensions, which depends on the use case.
- $\mathtt{O}$ is the observation probability function, defined as $\mathtt{O} : \mathbb{O} \times \mathbb{S} \times \mathbb{A} \to [0, 1]$, i.e., $\mathtt{O}(\mathbf{o}|s', \mathbf{a})$ is the probability of receiving the observations $\mathbf{o}$ after taking the action $\mathbf{a}$ and arriving in state s'.
- $\mathtt{R}$ is the reward function, defined as $\forall l \in \mathcal{L} \quad \mathtt{R}_l : \mathbb{S} \times \mathbb{A}_l \to \mathbb{R}$. Typically, the reward function itself will be the same for all agents, however, agents are rewarded individually, based on their own contribution to the environment through their action. In other words, $\mathtt{R}_l(s, \mathbf{a}_l)$ is the reward that learning agent l receives for taking action $\mathbf{a}_l$ in state s.
- γ is the discount factor, to allow for potentially infinite horizon of time steps, with $\gamma \in [0, 1[$.

The RL algorithm must learn a stochastic strategy π_l, defined as $\pi_l : \mathbb{O}_l \times \mathbb{A}_l \rightarrow [0, 1]$. In other words, given the observations $\mathbf{o}_l$ received by an agent l, $\pi(\mathbf{o}_l, \mathbf{a})$ is the probability that agent l will take action $\mathbf{a}$.

We recall that observations and actions are vectors of floating numbers, the RL algorithm must therefore handle this accordingly. However, it was mentioned in Section 4.2 that the Q-Table is not suitable for continuous data. To solve this, we take inspiration from an existing work (Smith, 2002a, 2002b) and propose to use variants of Self-Organizing Maps (SOMs) (Kohonen, 1990).

We can leverage SOMs to learn to handle the observation and action spaces: neurons learn the topology of the latent space and create a discretization. By associating each neuron with a unique index, we are able to discretize the multi-dimensional data: each data point is recognized by the neuron with the closest prototype vector, and thus is represented by a discrete identifier, i.e., the neuron's index.

The proposed algorithms are thus based on 2 (Dynamic) SOMs, a State-SOM, and an Action-SOM, which are associated to a Q-Table. To navigate the Q-Table and access the Q-Values, we use discrete identifiers obtained from the SOMs. The Q-Table's dimensions thus depend on the (D)SOMs' number of neurons: the Q-Table has exactly as many rows as the State-(D)SOM has neurons, and exactly as many columns as the Action-(D)SOM has neurons, such that each neuron is represented by a row or column, and reciprocally.

Our algorithms are separated into 2 distinct parts: the *decision* process, which chooses an action from received observations about the environment, and the *learning* process, which updates the algorithms' data structures, so that the next decision step will yield a better action. We present in details these 2 parts below.

4.4.1 The decision process

We now explain the decision process that allows an agent to choose an action from received observations, which is described formally in Algorithm 4.1 and represented in Figure 4.3. First, we need to obtain a discrete identifier from an observation $\mathbf{o}$ that is a vector $\in \mathbb{O}_l \subseteq \mathbb{R}^g$, in order to access the Q-Table. To do so, we look for the Best Matching Unit (BMU), i.e., the neuron whose prototype vector is the closest to the observations, from the State-SOM, which is the SOM tasked with learning the observation space. The unique index of the BMU is used as the state identifier s (line 2).

Algorithm 4.1: Decision algorithm

Function `decision()`:

Data:

$\mathcal{U}$ the neurons in the State-(D)SOM

$\mathbf{U_i}$ the vector associated to neuron i in the State-(D)SOM

$\mathcal{W}$ the neurons in the Action-(D)SOM

$\mathbf{W_j}$ the vector associated to neuron j in the Action-(D)SOM

$\mathtt{Q}(s, a)$ the Q-value of action a in state s

τ the Boltzmann's temperature

ϵ Noise control parameter

Input: Observations $\mathbf{o}$

Output: An action $\mathbf{a}$

`/* Determine the Best Matching Unit, closest neuron from the State-SOM to the observations */`

$s \leftarrow \text{argmin}_{\forall i \in \mathcal{U}} ||\mathbf{o} - \mathbf{U_i}||$

`/* Choose action identifier using Boltzmann probabilities */`

Let P be the Boltzmann distribution over the Q-Values. We draw a random variable X from P, and we denote the probability that X equals a given value j : $P(X = j)$.

Draw $j \sim P(X = j) = \frac{\frac{exp(\mathtt{Q}(s,j))}{\tau}}{\sum_{k=1}^{|\mathcal{W}|} \frac{exp(\mathtt{Q}(s,k))}{\tau}}$

Let $\mathbf{W_j}$ be the chosen action's parameters

`/* Randomly noise the action's parameters to explore */`

for $k \in$ *all dimensions of* $\mathbf{W_j}$ **do**

`/* The random distribution can be either uniform or normal */`

$noise \sim \mathtt{random}(\epsilon)$

$W'_{j,k} \leftarrow W_{j,k} + noise$

end

return a $\leftarrow \mathbf{W'_j}$

end

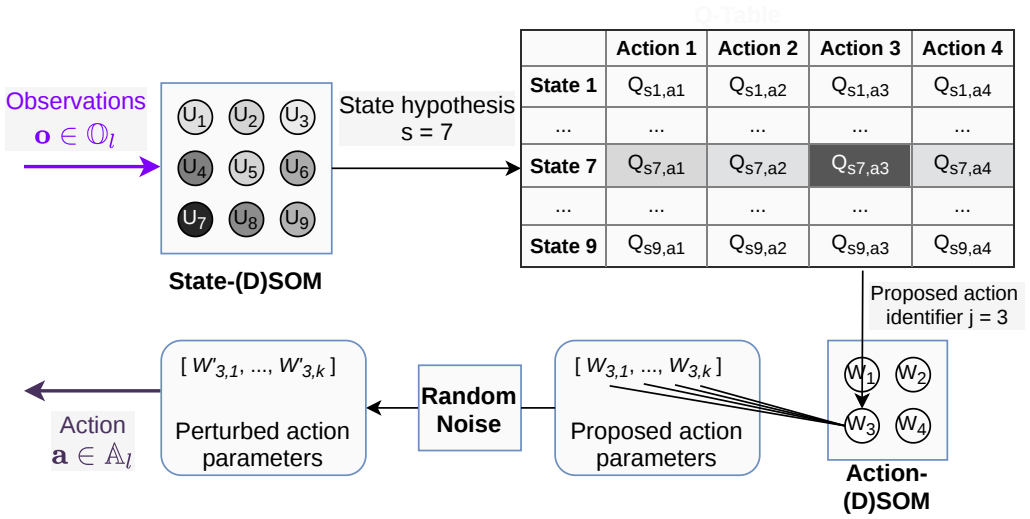


Fig. 4.3.: Dataflow of the Q-(D)SOM decision process.

We call this identifier a "state hypothesis", and we use it to navigate the Q-Table and obtain the expected interest of each action, assuming we have correctly identified the state. Knowing these interests $\mathtt{Q}(s,.)$ for all actions, we can assign a probability of taking each one, using a Boltzmann distribution (line 3). Boltzmann is a well-known and used method in RL that helps with the exploration-exploitation dilemma. Indeed, as we saw in Section 2.2, agents should try to maximize their expectancy of received rewards, which means they should *exploit* high-rewarding actions, i.e., those with a high interest. However, the true interest of the action is not known to agents: they have to discover it incrementally by trying actions into the environment, in various situations, and memorizing the associated reward. If they only choose the action with the maximum interest, they risk focusing on few actions, thus not exploring the others. By not sufficiently exploring, they maintain the phenomenon, as not explored actions will stay at a low interest, reducing their probability of being chosen, and so on. Using Boltzmann mitigates this problem, by giving similar probabilities to similar interests, and yet, a non-zero probability of being chosen even for actions with low interests.

The Boltzmann probability of an action j being selected is computed based on the action's interest, in the current state, relatively to all other actions' interests, as follows:

$$P(X=j) = \frac{\frac{\exp(\mathtt{Q}(s,j))}{\tau}}{\sum_{k=1}^{|\mathcal{W}|} \frac{\exp(\mathtt{Q}(s,k))}{\tau}} \tag{4.4}$$

Traditionally, the Boltzmann parameter τ should be decreasing over the time steps, such that the probabilities of high-interest actions will rise, whereas low-interest actions will converge towards a probability of 0. This mechanism ensures the convergence of the agents' policy towards the optimal one, by reducing exploration in later steps, in favour of exploitation. However, and as we have already mentioned, we chose to disable the convergence mechanisms in our algorithms, because it prevents, by principle, continuous learning and adaptation.

We draw an action identifier j from the list of possible actions, according to Boltzmann probabilities (line 4). From this discrete identifier, we get the action's parameters from the Action-SOM, which is tasked with learning the action space. We retrieve the neuron with identifier j, and take its prototype vector as the proposed action's parameters (line 5).

We can note that this is somewhat symmetrical to what is done with the State-SOM. To learn the State-SOM, we use the data points, i.e., the observations, that come from the environment; to obtain a discrete identifier, we take the neurone with the closest prototype. For the Action-SOM, we start with a discrete identifier, and we take the prototype of the neuron with this identifier. However, we need to learn what are those prototype vectors. We do not have data points as for the State-SOM, since we do not know what is the "correct" action in each situation. In order to learn better actions, we apply an exploration step after choosing an action: the action's parameters are perturbed by a random noise (lines 6-9).

In the original work of Smith (2002a), the noise was taken from a uniform distribution $\mathcal{U}_{[-\epsilon,+\epsilon]}$, which we will call the *epsilon* method in our experiments. However, in our algorithms, we implemented a normal, or *gaussian*, random distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the mean, which we set to 0 so that the distribution ranges over both negative and positive values, σ^2 is the variance, and σ is the standard deviation. ϵ and σ^2 are the "noise control parameter" for their respective distribution. The advantage over the uniform distribution is to have a higher probability of a small noise, thus exploring very close actions, while still allowing for a few rare but longer "jumps" in the action space. These longer jumps may help to escape local extremas, but should be rare, so as to slowly converge towards optimal actions most of the time, without overshooting them. This was not permitted by the uniform distribution, as the probability is the same for each value in the range $[-\epsilon, +\epsilon]$.

The noised action's parameters are considered as the chosen action by the decision process, and the agent executes this action in the environment (line 10).

4.4.2 The learning process

After all agents executed their action, and the environment simulated the new state, agents receive a reward signal which indicates to which degree their action was a "good one". From this reward, agents should improve their behaviour so that their next choice will be better. The learning process that makes this possible is formally described in Algorithm 4.2, and we detail it below. First, we update the Action-(D)SOM. Remember that we do not have the ground-truth for actions: we do not know which parameters yield the best rewards. Moreover, we explored the action space by randomly noising the proposed action; it is possible that the perturbed action is actually worse than the learned one. In this case, we do not want to update the Action-(D)SOM, as this would worsen the agent's performances. We thus determine whether the perturbed action is better than the proposed action by comparing the received reward with the memorized interest of the proposed action, using the following equation:

$$r + \gamma \max_{j'} \mathtt{Q}(s', j') \overset{?}{>} \mathtt{Q}(s, j) \tag{4.5}$$

If the perturbed action is deemed better than the proposed one, we update the Action-(D)SOM towards the perturbed action (lines 4-8). To do so, we assume that the Best Matching Unit (BMU), i.e., the center of the neighborhood, is the neuron that was selected at the decision step, j (line 3). We then apply the corresponding update equation, Equation (4.1) for a SOM, or Equation (4.2) for a DSOM, to move the neurons' prototypes towards the perturbed action.

Secondly, we update the actions' interests, i.e., the Q-Table (line 9). To do so, we rely on the traditional Bellman's equation that we presented in the State of the Art: Equation (2.1). However, Smith's algorithm introduces a difference in this equation to increase the learning speed. Indeed, the State- and Action-(D)SOMs offer additional knowledge about the states and actions: as they are discrete identifiers mapping to continuous vectors in a latent space, we can define a notion of *similarity* between states (resp. actions) by measuring the distance between the states' vectors (resp. actions' vectors). Similar states and actions will most likely have a similar interest, and thus each Q-Value is updated at each time step, instead of only the current state-action pair, by taking into account the

Algorithm 4.2: Learning algorithm

Function `learning()`:

Data:

$\mathcal{U}$ the neurons in the State-(D)SOM
$\mathbf{U_u}$ the vector associated to neuron u in the State-(D)SOM
$\mathcal{W}$ the neurons in the Action-(D)SOM
$\mathbf{W_w}$ the vector associated to neuron w in the Action-(D)SOM
$\mathtt{P}_U(u)$ is the position of neuron u in the State-(D)SOM grid
$\mathtt{P}_W(w)$ is the position of neuron w in the Action-(D)SOM grid
$\mathtt{Q}(s,a)$ the Q-value of action a in state s
η_U, η_W elasticity for State- and Action-(D)SOMs
$\alpha_Q, \alpha_U, \alpha_W$ learning rates for Q-Table, State-, Action-(D)SOMs
γ the discount factor

Input:

Previous observations $\mathbf{o}$
New observations $\mathbf{o}'$
Received reward r
State hypothesis s
Chosen action identifier j
Chosen action parameters $\mathbf{a}$

`/* Compute the neighborhood of neurons */`

$\forall u \in \mathcal{U} \quad \psi_U(u) \leftarrow exp\left(-\frac{1}{\eta_U^2} \frac{\|\mathtt{P}_U(u) - \mathtt{P}_U(s)\|}{\|\mathbf{o} - \mathbf{U_u}\|}\right)$

$\forall w \in \mathcal{W} \quad \psi_W(w) \leftarrow exp\left(-\frac{1}{\eta_W^2} \frac{\|\mathtt{P}_W(w) - \mathtt{P}_W(j)\|}{\|\mathbf{a} - \mathbf{W_w}\|}\right)$

`/* If the action was interesting */`

if $r + \gamma \max_{j'} \mathtt{Q}(s', j') \overset{?}{>} \mathtt{Q}(s, j)$ **then**

`/* Update the Action-(D)SOM */`

forall *neuron* $w \in \mathcal{W}$ **do**

$\mathbf{W_w} \leftarrow \alpha_W \|\mathbf{a} - \mathbf{W_w}\| \psi_W(w) (\mathbf{a} - \mathbf{W_w}) + \mathbf{W_w}$

end

end

`/* Update the Q-Table */`

$\mathtt{Q}(s,j) \leftarrow \alpha_Q \psi_U(s) \psi_W(j) \left[r + \gamma \max_{j'} \mathtt{Q}(i', j') - \mathtt{Q}(s,j)\right] + \mathtt{Q}(s,j)$

`/* Update the State-(D)SOM */`

forall *neuron* $u \in \mathcal{U}$ **do**

$\mathbf{U_u} \leftarrow \alpha_U \|\mathbf{o} - \mathbf{U_u}\| \psi_U(u) (\mathbf{o} - \mathbf{U_u}) + \mathbf{U_u}$

end

end

neighborhoods of the State- and Action-(D)SOMs (computed on lines 2 and 3). Equation (4.6) shows the resulting formula:

$$\mathtt{Q}_{t+1}(s,j) \leftarrow \alpha\psi_U(s)\psi_W(j)\left[r + \gamma \max_{j'} \mathtt{Q}_t(s',j')\right] + (1-\alpha)\mathtt{Q}_t(s,j) \tag{4.6}$$

where s was the state hypothesis at step t, j was the chosen action identifier, r is the received reward, s' is the state hypothesis at step $t+1$ (from the new observations). $\psi_U(s)$ and $\psi_W(j)$ represent, respectively, the neighborhood of the State- and Action-(D)SOMs, centered on the state s and the chosen action identifier j. Intuitively, the equation takes into account the interest of arriving in this new state, based on the maximum interest of actions available in the new state. This means that an action could yield a medium reward by itself, but still be very interesting because it allows to take actions with higher interests. On the contrary, an action with a high reward, but leading to a state with only catastrophic actions would have a low interest.

Finally, we learn the State-SOM, which is a very simple step (lines 10-12). Indeed, we have already mentioned that we know data points, i.e., observations, that have been sampled from the distribution of states by the environment. Therefore, we simply update the neurons' prototypes towards the received observation at the previous step. Prototype vectors are updated based on both their own distance to the data point, within the latent space, and the distance between their neuron and the best matching unit, within the 2D grid neighborhood (using the neighborhood computed on line 2). This ensures that the State-SOM learns to represent states which appear in the environment.

Remark. In the presented algorithm, the neighborhood and update formulas correspond to a DSOM. When using the Q-SOM algorithm, these formulas must be replaced by their SOM equivalents. The general structure of the algorithm, i.e., the steps and the order in which they are taken, stays the same.

Remark. Compared to Smith's algorithm, our extensions differ in the following aspects:

- DSOMs can be used in addition to SOMs.
- Hyperparameters are not annealed, i.e., they are constant throughout the simulation, so that agents can continuously learn instead of slowly converging.
- Actions are chosen through a Boltzmann distribution of probabilities based on their interests, instead of using the ϵ-greedy method.
- The random noise to explore the actions' space is drawn from a Gaussian distribution instead of a uniform one.

- The neighborhood functions of the State- and Action-(D)SOMs is a gaussian instead of a linear one.
- The number of dimensions of the actions' space in the following experiments is greater (6) than in Smith's original experiments (2). This particularly prompted the need to explore other ways to randomly noise actions, e.g., the gaussian distribution. Note that some other methods have been tried, such as applying a noise on a single dimension each step, or randomly determining for each dimension whether it should be noised at each step; they are not disclosed in the results as they performed slightly below the gaussian method. Searching for better hyperparameters could yield better results for these methods.

4.5 Experiments

In order to validate our proposed algorithms, we ran some experiments on our Smart Grid use-case that we presented in 3.4.

First, let us apply the algorithms and formal model on this specific use-case. The observation space, $\mathbb{O}$, is composed of the information that agents receive: the time (hour), the available energy, their personal battery storage, ... The full list of observations was defined in Section 3.4.4. These values range from 0 to 1, and we have 11 such values, thus we define $\mathbb{O}_l = [0, 1]^{11}$.

Similarly, actions are defined by multiple parameters: consume energy from grid, consume from battery, sell, ... These actions were presented in Section 3.4.3. To simplify the learning of actions, we constrain these parameters to the $[0, 1]$ range; they are scaled to the true agent's action range outside the learning and decision processes. For example, let us imagine an agent with an action range of $6,000$, and an action parameter, as outputted by the decision process, of 0.5, the scaled action parameter will be $0.5 \times 6,000 = 3,000$. We have 6 actions parameters, and thus define $\mathbb{A}_l = [0, 1]^6$.

In the sequel, we present the reward functions that we implemented to test our algorithms, as well as the experiments' scenarii. Finally, we quickly describe the 2 algorithms that we chose as baselines: *DDPG* and *MADDPG*.

4.5.1 Reward functions

We implemented multiple reward functions that each focus on different ethical stakes. Most of them are based on the principle of Difference Reward (Yliniemi & Tumer, 2014b) to facilitate the Credit Assignment, as discussed in Section 2.3.1. Additionally, 2 functions focus on multiple objectives, but with a rather naïve approach (see our next contributions for a better approach), and another 2 focus on adaptation, i.e., the agents' capacity to adapt their behaviour to changing mores, by making the reward function artificially change at a fixed point in time.

We give an intuitive definition and a mathematical formula for each of these reward functions below.

Equity Determine the agent's contribution to the society's equity, by comparing the current equity with the equity if the agent did not act. The agent's goal is thus to maximize the society's equity.

$$\mathtt{R}_{eq}(agent) = (1 - \mathtt{Hoover}(Comforts)) - (1 - \mathtt{Hoover}(Comforts \setminus \{agent\}))$$

Over-Consumption Determine the agent's contribution to over-consumption, by comparing the current over-consumed amount of energy, with the amount that would have been over-consumed if the agent did not act. The agent's goal is thus to minimize society's over-consumption.

$$\mathtt{R}_{oc}(agent) = 1 - \frac{OC}{\sum_{\forall a}(Consumed_a + Stored_a)} - \frac{OC - (Consumed_{agent} + Stored_{agent})}{\sum_{\forall a \neq agent}(Consumed_a + Stored_a)}$$

Comfort Simply return the agent's comfort, so that agents aim to maximize their comfort. This intuitively does not seem like an ethical stake, however it can be linked to Schwartz' "hedonistic" value, and therefore is an ethical stake, focused on the individual aspect. We will mainly use this reward function in combination with others that focus on the societal aspect, to demonstrate the algorithms' capacity to learn opposed moral values.

$$\mathtt{R}_{comfort}(agent) = Comforts_{agent}$$

Multi-Objective Sum A first and simple reward function that combines multiple objectives, namely limitation of over-consumption and comfort. The goal of agents is thus to both minimize the society's over-consumption while maximizing their own comfort. This may be a difficult task, because the simulation is designed so that there is a scarcity of energy most of the time, and agents will most likely over-consume if they all try to maximize their comfort. On the contrary, reducing the over-consumption means they need to diminish their comfort. There is thus a trade-off to be achieved between over-consumption and comfort.

$$\mathtt{R}_{mos}(agent) = 0.8 \times \mathtt{R}_{oc}(agent) + 0.2 \times \mathtt{R}_{comfort}(agent)$$

Multi-Objective Product A second, but also simple, multi-objective reward functions. Instead of using a weighted sum, we multiply the reward together. This function is more punitive than the sum, as a low reward cannot be "compensated". For example, let us consider a vector of reward components $[0.1, 0.9]$. Using the weighted sum, the result depends on the weights: if the first component has a low coefficient, then the result may actually be high. On contrary, the product will return $0.1 \times 0.9 = 0.09$, i.e., a very low reward. Any low component will penalize the final result.

$$\mathtt{R}_{mop}(agent) = \mathtt{R}_{oc}(agent) \times \mathtt{R}_{comfort}(agent)$$

Adaptability1 A reward function that simulates a change in its definition after 2000 time steps, as if society's ethical mores had changed. During the first 2000 steps, it behaves similarly as the Over-Consumption reward function, whereas for later steps it returns the mean of Over-Consumption and Equity rewards.

$$\mathtt{R}_{ada1}(agent) = \begin{cases} \mathtt{R}_{oc}(agent) & \text{if } t < 2000 \\ \frac{\mathtt{R}_{oc}(agent) + \mathtt{R}_{eq}(agent)}{2} & \text{else} \end{cases}$$

Adaptability2 Similar to Adaptability1, this function simulates a change in its definition. We increase the difficulty by making 2 changes, one after 2000 time steps, and another after 6000 time steps, and by considering a combination of 3 rewards after the second change.

$$\mathtt{R}_{ada2}(agent) = \begin{cases} \mathtt{R}_{oc}(agent) & \text{if } t < 2000 \\ \frac{\mathtt{R}_{oc}(agent) + \mathtt{R}_{eq}(agent)}{2} & \text{else if } t < 6000 \\ \frac{\mathtt{R}_{oc}(agent) + \mathtt{R}_{eq}(agent) + \mathtt{R}_{comfort}(agent)}{3} & \text{else} \end{cases}$$

As we can see, the various reward functions have different aims. Some simple functions, such as *equity*, *overconsumption*, or *comfort*, serve as a baseline and building blocks for other functions. Nevertheless, they may be easy to optimize: for example, by consuming absolutely nothing, the *overconsumption* function can be satisifed. On the contrary, the *comfort* function can be satisfied by consuming the maximum amount of energy, such that the comfort is guaranteed to be close to 1. The 2 *multi-objective* functions thus try to force agents to learn several stakes at the same time, especially if they are contradictory, such as *overconsumption* and *comfort*. The agent thus cannot learn a "trivial" behaviour and must find the optimal behaviour that manages to satisfy both as much as possible. Finally, the *adaptability* functions go a step further and evaluate agents' ability to adapt when the considerations change.

4.5.2 Scenarii

In order to improve the richness of our experiments, we designed several scenarii. These scenarii are defined by 2 variables: the agents' consumption profile, and the environment's size, i.e., number of agents.

We recall from Section 3.4.6 that learning agents are instantiated with a profile, determining their battery capacity, their action range, and their needs, i.e., the quantity of energy they want to consume at each hour. These needs are extracted from real consumption profiles; we propose 2 different versions, the *daily* and the *annual* profiles. In the *daily* version, needs are averaged over every day of the year, thus yielding a need for each hour of a day: this is illustrated in Figure 4.4. This is a simplified version, averaging the seasonal differences; its advantages are a reduced size, thus decreasing the required computational resources, a simpler learning, and an easier visualization for humans. On the other hand, the *annual* version is more complete, contains seasonal differences, which improve the environment's richness and force agents to adapt to important changes.

The second property is the environment size. We wanted to test our algorithms with different sets of agents, to ensure the scalability of the approach, in the sense that agents are able to learn a correct behaviour and adapt to many other agents in the same environment. This may be difficult as the number of agents increases, since there will most certainly be more conflicts. We propose a first, *small* environment, containing 20 Households agents, 5 Office agents, and 1 School agent. The second environment,

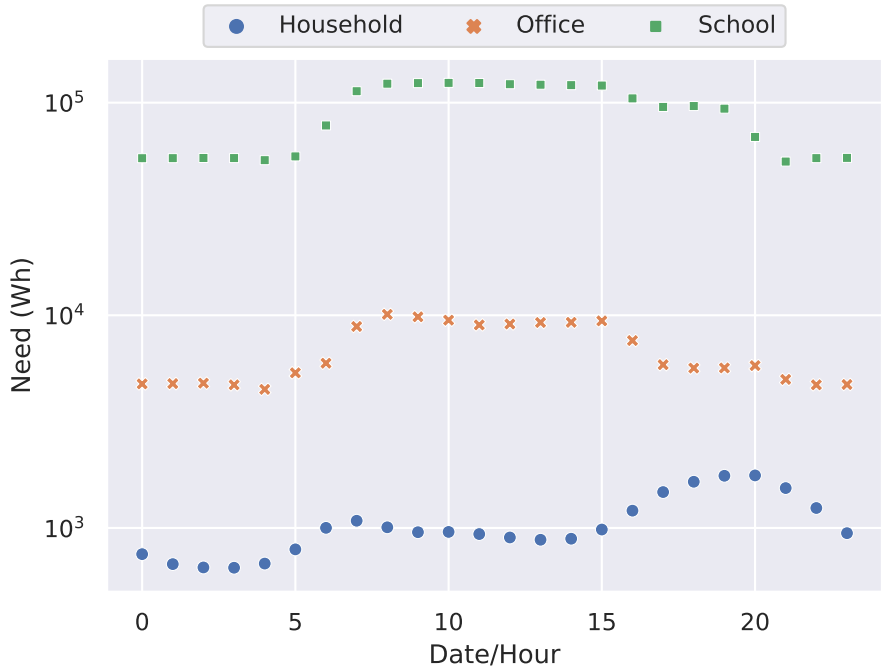


Fig. 4.4.: The agents' needs for every hour of the day in the *daily* profile.

medium, contains roughly 4 times more agents than in the *small* case: 80 Household agents, 19 Office agents, and 1 School.

4.5.3 DDPG and MADDPG baselines

In order to prove our algorithms' advantages, we chose to compare them to the well-known DDPG (Lillicrap et al., 2015) and its multi-agent extension, MADDPG (Lowe et al., 2017).

DDPG, or *Deep Deterministic Policy Gradient* is one of the algorithms that extended the success of Deep Reinforcement Learning to continuous domains (Lillicrap et al., 2015). It follows the quite popular Actor-Critic architecture, which uses 2 different Neural Networks: one for the Actor, i.e., to decide which action to perform at each time step, and another for the Critic, i.e., to evaluate whether an action is interesting. We chose it as a baseline since it focuses on problems with similar characteristics, e.g., continuous domains, and is a popular baseline in the community.

MADDPG, or *Multi-Agent Deep Deterministic Policy Gradient*, extends the idea of DDPG to the multi-agent setting (Lowe et al., 2017), by relying on the Centralized Training idea. As we mentioned in Section 2.3.2, it is one of the most used methods to improve multi-agent learning, by sharing data among agents during the learning phase. This helps agents make a model of other agents and adapt to their respective behaviours. However, during execution, sharing data in the same manner is often impracticable or undesirable, as it would impair privacy and require some sort of communication between agents; thus, data is not shared any more at this point (Decentralized Execution). We recall that, as such, Centralized Training - Decentralized Execution makes a distinction between training and execution, and is thus inadequate for continuous learning, and constant adaptation to changes. On the other hand, if we were to make agents continuously learn with centralized data sharing, even in the execution phase, we would impair privacy of users that are represented or impacted by the agents. These reasons are why we chose not to use this setting for our own algorithms Q-SOM and Q-DSOM. While we do not use centralized training, we want to compare them to an algorithm that uses it, such as MADDPG, in order to determine whether there would be a performance gain, and what would be the trade-off between performance and privacy. In MADDPG, the Centralized Training is simply done by using a centralized Critic network, which receives observations, actions, and rewards from all agents, and evaluates all agents' actions. The Actor networks, however, are still individualized: each agent has its own network, which the other agents cannot access. During the training phase, the Critic network is updated thanks to the globally shared data, whereas Actor networks are updated through local data and the global Critic. Once the learning is done, the networks are frozen: the Critic does not require receiving global data any more, and the Actors do not rely on the Critic any more. Only the decision part, i.e., which action should we do, is kept, by using the trained Actor network as-is.

4.6 Results

Several sets of experiments were performed:

- First, numerous experiments were launched to search for the best hyperparameters of each algorithm, to ensure a fair comparison later. Each set of hyperparameters was run 10 times to obtain average results, and a better statistical significance. In order to limit the number of runs and thus the computational resources required, we

decided to focus on the *adaptability2* reward for these experiments. This function is difficult enough so that the algorithms will not reach almost 100% immediately, which would make the hyperparameter search quite useless, and is one of the 2 that interest us the most, along with *adaptability1*, so it makes sense that our algorithms are optimized for this one. The *annual* consumption profile was used to increase the richness, but the environment size, i.e., number of agents, was set to *small* in order to once again reduce the computational power and time.

- Then, the 4 algorithms, configured with their best hyperparameters, were compared on multiple settings: both *annual* and *daily* consumption profiles, both *small* and *medium* sizes of environment, and all the reward functions. This resulted in $2 \times 2 \times 7$ scenarii, which we ran 10 times for each of the 4 algorithms.

In the following results, we define a run's *score* as the average of the global rewards per step. The global reward corresponds to the reward, without focusing on a specific agent. For example, the *equity* reward compares the Hoover index of the whole environment to a hypothetical environment without the agent. The global reward, in this case, is simply the Hoover index of the entire environment. This represents, intuitively, how the society of agents performed, globally. Taking the average is one of the simplest methods to get a single score for a given run, which allows comparing runs easily.

4.6.1 Searching for hyperparameters

Tables 4.1, 4.2, 4.3, 4.4 summarize the best hyperparameters that have been found for each algorithm, based on the average runs' score obtained when using these parameters.

Tab. 4.1.: Best hyperparameters on 10 runs for the *Q-SOM* algorithm, using the *annual small* scenario and *adaptability2* reward function.

Parameter	Value	Description
State-SOM shape	12x12	Shape of the neurons' grid
State-SOM learning rate	0.5	Update speed of State-SOM neurons
Action-SOM shape	3x3	Shape of the neurons' grid
Action-SOM learning rate	0.2	Update speed of Action-SOM neurons

Tab. 4.1.: Best hyperparameters on 10 runs for the *Q-SOM* algorithm, using the *annual small* scenario and *adaptability2* reward function.

Parameter	Value	Description
Q Learning rate	0.6	Update speed of Q-Values
Discount rate	0.9	Controls the horizon of rewards
Action perturbation	gaussian	Method to randomly explore actions
Action noise	0.06	Parameter for the random noise distribution
Boltzmann temperature	0.4	Controls the exploration-exploitation

Tab. 4.2.: Best hyperparameters on 10 runs for the *Q-DSOM* algorithm, using the *annual small* scenario and *adaptability2* reward function.

Parameter	Value	Description
State-DSOM shape	12x12	Shape of the neurons' grid
State-DSOM learning rate	0.8	Update speed of State-DSOM neurons
State-DSOM elasticity	1	Coupling between State-DSOM neurons
Action-DSOM shape	3x3	Shape of the neurons' grid
Action-DSOM learning rate	0.7	Update speed of Action-DSOM neurons
Action-DSOM elasticity	1	Coupling between Action-DSOM neurons
Q Learning rate	0.8	Update speed of Q-Values
Discount rate	0.95	Controls the horizon of rewards
Action perturbation	gaussian	Method to randomly explore actions
Action noise	0.09	Parameter for the random noise distribution
Boltzmann temperature	0.6	Controls the exploration-exploitation

Tab. 4.3.: Best hyperparameters on 10 runs for the *DDPG* algorithm, using the *annual small* scenario and *adaptability2* reward function.

Parameter	Value	Description
Batch size	256	Number of samples to use for training at each step
Learning rate	5e-04	Update speed of neural networks
Discount rate	0.99	Controls the horizon of rewards
Tau	5e-04	Target network update rate
Action perturbation	gaussian	Method to randomly explore actions
Action noise	0.11	Parameter for the random noise distribution

Tab. 4.4.: Best hyperparameters on 10 runs for the *MADDPG* algorithm, using the *annual small* scenario and *adaptability2* reward function.

Parameter	Value	Description
Batch size	128	Number of samples to use for training at each step
Buffer size	50000	Size of the replay memory.
Actor learning rate	0.01	Update speed of the Actor network
Critic learning rate	0.001	Update speed of the Critic network
Discount rate	0.95	Controls the horizon of rewards
Tau	0.001	Target network update rate
Noise	0.02	Controls a gaussian noise to explore actions
Epsilon	0.05	Controls the exploration-exploitation

4.6.2 Comparing algorithms

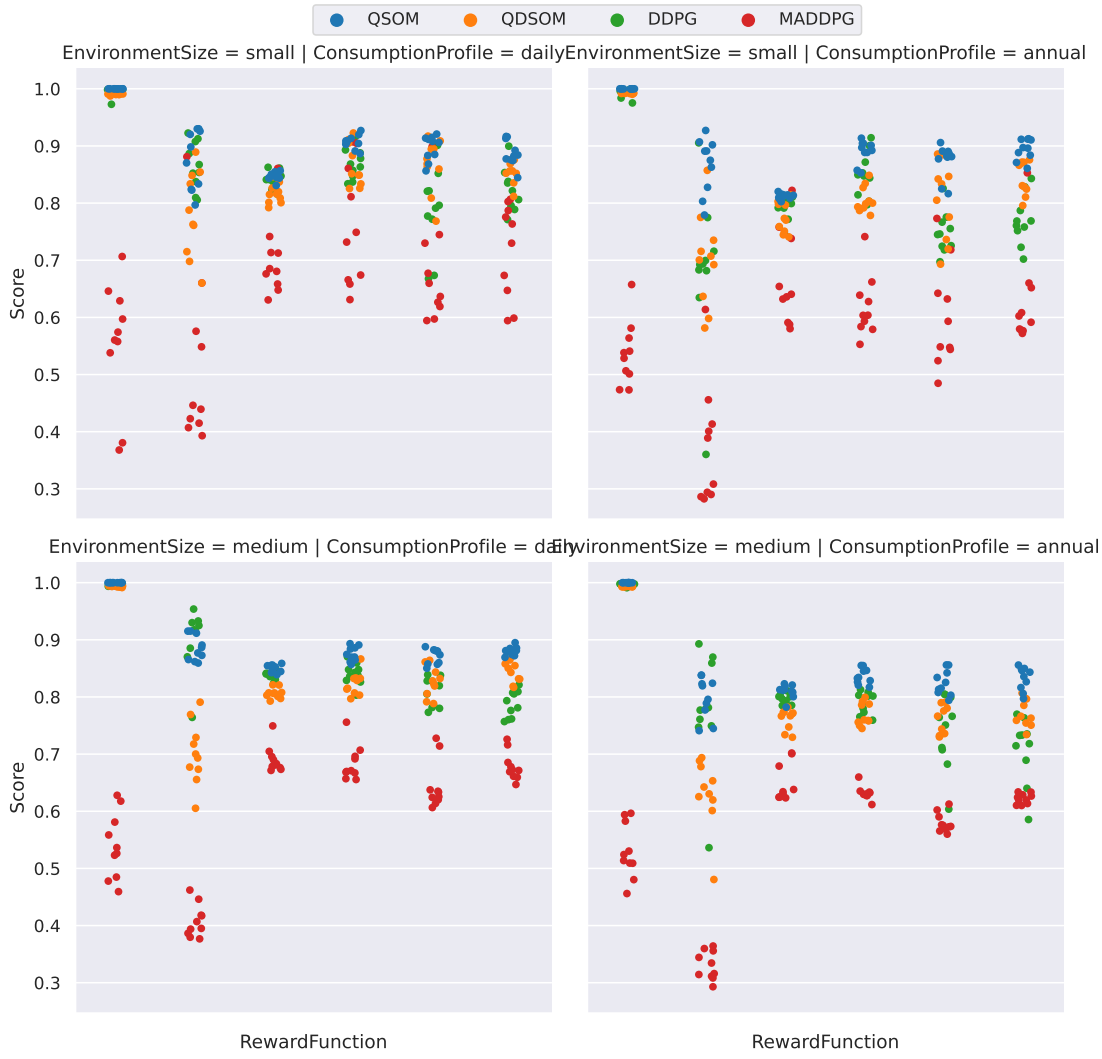


Fig. 4.5.: Distribution of scores per learning algorithm, on every scenario, for 10 runs with each reward function.

Tab. 4.5.: Average score for 10 runs of each algorithm, on each reward function and each scenario. The standard deviation is shown inside parentheses.

RewardFunction	QSOM	QDSOM	DDPG	MADDPG
Scenario: daily / small				
equity	1.00 (± 0.00)	0.99 (± 0.00)	0.99 (± 0.01)	0.56 (± 0.11)
overconsumption	0.88 (± 0.05)	0.78 (± 0.08)	0.87 (± 0.04)	0.52 (± 0.15)
multiobj-sum	0.91 (± 0.01)	0.87 (± 0.04)	0.87 (± 0.03)	0.76 (± 0.11)
multiobj-prod	0.85 (± 0.01)	0.82 (± 0.02)	0.84 (± 0.01)	0.70 (± 0.07)
adaptability1	0.90 (± 0.02)	0.87 (± 0.05)	0.79 (± 0.07)	0.68 (± 0.09)
adaptability2	0.89 (± 0.02)	0.86 (± 0.02)	0.82 (± 0.04)	0.72 (± 0.08)
Scenario: daily / medium				
equity	1.00 (± 0.00)	0.99 (± 0.00)	1.00 (± 0.00)	0.54 (± 0.06)
overconsumption	0.89 (± 0.02)	0.70 (± 0.05)	0.90 (± 0.05)	0.41 (± 0.03)
multiobj-sum	0.88 (± 0.01)	0.82 (± 0.02)	0.84 (± 0.02)	0.68 (± 0.03)
multiobj-prod	0.85 (± 0.01)	0.81 (± 0.01)	0.84 (± 0.01)	0.69 (± 0.02)
adaptability1	0.87 (± 0.01)	0.83 (± 0.03)	0.81 (± 0.03)	0.64 (± 0.04)
adaptability2	0.88 (± 0.01)	0.84 (± 0.02)	0.79 (± 0.02)	0.68 (± 0.02)
Scenario: annual / small				
equity	1.00 (± 0.00)	0.99 (± 0.00)	0.99 (± 0.01)	0.54 (± 0.06)
overconsumption	0.87 (± 0.05)	0.70 (± 0.08)	0.68 (± 0.14)	0.37 (± 0.11)
multiobj-sum	0.89 (± 0.02)	0.81 (± 0.02)	0.85 (± 0.04)	0.62 (± 0.05)
multiobj-prod	0.81 (± 0.00)	0.78 (± 0.03)	0.79 (± 0.02)	0.66 (± 0.08)
adaptability1	0.87 (± 0.03)	0.80 (± 0.07)	0.75 (± 0.04)	0.60 (± 0.09)
adaptability2	0.89 (± 0.02)	0.84 (± 0.03)	0.77 (± 0.04)	0.63 (± 0.09)

Tab. 4.5.: Average score for 10 runs of each algorithm, on each reward function and each scenario. The standard deviation is shown inside parentheses.

RewardFunction	QSOM	QDSOM	DDPG	MADDPG
Scenario: annual / medium				
equity	1.00 (± 0.00)	0.99 (± 0.00)	1.00 (± 0.00)	0.53 (± 0.05)
overconsumption	0.80 (± 0.04)	0.63 (± 0.06)	0.78 (± 0.10)	0.33 (± 0.02)
multiobj-sum	0.84 (± 0.01)	0.77 (± 0.02)	0.79 (± 0.02)	0.63 (± 0.01)
multiobj-prod	0.81 (± 0.01)	0.76 (± 0.02)	0.80 (± 0.01)	0.65 (± 0.03)
adaptability1	0.82 (± 0.02)	0.76 (± 0.02)	0.74 (± 0.06)	0.58 (± 0.02)
adaptability2	0.83 (± 0.02)	0.77 (± 0.02)	0.71 (± 0.06)	0.62 (± 0.01)

The results presented in Figure 4.5 and Table 4.5 show that the Q-SOM algorithm performs better. We use the Wilcoxon statistical test, which is the non-parametric equivalent of the well-known T-test, to determine whether there is a statistically significant difference in the means of runs' scores between different algorithms. Wilcoxon's test, when used with the *greater* alternative, assumes as a null hypothesis that the 2 algorithms have similar means, or that the observed difference is negligible and only due to chance. The Wilcoxon method returns the *p-value*, i.e, the likelihood of the null hypothesis being true. When $p < \alpha = 0.05$, we say that it is more likely that the null hypothesis can be refuted, and we assume that the alternative hypothesis is the correct one. The alternative hypothesis, in this case, is that the Q-SOM algorithm obtains better results than its opposing algorithm. We thus compare algorithms 2-by-2, on each reward function and scenario.

The statistics, presented in Table 4.6, prove that the Q-SOM algorithm statistically outperforms other algorithms, in particular DDPG and MADDPG, on most scenarii and reward functions, except a few cases, indicated by the absence of $*$ next to the *p-value*. For example, DDPG obtains similar scores on the *daily / small overconsumption* and *multiobj-prod* cases, as well as *daily / medium overconsumption*, and *annual / medium overconsumption*. Q-DSOM is also quite on par with Q-SOM on the *daily / small adaptability1* case. Yet, MADDPG is consistently outperformed by Q-SOM.

Tab. 4.6.: Comparison of the *Q-SOM* algorithm with the others, using a Wilcoxon statistical test, with the 'greater' alternative.

	Wilcoxon's p-value (Q-SOM vs ...)		
RewardFunction	**QDSOM**	**DDPG**	**MADDPG**
Scenario: daily / small			
equity	5.41e-06***	5.41e-06***	5.41e-06***
overconsumption	0.005748 **	0.289371	0.000103***
multiobj-sum	0.011615 *	0.001045 **	0.000525***
multiobj-prod	0.000162***	0.071570	0.000752***
adaptability1	0.061503	6.50e-05***	6.50e-05***
adaptability2	0.001440 **	0.000363***	5.41e-06***
Scenario: daily / medium			
equity	5.41e-06***	5.41e-06***	5.41e-06***
overconsumption	5.41e-06***	0.973787	5.41e-06***
multiobj-sum	3.79e-05***	0.000162***	5.41e-06***
multiobj-prod	5.41e-06***	0.005748 **	5.41e-06***
adaptability1	0.000363***	5.41e-06***	5.41e-06***
adaptability2	1.08e-05***	5.41e-06***	5.41e-06***
Scenario: annual / small			
equity	5.41e-06***	5.41e-06***	5.41e-06***
overconsumption	3.79e-05***	0.000363***	5.41e-06***
multiobj-sum	5.41e-06***	0.009272 **	5.41e-06***
multiobj-prod	0.000363***	0.000525***	0.000752***

Signif. codes: 0 <= '***' < 0.001 < '**' < 0.01 < '*' < 0.05

Tab. 4.6.: Comparison of the *Q-SOM* algorithm with the others, using a Wilcoxon statistical test, with the 'greater' alternative.

	Wilcoxon's p-value (Q-SOM vs ...)		
RewardFunction	**QDSOM**	**DDPG**	**MADDPG**
adaptability1	0.007345 **	2.17e-05***	5.41e-06***
adaptability2	0.000363***	5.41e-06***	5.41e-06***
Scenario: annual / medium			
equity	5.41e-06***	5.41e-06***	5.41e-06***
overconsumption	5.41e-06***	0.342105	5.41e-06***
multiobj-sum	5.41e-06***	5.41e-06***	5.41e-06***
multiobj-prod	5.41e-06***	0.009272 **	5.41e-06***
adaptability1	5.41e-06***	0.000103***	5.41e-06***
adaptability2	3.79e-05***	5.41e-06***	5.41e-06***

Signif. codes: 0 <= '***' < 0.001 < '**' < 0.01 < '*' < 0.05

Figure 4.6 shows the evolution of individual rewards received by agents over the time steps, in the *annual / small* scenario, using the *adaptability2* reward function. We chose to focus on this combination of scenario and reward function as they are, arguably, the most interesting. *Daily* scenarii are perhaps too easy for the agents as they do not include as many variations as the *annual*; additionally, *small* scenarios are easier to visualize and explore, as they contain fewer agents than *medium* scenarios. Finally, the *adaptability2* is retained for the same arguments that made us choose it for the hyperparameters search. We show a moving average of the rewards in order to erase the small and local variations to highlight the larger trend of the rewards' evolution.

We can see from the results that the *small* scenarii seem to yield a slightly better score than the *medium* scenarii. Thus, agents are impacted by the increased number of other agents, and have difficulties learning the whole environment dynamics. Still, the results reported for the *medium* scenarii are near the *small* results, and very close to 1. Even

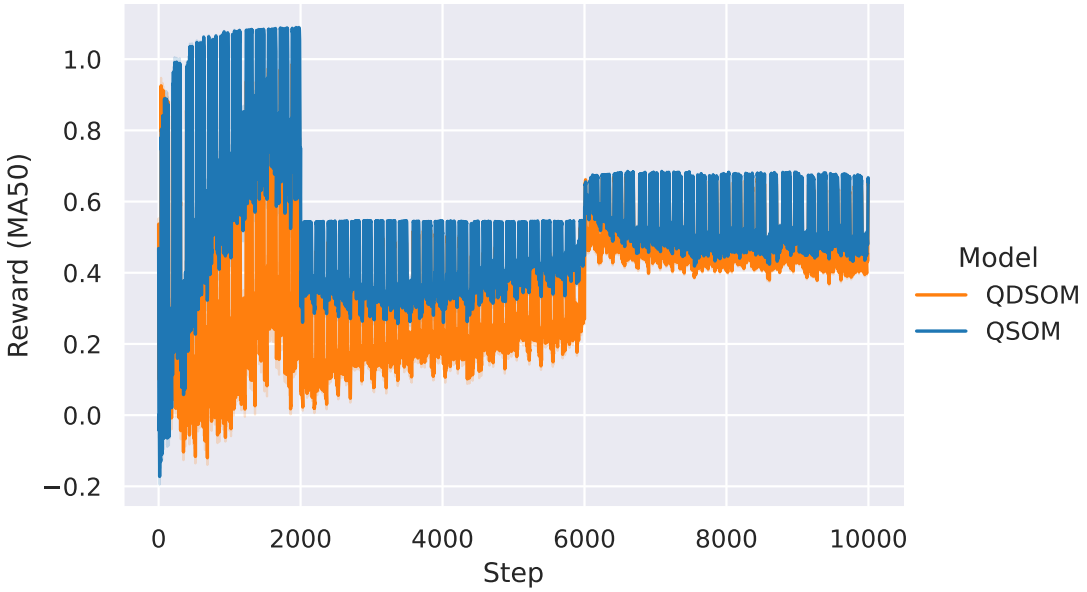


Fig. 4.6.: Agents' individual rewards received per time step, over 10 runs in the *annual / small* scenario, and using the *adaptability2* reward function. Rewards are averaged in order to highlight the trend.

though there is indeed an effect of the environment size on the score, this hints towards the scalability of our approach, as the agents managed to learn a "good" behaviour that yields high rewards.

4.7 Discussion

In this chapter, we presented our first contribution consisting in two reinforcement learning algorithms, Q-SOM and Q-DSOM, which aimed to answer the following research question: How to learn behaviours aligned with moral values, using Reinforcement Learning with complex, continuous, and multi-dimensional representations of actions and situations? How to make the multiple learning agents able to adapt their behaviours to changes in the environment?

We recall that the principal important aspects and limitations identified in the State of the Art, and which pertains to our 1st research question, were the following:

- Using continuous and multi-dimensional domains to improve the environment's richness.
- Continuously learning and adapting to changes in the environment, including in the reward function, i.e., the structure that encodes and captures the ethical considerations that agents should learn to exhibit.
- Learning in a multi-agent setting, by taking into account the difficulties posed by the presence of other agents.

The continuous and multi-dimensional aspect was solved by design, thanks to the SOMs and DSOMs that we use in our algorithms. They learn to handle the complex observations and actions domains, while advantageously offering a discrete representation that can be leveraged with the Q-Table, permitting a modular approach. This modular approach, and the use of Q-Tables, allow for example to compare different actions, which is not always possible in end-to-end Deep Neural Networks. This point in particular will be leveraged in our 3rd contribution, through multi-objective learning, in Chapter 6.

The continuous adaptation was also handled by our design choices, notably by disabling traditional convergence mechanisms. The use of (D)SOMs also help, as the representation may shift over time by moving the neurons. Additionally, our experiments highlight the ability of our algorithms to adapt, especially when compared to other algorithms, through the specific *adaptability1* and *adaptability2* functions.

Finally, challenges raised by the multi-agent aspect were partially answered by the use of Difference Rewards to create the reward functions. On the other hand, the agents themselves have no specific mechanism that help them learn a behaviour while taking account of the other agents in the shared environment, e.g., contrary to Centralized Training algorithms such as MADDPG. Nevertheless, our algorithms managed to perform better than MADDPG on the proposed scenarii and reward functions, which means that this limitation is not crippling.

In addition to the multi-agent setting, our algorithms still suffer from a few limitations, of which we distinguish 2 categories: limitations that were set aside in order to propose a working 1st step, and which will be answered incrementally in the next contributions; and longer-terms limitations that are still not taken care of at the end of this thesis. We will first briefly present the former, before detailing the latter.

From the objectives we set in this manuscript's introduction, and the important points identified in the State of the Art, the following were set aside temporarily:

- Interpretable rewards and reward functions constructed from domain expert knowledge: The reward functions we proposed have the traditional form of mathematical formulas; the resulting rewards are difficult to interpret, as the reasoning and rationale are not present in the formula. In this contribution, proposed reward functions were kept simple as a validation tool to evaluate the algorithms' ability to learn. We tackle this point in the next contribution, in Chapter 5, in which we propose to construct reward functions through symbolic-based judgments.

- Multiple moral values: Some of our reward functions hinted towards the combination of multiple moral values, such as the *multi-objective-sum*, *multi-objective-prod*, *adaptability1*, and *adaptability2*. However, the implementations of moral values were rather simple and did not include many subtleties, e.g., for the equity, we might accept an agent to consume more than the average, if the agent previously consumed less, and is thus "entitled" to a compensation. We also improve this aspect in the next contribution, by providing multiple judging agents that each focus on a different moral value.

- Multiple moral values / objectives: Moreover, these moral values, which can be seen as different objectives, are simply aggregated in the reward function. On the agents' side, the reward is a simple scalar, and nothing allows to distinguish the different moral values, nor to recognize dilemmas situations, where multiple values are in conflict and cannot be satisfied at the same time. This is addressed by our 3rd contribution, in Chapter 6, in which we extend the Q-(D)SOM algorithms to a multi-objective setting.

As for the longer-terms limitations and perspectives:

- As we already mentioned, the multi-agent aspect could be improved, for example by adding communication mechanisms between agents. Indeed, by being able to communicate, agents could coordinate their actions so that the joint-action could be even better. Let us assume that an agent, which approximately learned the environment dynamics, believes that there is not much consumption at 3AM, and chooses the strategy of replenishing its battery at this moment, so as to have a minimal impact on the grid. Another agent may, at some point, face an urgent situation that requires it to consume exceptionally at 3AM this day. Without coordination, the 2 agents will both consume an import amount of energy at the same time, thus impacting the grid and potentially over-consuming. On the other hand, if the agents could communicate, the second one may inform other agents of

its urgency. The first one would perhaps choose to consume only at 4AM, or they would both negotiate an amount of energy to share, in the end proposing a better joint-action than the uncoordinated sum of their individual actions. However, such communication should be carefully designed in a privacy-respectful manner.

In order to make the reward functions more understandable, in particular by non-AI expert, such as lay users, domain experts, or regulators, to be able to capture expert knowledge, and to allow easier modification, e.g., when the behaviour is not what we expected, we propose to replace the numerical reward functions by separate, symbolic, judging agents. We detail this in the next chapter.

5 Designing a reward function through symbolic judgments

In this chapter, we present the second contribution, which is the judging part from the conceptual architecture in Figure 3.2. First, Section 5.1 starts with an overview of this contribution, makes the link with the previous contribution, and motivates it with respect to the research question. We present two different models: the first, using logic rules, is described in Section 5.2, whereas the second, using argumentation, is described in Section 5.3. Section 5.4 details the experiments' conditions, which are based on the previous chapter, and results are reported in Section 5.5. Finally, we discuss the two models' advantages and limitations and compare their differences, in Section 5.6.

5.1 Overview

In the previous chapter, we defined and experimented the learning agent component of our architecture, to learn behaviours with ethical considerations. The logical next step is thus to guide the learning agents through a reward signal, which indicates the degree of their actions' correctness, i.e., alignment with moral values. This corresponds to the judgment of behaviours in the global architecture presented in Section 3.3.

We recall that we do not know beforehand which action should an agent take, otherwise we would not need learning behaviours. This lack of knowledge stems from the environment's complexity, and the question of long-term consequences. Would it be acceptable to take an action almost perfect, from a moral point of view, which leads to situations where it is difficult or infeasible to take further morally good actions? Or would it be better to take a morally good action, although not the best one, which leads to situations where the agent is able to take more good actions?

These questions are difficult to answer, without means to compute the weight of consequences, for each action in each situation. If the horizon of events is infinite, this is even computationally intractable.

However, we assume that we are able to judge whether a proposed action is a good one, based on a set of expected moral values. The more an action supports the moral values, the higher the reward. On the contrary, if an action defeats a moral value, its associated reward diminishes.

The task thus becomes: how to judge proposed actions and send an appropriate reward signal to agents, such that they are able to learn to undertake morally good actions? This is in line with our second research question: How to guide the learning of agents through the agentification of reward functions, based on several moral values to capture the diversity of stakes?

Traditionally, most reinforcement learning algorithms use some sort of mathematical functions to compute the rewards, as we did in the previous chapter. This seems rather intuitive, since the learning algorithm expects a real number, and thus reward functions usually output real numbers. However, such mathematical functions also entail disadvantages.

The first point is their difficulty to be understandable, especially to non-AI experts, such as external regulators, domain experts, lay users, ... We argue that understanding the reward function, or the individual rewards that it produces, may help us understand the resulting agent's behaviour. Indeed, the reward function serves as an incentive, it measures the distance with the behaviour that we expect from the agent, and thus intuitively describes this expected behaviour.

A second point is that some processes are easier to describe using symbolic reasoning rather than mathematical formulas. As we have seen in the state of the art, many works in Machine Ethics have proposed to use various forms of symbolic AI, such as argumentation, logic, event calculus, etc. Symbolic reasoning has already been used to judge the behaviour of other agents', and we propose to extend this judgment to the computation of rewards.

Symbolic formalizations allow to explicitly state the moral values, their associated rules, and the conflicts between rules. In this chapter, we propose to apply two different symbolic methods to design judgment-based reward functions. First, we partially use the Ethicaa platform (Cointe et al., 2016) to create judging agents based on logic reasoning,

beliefs, and Prolog-like rules. The second alternative uses argumentation graphs, with arguments and attack relations between them, to compute the judgment.

Figure 5.1 shows the abstract architecture of this idea, without taking into account the details of either logic-based or argumentation implementations. On the right-side of the figure, learning agents still receive observations from the environment, and they output actions to the environment. However, we can see that the "Compute reward" function, which was previously within the environment, is now "agentified", by introducing *judging agents*, on the left side of the figure. To perform this computation, judging agents receive the observations from the environment, and actions from learning agents (through the environment).

We want our learning agents to receive rewards based on multiple moral values, as highlighted by our objective O1.1. Each judging agent is attributed to a unique moral value, and rely on this explicitly defined value and its associated moral rules to determine the reward that should be sent to each learning agent, as part of their judgment process. It ensues that multiple rewards are produced for each learning agent. However, the learning algorithm expects a single, scalar reward. Thus, rewards from the different judging agents are aggregated before they are sent to learning agents. The two propositions, logic-

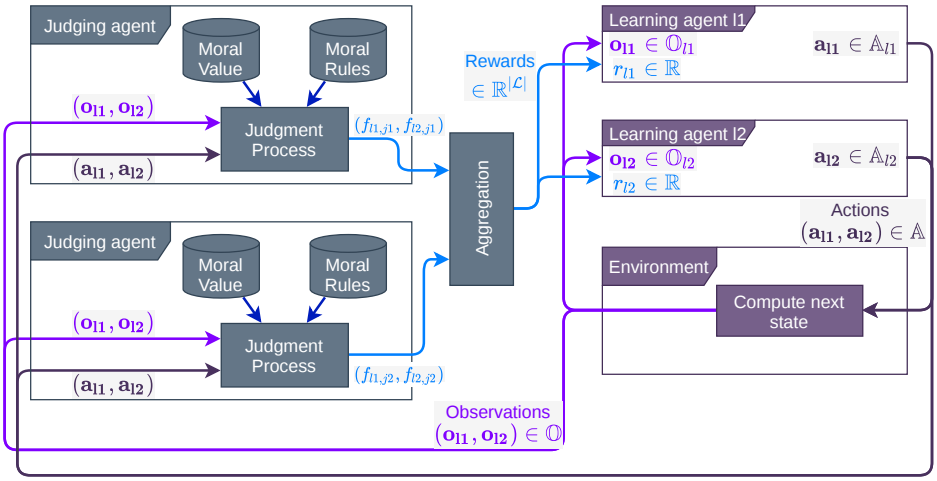


Fig. 5.1.: Architecture of the symbolic judgment. The environment and learning agents are the same as in the previous chapter, but the rewards' computation is moved from the environment to newly introduced judging agents, which rely on explicitly defined moral values and rules.

based and argumentation-based judging agents, are detailed below, as well as several experiments that demonstrate learning agents are able to correctly learn from such

symbolic-based rewards. We then discuss the differences between these 2 propositions, advantages and drawbacks with respect to the numeric-based reward functions, and remaining limitations and perspectives.

Remark. These two implementations outcome from collaboration with interns. Jérémy Duval worked on the implementation of the first method, based on Ethicaa agents (Duval, 2020 ; Chaput, Duval, Boissier, Guillermin, & Hassas, 2021). Benoît Alcaraz worked on the second method, judging through argumentation (Alcaraz, 2021). Christopher Leturc, who was at the time an associate lecturer at École des Mines St-Étienne (EMSE), brought his expertise on argumentation for this second internship.

5.2 Designing a reward function through logic rules

In this section, we first explain the motivations for combining a symbolic judgment with reinforcement learning agents, and for using logic-based agents to do so. To perform the symbolic judgment, we then introduce new, specific judging agents that are based on Ethicaa agents. Finally, the proposed model for the computation of rewards, which can be integrated with a reinforcement learning algorithm, is presented.

5.2.1 Motivations

As mentioned previously, this contribution focuses on leveraging simplified Ethicaa agents (Cointe et al., 2016) to judge the learning agents' actions, and determine an appropriate reward. There are multiple advantages and reasons behind this idea.

- The combination of symbolic (top-down) judgments and neural (bottom-up) learning constitutes a hybrid approach, which cumulates advantages of both.
  - Neural learning has the ability to generalize over unexpected situations.
  - Symbolic reasoning offers a better intelligibility of the expected behaviour, and the possibility to integrate prior knowledge from domain experts.
- Agentifying the reward function, by introducing judging agents, allows the judging and learning agents to evolve independently, and paves the way to co-construction.

  - Judging agents' rules can be updated by human designers, and learning agents must adapt their behaviour to comply with the judgments resulting from the new rule set. This opens to a perspective of human-centered AI with a human-in-the-loop schema.

- As mentioned in our general framework, intelligibility is important, especially to confirm whether the expected behaviour is aligned with our desired moral values.
  - The judgment process is implemented on explicit moral values and rules, expressed in a symbolic form, which improves the intelligibility.
- We obtain a richer feedback by combining the judgment of multiple judging agents, each corresponding to a single moral value.
  - This facilitates the implementation of the judgment process on one single moral value, and makes it more intelligible.
  - It also offers the possibility of more complex interactions between different judging agents each in charge of a single moral value with a dedicated rule set, such as negotiation processes.
  - Finally, it offers a way to update rules by adding, removing, or replacing judging agents.

We begin by introducing our judging agents, which are based on Ethicaa agents, how they work, and how they produce rewards.

5.2.2 Judging agents

To perform symbolic judgments with respect to given moral values and rules, we leverage the existing Ethicca agents (Cointe et al., 2016) that we have already mentioned in the State of the Art. These agents use the Beliefs - Desires - Intentions (BDI) architecture, and have an ethical judgment process to determine which actions are acceptable in a given situation.

An Ethicaa agent may perform a judgment on itself to determine which action it should take, or on another agent to determine whether it agrees with the other agent's behaviour. We propose to adapt this judgment of others to compute the reward functions for learning agents. The original Ethicaa agent includes several processes, such as the awareness and evaluation processes, to obtain beliefs about the situation, and determine the feasibility on an action. We ignore such processes, as we judge actions already taken, and focus on

the "goodness process", retaining only the components that we need for our approach. Figure 5.2 represents such a simplified agent, with an explicit base of moral rules (MR), and moral values in the form of "value support" rules (VS).

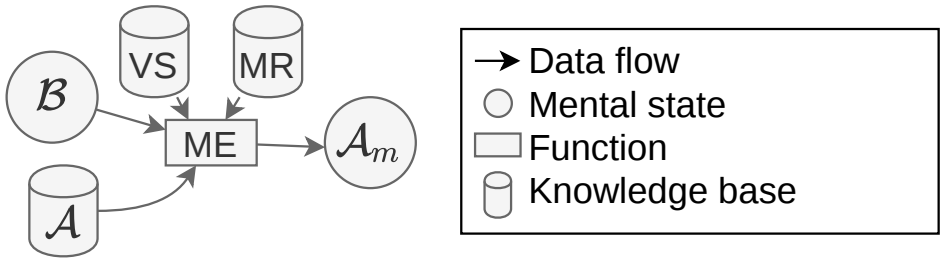


Fig. 5.2.: The Goodness Process of Ethicaa agents, adapted from Cointe et al. (2016).

Our proposed simplified agents use their beliefs about the current situation ($\mathcal{B}$), the moral values (VS), the moral rules (MR), and the knowledge about actions (A) to produce a Moral Evaluation (ME) of the actions. The Moral Evaluation returns a symbol, either $moral$, $immoral$ or $neutral$, from which we will compute the reward. We describe how we leverage this for our proposed model in the next section.

5.2.3 The proposed model: LAJIMA

We now describe the Logic-based Agents for JudgIng Morally-embedded Actions (LAJIMA) model, which is represented in Figure 5.3. Each judging agent contains an explicit moral value (VS) and associated moral rules (MR). They receive observations ($\mathbf{o_l}$) that are transformed into beliefs ($\mathcal{B}$), and actions ($\mathbf{a_l}$) that are similarly transformed into symbols ($\mathcal{A}$). Their judgment process relies on a moral evaluation function (ME) for each of the actions component $a_{l,1}, a_{l,2}, \cdots, a_{l,d}$. The moral evaluation function leverages the moral value (VS) and moral rules (MR) to do so. Symbolic moral evaluations compose the judgment of each judging agent j, which is later transformed into a numeric feedback through the F function. Per-judge feedbacks are finally aggregated to form the reward r_l sent to the learning agent l. This contribution focuses on the reward function, and, for all things necessary, we assume the same DecPOMDP framework as described in the previous chapter. More specifically, the reward functions we will construct through judging agents work with any RL algorithm, under two assumptions:

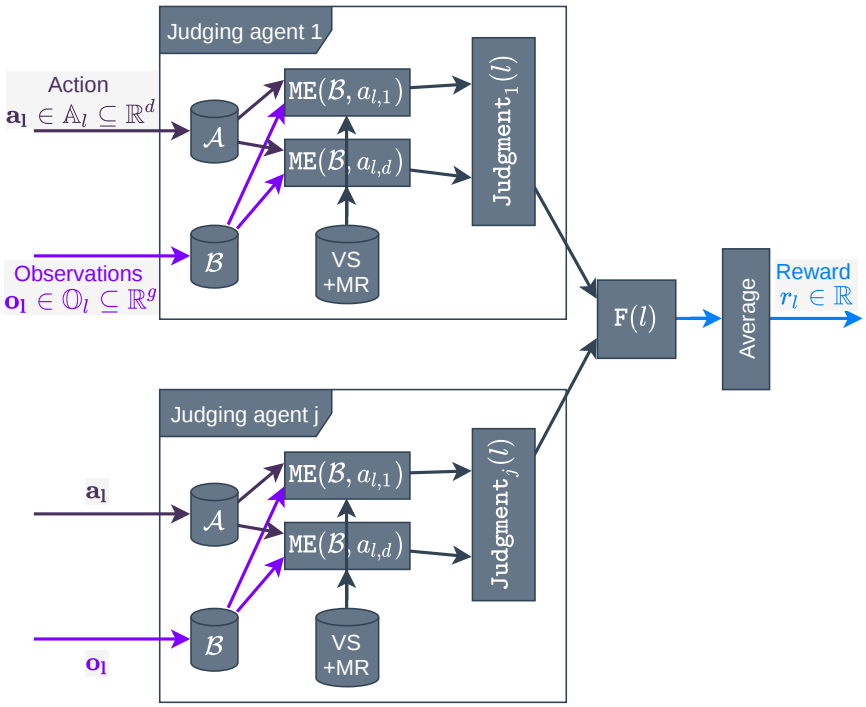


Fig. 5.3.: The judgment process of logic-based judging agents, which produces a reward as a scalar value for a learning agent l. This process is duplicated for each learning agent. The symbolic judgments are then transformed as numbers through the feedback function $\mathtt{F}$, and finally averaged to form a scalar reward r_l.

- Observations and actions are multi-dimensional and continuous.
- RL agents expect a scalar reward signal.

These assumptions are sufficiently generic to support a large part of existing RL algorithms, in addition to our own Q-SOM and Q-DSOM algorithms. They mean that judging agents will have to handle real vectors as inputs, and produce a real scalar as output.

At each step $t+1$ of the simulation, after learning agents took an action at step t, observations and executed actions are sent to judging agents. We recall, from the DecPOMDP framework 4.1, that an observation $\mathbf{o_{l,t}}$ is a vector $\in \mathbb{O}_l \subseteq \mathbb{R}^g$, and an action $\mathbf{a_{l,t}}$ is a vector $\in \mathbb{A}_l \subseteq \mathbb{R}^d$. In this case, observations received by judging agents are the same as observations received by learning agents, yet they do not know the full state of the environment, nor the exact process that learning agents used. In Ethicaa's vocabulary, this is deemed to be a *partially informed ethical judgment*, since judging agents have some but not all information about the judged agents.

From these observations, judging agents generate beliefs ($\mathcal{B}$) about the current situation, simply by creating a belief for each component of the observation vector $\mathbf{o_l}$, with the same name as the component, and a parameter that corresponds to the component's value. This produces symbols that can be handled by the judging agents' moral evaluation mechanism, from the numeric vectors.

Example 5.1. Let us imagine a learning agent l that receives an observation vector $\mathbf{o_l} = [0.27, 0.23, 0.35, 0.29, 0.51, 0.78, 0.47, 0.64, 0.95, 0.65]$. A judging agent will thus generate the following beliefs when judging l: `storage(0.27)`, `comfort(0.23)`, `payoff(0.35)`, `hour(0.29)`, etc.

Similarly, judging agents must judge actions based on symbols, whereas enacted actions by learning agents are vectors. Actions imply an additional difficulty, compared to observations: a single action vector represents in fact several "sub-actions" that happen at the same time, on different dimensions, e.g., consuming energy from the grid, and buying energy from the national network. A judging agent may judge one of the dimensions as supporting its associated moral value, whereas another dimension defeats the same moral value. Since they must return a scalar reward from their judgment, what should the judging agent return in this case?

To simplify this, we propose to decompose the enacted action into a set of action symbols $\{\forall i \in [[1, d]] : \mathbf{a}_{l,i}\}$, where l is the learning agent, $\mathbf{a_l}$ is the enacted action by l, i.e., a

vector of d components, and $\mathbf{a}_{l,i}$ is the i-th component, i.e., a real value. Each "action symbol" will be first judged individually, before being aggregated by the judging agent. As for the observations, a belief is generated with the component name and the component value as a parameter, for each component. However, contrary to the observations, the actions are scaled accordingly to the learning agent's profile.

Example 5.2. Let us imagine a learning agent with a "Household" building profile, which has an action range of $2,500$Wh. The agent chooses an action vector $\mathbf{a_l} = [0.78, 0.80, 0.53, 0.36, 0.52, 0.95]$. As a result, the action truly enacted by the agent is the action vector, scaled by the action range, i.e., $[0.78 \times 2500, 0.80 \times 2500, \cdots]$. The judging agent will therefore generate the following beliefs: `consumegrid(1950)`, `store(2000)`, etc.

Judging agents use the set of generated scalar values, known moral values and associated moral rules to determine if each component of the action, or "action symbol" supports or defeats their moral value. Moral rules are logical predicates expressing the support (or defeat) of an action component to a moral value, based on observations. To do so, we define the Moral Evaluation mechanism of judging agent j as a function $\texttt{ME}_j : \mathbb{B} \times \mathbb{R} \rightarrow \mathcal{V}$, where we define $\mathbb{B}$ as the space of possible beliefs about the situation, and $\mathcal{V}$ as the set of possible valuations $\mathcal{V} = \{moral, immoral, neutral\}$.

For example, the following lines, which are in a Prolog-like language, indicate that a learning agent's action supports the sub-value of "promoting grid autonomy" if the quantity `X` associated to the `buy_energy` component is not more than 100W. This sub-value is related to the *environmental sustainability* value, and each action component is judged individually and receives a moral evaluation, with respect to this moral value. If the action component defeats the moral value, or one of its sub-values, the evaluation is said to be $immoral$. Otherwise, if the action component supports the value, its evaluation is said to be $moral$. A default $neutral$ evaluation is assigned when the action component neither supports nor defeats the value.

```
valueSupport(buy_energy(X), "promote_grid_autonomy") :- X <= 100.
valueDefeat(buy_energy(X), "promote_grid_autonomy") :- X > 100.

subvalue("promote_grid_autonomy","env_sustain").

moral_eval(_,X,V1,immoral):- valueDefeat(X,V1) & subvalue(V1,"env_sustain").
```

```
moral_eval(_,X,V1,moral):- valueSupport(X,V1) & subvalue(V1,"env_sustain").
moral_eval(_,_,_,neutral).
```

Remark. Note that, in the previous example, the `subvalue` fact highlights the possibility of creating a value hierarchy, represented by the "VS" knowledge base of value supports. In our case, this hierarchy is not necessary and can be simplified: we use it mainly to make the code more understandable, by aptly naming the sub-values, e.g., here `promote_grid_autonomy` for actions that avoid buying too much energy from the national grid.

The judgment of a learning agent l by a judging agent j is thus the Moral Evaluation on every action symbol. Mathematically, we define $\texttt{Judgment}_j(l) = \{\forall i \in [[1, d]] : \texttt{ME}_j(\texttt{beliefs}(\mathbf{o_l}), a_{l,i})\}$, where $\mathbf{o_l}$ is the observation vector received by l, and $\mathbf{a_l}$ is the action vector chosen by l.

A judging agent represents a single moral value; however, we want to judge a learning agent's behaviour based on various moral values. A learning agent l is judged by all judging agents, which results in a list of lists of valuations.

Example 5.3. Let $d = 3$ the size of an action vector, l be a learning agent, and j_1, j_2 two judging agents. To simplify the notation, we have $\mathbf{B} = \texttt{beliefs}(\mathbf{o_l})$. Assume the judgments received by l is

$$\begin{aligned}
&\left\{\texttt{Judgment}_{j_1}(l), \texttt{Judgment}_{j_2}(l)\right\} \\
&= \{\{\texttt{ME}_{j_1}(\mathbf{B}, a_{l,1}), \texttt{ME}_{j_1}(\mathbf{B}, a_{l,2}), \texttt{ME}_{j_1}(\mathbf{B}, a_{l,3})\}, \{\texttt{ME}_{j_2}(\mathbf{B}, a_{l,1}), \texttt{ME}_{j_2}(\mathbf{B}, a_{l,2}), \texttt{ME}_{j_2}(\mathbf{B}, a_{l,3})\}\} \\
&= \{\{moral, neutral, neutral\}, \{immoral, immoral, moral\}\}
\end{aligned}$$

In this example, we can see the first judging agent j_1 deemed the first dimension of the action $a_{l,1}$ to be moral, i.e., consistent with its moral value and rules, whereas the second judging agent j_2 deemed this same dimension to be immoral, i.e., inconsistent with its moral value and rules. The second dimension of the action $a_{l,2}$ was deemed neutral by the first judge, and immoral by the second judge; the third and final dimension $a_{l,3}$ was deemed neutral by the first judge, and moral by the second.

The reward function $\texttt{R}_l : \mathbb{S} \times \mathbb{A}_l \times \mathbb{S} \to \mathbb{R}$ must return a single, real number r_l. However, we have a list of lists of valuations. Additionally, as we saw in the previous example, we may have conflicts between judgments, both within a single judge, and between judges. A

judge may determine that some components of the action are immoral and others moral; multiple judges may determine that the same component of an action is both immoral and moral, according to their own different moral values.

We propose the following method to transform the set of valuations, although many other methods are possible. Such methods, how they differ with ours, and what these differences imply, are discussed in Section 5.6.

The Feedback function $\mathtt{F} : \left(\mathcal{V}^d\right)^{|\mathcal{J}|} \to \mathbb{R}^{|\mathcal{J}|}$ transforms the valuations into a list of numbers. The judgment of each judging agent, i.e., a list of valuations, is transformed into a single number, by counting the number of $moral$, and dividing it by the sum of the number of $moral$ and $immoral$ valuations. This means that the more $moral$ valuations an action receives, the more it will tend towards 1. On the contrary, the more $immoral$ valuations an action receives, the more it will tend towards 0. If an action only received $neutral$ valuations, we consider it was neither good nor bad, and we set the number to 0.5 as a special case. We can note that this special case also corresponds to a situation where an action received as much $moral$ valuations as $immoral$.

We thus have a single number for each judging agent, which resolves conflicts within judges. Then, to solve the conflicts between judges, and produce a single reward for all judges, we set the reward r_l to be the average of these judgment numbers produced by each judging agent.

Example 5.4. We reuse the same judgments from example 5.3: $\mathtt{Judgment}_{j_1}(l) = \{moral, neutral, neutral\}$, and $\mathtt{Judgment}_{j_2}(l) = \{immoral, immoral, moral\}$. The first judgment contains 1 moral valuation, and 0 immoral; thus, the resulting number is $\frac{1}{1+0} = 1$. On the other hand, the second judgment contains 1 moral valuation but 2 immoral valuations; thus, its resulting number is $\frac{1}{1+2} = \frac{1}{3}$. In other words, $\mathbf{f_l} = \mathtt{F}\left(\{\{moral, neutral, neutral\}, \{immoral, immoral, moral\}\}\right) = \left\{1, \frac{1}{3}\right\}$. Finally, the reward produced by the aggregation of these two judgments is simply the average of these numbers, i.e., $r_l = \mathtt{average}(\mathbf{f_l}) = \mathtt{average}(\left\{1, \frac{1}{3}\right\}) = \frac{2}{3}$.

Algorithm 5.1 summarizes the judgment process that we defined in this section. The foreach loop that begins on line 3 performs the judgment of all learning agents individually and sequentially. Each judging agent computes its beliefs over the observations $\mathbf{o_l}$ received by a learning agent l (line 5), and uses them to judge each component of the action parameters (lines 7-9). The judgment for each dimension of the action is retained in a

Algorithm 5.1: Logic-based judgment process

Function `judgment()`:
  Input:
    Observations $\mathbf{o_l}$
    Chosen action parameters $\mathbf{a_l}$
  Output: Rewards $\mathbf{r}$
  $\mathbf{r} \leftarrow \emptyset$
  foreach *learning agent* $l \in \mathcal{L}$ **do**
    /* Perform symbolic judgment of all judging agents */
    foreach *judging agent* $j \in \mathcal{J}$ **do**
      $\mathbf{B} \leftarrow \texttt{beliefs}(\mathbf{o_l})$
      $\texttt{Judgment}_j(l) \leftarrow \emptyset$
      foreach *dimension* $i \in [[1, d]]$ **do**
        $\texttt{Judgment}_j(l)_i \leftarrow \texttt{ME}(\mathbf{B}, a_{l,i})$
      end
    end
    /* Transform symbolic judgments into a single scalar reward */
    $\mathbf{f_l} \leftarrow \emptyset$
    foreach *judging agent* $j \in \mathcal{J}$ **do**
      if *at least one moral or immoral valuation in* $\texttt{Judgment}_j(l)$ **then**
        $moral \leftarrow \left|moral \in \texttt{Judgment}_j(l)\right|$
        $immoral \leftarrow \left|immoral \in \texttt{Judgment}_j(l)\right|$
        $f_{l,j} \leftarrow \frac{moral}{moral+immoral}$
      end
      else
        $f_{l,j} \leftarrow \frac{1}{2}$
      end
    end
    $r_l \leftarrow \texttt{average}\left(\{\forall f_{l,j} \in \mathbf{f_l}\}\right)$
  end
  return $\mathbf{r}$
end

vector, so that the judge can later count the number of `moral` and `immoral` valuations (lines 14-16). Remember that, if the learning agent received only `neutral` valuations by a given judge, a default reward of $\frac{1}{2}$ is attributed (lines 13 and 19). Finally, the reward ultimately received by a learning agent is the average of the per-judge rewards (line 22).

5.3 Designing a reward function through argumentation

We now propose a second method to design the reward function as symbolic judgments, through argumentation. The differences between logic rules and argumentation, their respective advantages and drawbacks, are discussed in Section 5.6.

5.3.1 Motivations

- Argumentation graphs offer a rich structure, by relying on the notion of arguments and attack relations. This allows explicit conflicts, represented by attacks between arguments, and offers a finer control over the targeted behaviour. In particular, it is easier to reprehend an undesired behaviour, by attacking arguments when specific conditions are met. For example, let us imagine that an agent learned to hoard energy at a time step t in order to give energy at a time step $t+1$, therefore receiving praises for its "generous" behaviour at $t+1$. If the hoarding behaviour at t is problematic, we may add an argument to the graph, which is activated when the agent stored a lot of energy at the previous step, and which attacks the pro-argument of having given energy. Thus, this pro-argument will be killed in such situations, which will prevent the agent from receiving a high reward: it will have to stop its hoarding behaviour to get better rewards.
- Note that it is easier for designers to structure the moral rules through the graph, as mentioned in the previous point. The attack relationship makes explicit the impact of an argument on another.
- Additionally, argumentation using a graph structure makes it feasible to visualize the judgment process, by plotting the arguments as nodes and attacks as edges between nodes. Non-developers, such as external regulators, lawyers, ethicists, or even lay users, can look at the graph and get a glimpse of the judgment function.

- Arguments themselves, and whether they were activated at a given time step t, can be leveraged to understand the learnt and exhibited behaviour. For example, if the argument "has given energy" was activated, we can understand why the agent received a high reward. Or, on the contrary, if both arguments "has given energy" and "has previously hoarded energy" were activated, we can understand why the agent did not receive a high reward, although its actions at this specific time step seemed praiseworthy. They can be further used in explanations techniques: whereas we do not consider our method an explanation technique per se, these arguments represent elements of explanations that can be leveraged in explanation methods to explore and understand the expected behaviour, the incentives that led to each reward and ultimately that led to learn a given behaviour, and compare the exhibited behaviour to the expected one. In most AI techniques, these elements simply do not exist.

- Finally, the previous points may help us diminish the *reward hacking* risk, i.e., the possibility that an agent learns to optimize the reward function through an undesired (and unexpected) behaviour. A classic example is an agent that is rewarded based on the distance towards a goal, and which may find that it is more profitable to indefinitely circle as close as possible to the goal, in order to obtain an infinitely positive reward. Such reward hacking can, first, be detected by looking at the sequence of activated arguments. Without using argumentation, we might detect that something is off, by seeing a sequence of low then high rewards, e.g., $0.2, 0.8, 0.2, 0.8$, but we might not understand *what* or *why*. We again take the example of the "hoarding then giving" behaviour presented earlier. Looking at arguments' activations might reveal that, when the reward is low, the argument "agent stored too much energy" was activated; when the reward is high, the argument "agent gave a lot of energy" was activated. We can thus understand what the problem is. Secondly, the reward hacking can be fixed, by adding new arguments that prevent such hacking, as we have already mentioned.

5.3.2 Argumentation

Before introducing our AJAR model in the next section, based on argumentation decision frameworks, we briefly cover some necessary knowledge about argumentation.

Abstract argumentation frameworks, first introduced by Dung (1995), allow us to express knowledge as a set of arguments, such as "There are clouds today", "The weather is nice". Arguments are arranged as nodes of a directed graph, where the links, or edges, between nodes represent binary attack relations. For example, the argument "There are clouds today" attacks "The weather is nice". This means that, if we consider the argument "There are clouds today" as true, or *alive*, in the current situation, it will be difficult to accept the argument "The weather is nice" as well. An example is given in Figure 5.4.

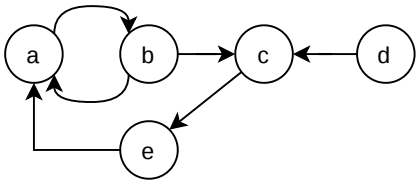


Fig. 5.4.: Simple example of an argumentation graph that contains 5 arguments (a, b, c, d, and e), represented by nodes, and 6 attack relations, represented by edges. Argument a attacks b, b attacks a and c, c attacks e, d attacks c, and e attacks a.

Multiple argumentation frameworks exist, e.g., attacks can be weighted (Coste-Marquis, Konieczny, Marquis, & Ouali, 2012), bipolar frameworks bring the notion of *supporting* an argument in addition of *attacking* (Amgoud, Cayrol, Lagasquie-Schiex, & Livet, 2008), arguments and attacks can have a probability (Li, Oren, & Norman, 2011), etc. In this contribution, we want to leverage the argumentation to produce a judgment of an agent's behaviour. This can be seen as some sort of decision, where the possible outcomes are "The agent's action was *moral* in regard to this specific moral value", or "The agent's action was *immoral* in regard to this specific moral value."

We propose to use a simplified version of the *Argumentation Framework for Decision-Making* (AFDM) of Amgoud & Prade (2009). Their model considers that multiple decisions can be possible, whereas in our case, we only need to determine the degree to which an action can be considered *moral*, or *immoral*. We thus restrict our arguments to be *pros*, *cons*, or *neutral* arguments, and drop the set of decisions. This simplified version, instead of targeting decision-making, focuses on judging decisions, and we thus name it *Argumentation Framework for Judging a Decision* (AFJD).

Definition 5.1 (AFJD)**.** An Argumentation Framework for Judging a Decision (AFJD) is defined as a tuple:

$AF = \langle \mathcal{A}rgs, \mathcal{A}tt, \mathcal{F}_p, \mathcal{F}_c \rangle$, where:

- $\mathcal{A}rgs$ is a non-empty set of arguments. An argument is represented by a name, and has an aliveness condition function. This function is defined as $\texttt{alive} : \mathbb{S} \to \mathbb{B}$, where $\mathbb{S}$ is the space of world states, and $\mathbb{B} = \{0, 1\}$ is the set of boolean values. It determines whether the argument can be considered "true" in the current state of the world, or "false", in which case the argument is ignored.
- $\mathcal{A}tt$ is a binary relation named the attack relation, defined on pairs of arguments, such that $A\ \mathcal{A}tt\ B$ means that argument A attacks argument B.
- $\mathcal{F}_p \in 2^{\mathcal{A}rgs}$ (for *pros*) is the set of pro-arguments, which indicate that the currently judged action was a moral one.
- $\mathcal{F}_c \in 2^{\mathcal{A}rgs}$ (for *cons*) is the set of con-arguments, which indicate that the currently judged action was an immoral one.

To simplify notations in the sequel, we will refer to the elements of an AFJD through subscripts. In other words, for a given $AF = \langle \mathcal{A}rgs, \mathcal{A}tt, \mathcal{F}_p, \mathcal{F}_c \rangle$, we will note: $AF_{[\mathcal{A}rgs]} = \mathcal{A}rgs$, $AF_{[\mathcal{A}tt]} = \mathcal{A}tt$, $AF_{[\mathcal{F}_p]} = \mathcal{F}_p$, $AF_{[\mathcal{F}_c]} = \mathcal{F}_c$.

We will have to filter out arguments that cannot be considered as *alive* in the current situation, before we can make a decision. For example, if in the current situation, there are no clouds, then we disable the "There are clouds today". Disabling an argument removes the node from the graph, as well as its attacks on other arguments of the graph. Formally, we call this filtered graph a "sub-AFJD", i.e., an AFJD which is some subset of another AFJD: its arguments are a subset of the other AFJD's arguments, and, by extension, its attack relation, *pros*, and *cons* are also subsets. We thus define the set of all possible sub-AFJD as:

$$\mathcal{P}(AF) := \{\Big\langle \mathcal{A}rgs', \mathcal{A}tt', \mathcal{F}'_p, \mathcal{F}'_c \Big\rangle : \begin{array}{l} \mathcal{A}rgs' \subseteq AF_{[\mathcal{A}rgs]}, \\ \mathcal{A}tt' \subseteq \mathcal{A}rgs'^2 \cap AF_{[\mathcal{A}tt]}, \\ \mathcal{F}'_p \subseteq \mathcal{A}rgs' \cap AF_{[\mathcal{F}_p]}, \\ \mathcal{F}'_c \subseteq \mathcal{A}rgs' \cap AF_{[\mathcal{F}_c]} \end{array} \}$$

As arguments may attack each other, and we want to compute the compliance of the learning agent's action, we need a way to determine whether we should take a specific argument into account.

Example 5.5 (Attacking and defending arguments)**.** Let us consider an argumentation graph composed of 3 arguments. The first one, arg_1, says "The agent did not consume

too much". A second one, arg_2 attacks arg_1 and says "The agent consumed more than the average of all agents". Finally, a third argument arg_3 attacks arg_2: "The agent had been in short supply for several time steps". As arg_2 attacks arg_1 and arg_3 attacks arg_2, we say that arg_3 *defends* arg_1. Which arguments should be taken into account to judge the agent's action? As arg_2 attacks arg_1, we might be tempted to ignore arg_1, however, arg_3 also attacks arg_2. Thus, if we ignore arg_2, can we keep arg_1 as well?

In argumentation theory, this problem is known as defining *acceptability*. An argument that is inacceptable should not be taken into account by the judge, contrary to an acceptable argument. Acceptability is often based on the notion of conflictness between arguments. The following definitions, *conflict-freeness* and *acceptability*, are paraphrased from Dung (1995, p. 6).

Definition 5.2 (Conflict-freeness)**.** Let $\mathcal{A}rgs$ be a set of arguments, and $\mathcal{A}tt$ be an attack relationship on $\mathcal{A}rgs$. A subset of arguments $\mathcal{SA}rgs \subseteq \mathcal{A}rgs$ is said to be *conflict-free*, with respect to $\mathcal{A}tt$, if and only if $\nexists (A, B) \in \mathcal{SA}rgs$ such that $A\ \mathcal{A}tt\ B$.

Definition 5.3 (Acceptability)**.** Let $\mathcal{A}rgs$ be a set of arguments, $\mathcal{A}tt$ be an attack relationship on $\mathcal{A}rgs$, and $\mathcal{SA}rgs$ be a subset of arguments, such that $\mathcal{SA}rgs \subseteq \mathcal{A}rgs$. For each argument $A \in \mathcal{A}rgs$, A is said to be *acceptable* with respect to $\mathcal{SA}rgs$, if and only if $\forall B \in \mathcal{A}rgs : B\ \mathcal{A}tt\ A, \exists C \in \mathcal{SA}rgs, C\ \mathcal{A}tt\ B$. In other words, an argument is acceptable with respect to a set, if all arguments that attack it are themselves attacked by at least an argument in the set. This represents some sort of "defense" between arguments: an argument A is defended by C, if B attacks A and C attacks B. The "defense" can be extended to the subset $\mathcal{SA}rgs$: if the arguments in $\mathcal{SA}rgs$ defend A by attacking all its attackers, then we say that $\mathcal{SA}rgs$ defends A.

We can note from these two definitions that acceptability depends on the considered set. It is possible to find a conflict-free set, in which all arguments are acceptable, but maybe another set exists, holding the same properties. In this case, which one should we choose? And more importantly, why?

To solve this question, argumentation scholars have proposed various definitions of admissible sets of arguments, called *extensions*. The simplest one may be the *admissible* extension, while others propose the *complete* extension, *stable* extension, or *preferred* extension (Dung, 1995). We define a few extensions that we will use in our proposed

algorithm; they are paraphrased from Caminada (2007, see Definition 3, p.2) to be slightly more explained.

Definition 5.4 (Extensions)**.** Let us consider $\mathcal{A}rgs$ a set of arguments within an Argumentation Framework, and $\mathcal{SA}rgs \subseteq \mathcal{A}rgs$ a subset of these arguments. The extensions are defined as follows:

- $\mathcal{SA}rgs$ is an *admissible* extension if and only if $\mathcal{SA}rgs$ is *conflict-free*, and all arguments $A \in \mathcal{SA}rgs$ are *acceptable*, with respect to $\mathcal{SA}rgs$.
- $\mathcal{SA}rgs$ is a *complete* extension if and only if $\mathcal{SA}rgs$ is admissible, and contains all acceptable arguments with respect to $\mathcal{SA}rgs$. In other words, $\forall A \in \mathcal{A}rgs$, if $\mathcal{SA}rgs$ defends A, then we must have $A \in \mathcal{SA}rgs$.
- $\mathcal{SA}rgs$ is a *grounded* extension if and only if $\mathcal{SA}rgs$ is a minimal complete extension, with respect to $\subsetneq$, i.e., $\nexists \mathcal{SA}rgs' \subseteq \mathcal{A}rgs$ such that $\mathcal{SA}rgs' \subsetneq \mathcal{SA}rgs$ and $\mathcal{SA}rgs'$ is a complete extension.

Although there is no clear consensus in the community as to which extension should be used, the uniqueness property of some extensions, such as the *grounded* or *ideal*, makes them an attractive choice (Caminada, 2007). Indeed, this property means that we can compute the extension, and be guaranteed of its unicity, without worrying about having to implement in the judging agent a choice mechanism between possible sets. The *grounded* extension also has the advantage of being computed through a very efficient algorithm in $\mathcal{O}\left(|\mathcal{A}rgs| + |\mathcal{A}tt|\right)$ time, where $|\mathcal{A}rgs|$ is the number of arguments in the whole graph, and $|\mathcal{A}tt|$ is the number of attacks between arguments in $\mathcal{A}rgs$ (Nofal, Atkinson, & Dunne, 2021). For these two reasons, we will consider the *grounded* extension in our proposed model.

5.3.3 The proposed model: AJAR

We now present the Argumentation-based Judging Agents for ethical Reinforcement learning (AJAR) framework. In this proposed model, similarly to the logic-based model, we introduce judging agents to compute the rewards. Each judging agent has a specific moral value, and embeds an AFJD relative to this moral value.

Definition 5.5 (Argumentation-based Judging Framework)**.** We define an Argumentation-based Judging Framework as a tuple $\langle \mathcal{J}, \{AF_j\}, \{\epsilon_j\}, \{\mathtt{J}_j\}, \mathtt{g}_{\mathtt{agr}} \rangle$, where:

- $\mathcal{J}$ is the set of judging agents.
- $\forall j \in \mathcal{J} \quad \epsilon_j : \mathcal{L} \times AF \times \mathbb{S} \to \mathcal{P}(AF_j)$ is a function that filters the AFJD to return the sub-AFJD that judging agent j uses to judge the learning agent i.
- $\forall j \in \mathcal{J} \quad \mathtt{J}_j : \mathcal{P}(AF_j) \to \mathbb{R}$ is the judgment function, which returns the reward from the sub-AFJD.
- $\mathtt{g_{agr}} : \mathbb{R}^{|\mathcal{J}|} \to \mathbb{R}$ is the aggregation function for rewards.

To simplify, we consider that the ϵ_j is the same function for each judging agent j; however, we denote it ϵ_j to emphasize that this function takes the AFJD associated to j as input, i.e., AF_j, and returns a sub-AFJD $\in \mathcal{P}(AF_j)$. The same reasoning is applied to the judgment function $\mathtt{J}_j$.

The ϵ_j function is used to filter the AFJD a judging agent relies on, according to the current state of the world. We recall that each argument of the AFJD has an *alive* condition function, which determines whether the argument is alive or not in this state. This means that designers create an AFJD with all possible arguments, and only the relevant arguments are retained during the judgment, through the ϵ function.

Example 5.6. For example, an AFJD may contain 3 arguments: "the agent consumed 10% more than the average", "the agent consumed 20% more than the average", and "the agent consumed 30% more than the average". At a given time step t, in state s_t, if the agent consumed 27% more than the average, the *alive* condition of the first two arguments will be true, whereas the last argument will be considered dead. Thus, when performing the judgment, the judging agent will only consider the first two, even for computing the *grounded* extension. Thanks to ϵ_j, the judgment acts as if the third argument was simply not part of the graph, and all its attacks are removed as well.

In the current definition, the ϵ_j function assumes that judging agents have full knowledge about the world's state s. This allows comparing the learning agent to other agents, e.g., for the average consumption, which we cannot access if we only have the learning agent's observations $\mathbf{o_l}$. This assumption can be lifted by changing the definition of ϵ_j to take $\mathbb{O}_l$ as domain; however, this would limit the available arguments, and thus the judgment as a whole.

The world's state s consists of data from the true environment's state and the agents' actions. This represents some sort of very limited history, with the world state containing both the action itself and its consequences. A better history could also be used, by

remembering and logging the actions taken by each agent: this would expand what the designers can judge within the argumentation graphs, e.g., behaviours over several time steps, such as first buying energy at t and then giving at t', with $t' > t$. Note that actions from all learning agents are included in the world's state, so that the currently judged agent can be compared to the others. This allows arguments such as "the learning agent consumed over 20% more than the average".

For the sake of simplicity, we pre-compute a few elements from these continuous data, such as the average comfort, the difference between the maximum and minimum comfort, the quantity of energy the currently judged learning agent consumed, etc. These elements are made available to the judging agents so that they can filter the graph through ϵ_j. This pre-computation makes the argumentation graphs easier to design, as arguments, and by extension graphs and judging agents, can directly rely on these elements rather than performing the computation themselves, and it avoids some overhead when 2 different graphs use the same element.

From the filtered sub-AFJD, which only contains the alive arguments according to the current situation, we then compute the *grounded* extension to keep only the acceptable arguments, i.e., we remove arguments that are attacked and not defended. This produces an even more filtered graph grd, which only contains arguments that are both alive and in the grounded extension.

From this graph grd, we want to produce a reward as a scalar number: this is the goal of the judgment function $\mathtt{J}_j$. There are multiple methods to do so, and we provide and compare a few in our experiments. Still, we give here an intuitive description to roughly understand what $\mathtt{J}_j$ is doing. We recall that, in an AFJD, an argument may belong to the pros set, $\mathcal{F}_p$, or the cons set, $\mathcal{F}_c$. An argument that is in the pros set supports the decision that the agent's behaviour was moral with respect to the moral value. $\mathtt{J}_j$ intuitively counts the number of arguments in $grd_{[\mathcal{F}_p]}$, and the number of arguments in $grd_{[\mathcal{F}_c]}$ and compares them. The more pros arguments, the more the reward will tend towards 1; conversely, the more cons arguments, the more the reward will tend towards 0.

The $\mathtt{J}_j$ function should however provide as much "gradient" as possible, i.e., an important graduation between 1 and 0, so that: 1) we can differentiate between various situations, and 2) we offer the agent an informative signal on its behaviour. If there is not enough gradient, we may have a case where the agent's behaviour improves, in the human's eye, but the reward does not increase, because the difference is too marginal. The agent would thus probably try another behaviour, to improve its reward, whereas its behaviour

was already improving. On the contrary, if there is enough gradient, the reward will effectively increase, and this will signal the learning algorithm that it should continue in this direction. For example, if we suppose the number of cons arguments stayed the same, but the number of pros arguments increased, the resulting reward should increase as well, to indicate that the behaviour improved.

The described AJAR judging process is formally presented in Algorithm 5.2 and graphically summarized in Figure 5.5. The figure shows the process for a single learning agent, which is then repeated for each learning agent, as the formal algorithm shows with the for loop on the set of learning agents $\mathcal{L}$. Similarly to the previous algorithm, the for loop

Algorithm 5.2: Argumentation-based judgment process

```
1  Function judgment():
       Input:
           Observations o_l
           Action parameters chosen by learning agent a_l
       Output: Rewards r
2      r ← ∅
3      foreach learning agent l ∈ 𝕃 do
           /* Perform symbolic judgment of all judging agents        */
4          F(l) ← ∅
           /* Compute the state                                      */
5          s ← ComputeState(o, a, l)
6          foreach judging agent j ∈ 𝕁 do
               /* Compute sub-AFJD                                   */
7              AF' ← ε_j(AF_j, l, s)
               /* Compute grounded extension                         */
8              grd ← grounded(AF')
               /* Compute reward through the F_p and F_c             */
9              F(l)_j ← J_j(grd)
10         end
           /* Agregate judgments into a single reward                */
11         r_l ← g_agr({∀f_{l,j} ∈ F(l)})
12     end
13     return r
14 end
```

beginning on line 3 performs the judgment individually and sequentially for each learning agent. We compute the world state from all observations of all learning agents (line 5): this world state may contain some pre-computed values to simplify the arguments' activation functions, such as "the learning agent consumed 27% more than the average". Judging agents use this world state to filter out arguments that are not enabled, through the ϵ_j function (line 7). This function returns the sub-AFJD that consists of all arguments

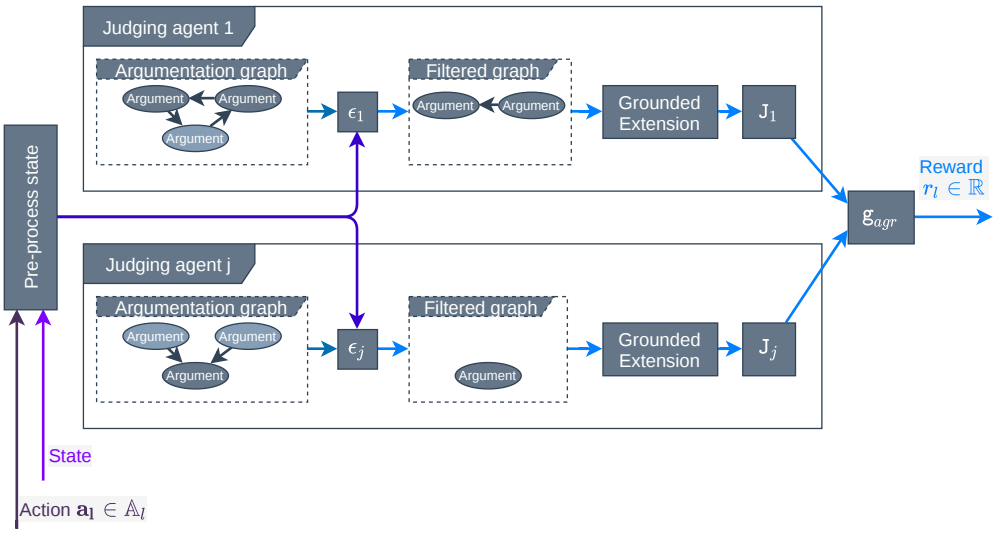


Fig. 5.5.: The judgment process of the AJAR framework, which leverages argumentation graphs to determine appropriate rewards. The argumentation graphs are first filtered based on the situation and the taken action to remove arguments that do not apply. Then, the reward is determined by comparing remaining pros and cons arguments.

whose activation function returns true for the current world state s; attacks of disabled arguments are also removed. In Figure 5.5, these disabled arguments are in a lighter shade of grey. From this sub-AFJD, judges compute the grounded extension (line 8), by removing arguments that are killed and not defended by other arguments. To do so, we used the (efficient) algorithm proposed by Nofal et al. (2021). The per-judge reward is computed by comparing the pros and cons arguments remaining in the grounded extension (line 9); we describe a few methods to do so in Section 5.4.2. Finally, the per-judge rewards are aggregated into a scalar one for each learning agent (line 10); again, we compare several approaches in Section 5.4.2.

Remark (Value-based Argumentation Frameworks). Usually, AI systems that leverage argumentation and deal with (moral) values focus on Value-based Argumentation Frameworks (VAF) (Bench-Capon, 2002). In these frameworks, compared to our AFJD, each argument is associated to a value, and the framework additionally contains a preference relation over the possible values. This allows comparing arguments w.r.t. their respective values, such that an argument can be deemed stronger than another because its associated value is preferred. When using such frameworks, all arguments belong to the same graph and can thus interact with each other. On the contrary, in our approach, we proposed

that different values are represented by different graphs, separated from each other. This means that an argument, relative to a given value, e.g., ecology, cannot attack an argument relative to another value, e.g., well-being. An advantage of this design choice is that we may add (resp. remove) moral values, i.e., argumentation graphs, without having to consider all possible interactions between the existing arguments and the new (resp. old) arguments. Yet, VAFs are well suited for decision-making, as they require the designer to consider such interactions when building the graph; we argue that using our AFJD is on the other hand suited for judgment-making.

5.4 Experiments

We have designed several experiments to evaluate our contribution, both on the logic-based and the argumentation-based agents. These experiments rely on the moral values we previously described in Section 3.4.5: security of supply, inclusiveness, environmental sustainability, and affordability. As for the previous experiments, we particularly emphasize the adaptation capability: in some scenarii, we incrementally enable or disable the judging agents. We recall that each judging agent is specific to a given moral value: in turn, this means that we add or remove moral values from the environment at different time steps. It might seem curious to remove moral values; we use this scenario as proof that, in combination with the ability to add moral values, we can effectively learn to follow evolutions of the ethical consensus within the society. Indeed, removing a previous moral value and adding a new one ultimately amounts to replace and update an existing moral value.

Remark. Note that, as the goal of this contribution is to propose an alternative way of specifying the reward function, i.e., through symbolic judgment rather than mathematical functions, the reward functions in these experiments are different from those used in the first contribution's experiments in Section 4.5. Thus, the hyperparameters are not optimized for these new reward functions: the agents could attain a sub-optimal behaviour. However, we have chosen to avoid searching for the new best hyperparameters, in order to save computing resources. We will also not compare the algorithms to baselines such as DDPG or MADDPG. Indeed, we present here a proof-of-concept work on the usability of symbolic judgments for reward functions, and not a "competitive" algorithm. The goal is not to evaluate whether the algorithm achieves the best score, but rather to evaluate whether: 1) the reward function can be learned by the algorithm, and 2)

the reward function implies an interesting behaviour, that is, a behaviour aligned with the moral values explicitly encoded in the function. These objectives do not require, per se, to use the best hyperparameters: if learning agents manage to learn an interesting behaviour by using sub-optimal parameters, it means that our proof-of-concept symbolic judgments can be used.

In both logic-based (LAJIMA) and argumentation-based (AJAR) models, we implement judging agents that represent the moral values detailed in Section 3.4.5:

- Security of supply, which motivates agents to satisfy their comfort need.
- Affordability, which motivates agents not to pay too much.
- Inclusiveness, which focuses on the equity of comforts between agents.
- Environmental Sustainability, which discourages transactions with the national grid.

Specific logic rules are detailed in Appendix C and argumentation graphs in Appendix D.

We design several scenarii, which notably depend on the 2 following variables:

- The environment size, i.e., the number of learning agents.
- The consumption profile, i.e., *annual* or *daily*.

These variables are exactly the same as in the previous chapter. We recall that a *small* environment corresponds to 20 Households, 5 Offices, and 1 School, whereas a *medium* environment corresponds to 80 Households, 20 Offices, and 1 School. The *annual* profile contains a consumption need for every hour of every day in a year, which we use as a target for agents: they want to consume as much as their need indicates; the *daily* profile is averaged on a single day of the year, and therefore does not contain the seasonal variations.

Additional variables are added separately for logic-based and argumentation-based experiments. In the sequel, we first describe experiments for logic-based agents, and then for argumentation-based agents.

5.4.1 Learning with LAJIMA judging agents

Our experiments leverage the Ethicaa agents (Cointe et al., 2016), which use the JaCaMo platform (Boissier, Bordini, Hübner, Ricci, & Santi, 2013). JaCaMo is a platform for

Multi-Agent Programming that combines the Jason language to implement agents with CArtAgo for specifying environments with which agents can interact, and Moise for the organization aspect. We will particularly focus on the agent aspect, i.e., the Jason language (Bordini, Hübner, & Wooldridge, 2007), which is derived from AgentSpeak(L), itself somewhat similar to Prolog. The CArtAgo environment, defined in the Java programming language, allows us to connect the simulation and learning algorithms to the judging agents, by representing them as an artifact, storing the received observations and actions, and thus making them available to the judging agents. In addition, the CArtAgo environment is also responsible for sending rewards back to the simulation after the judgment process occurred. Judges continuously observe the shared environment and detect when new data have been received: this triggers their judgment plan. They push their judgments, i.e., the rewards, to the shared environment; once all rewards have been pushed, by all judging agents and for all learning agents, the environment sends them to the learning algorithm.

However, as we mentioned, JaCaMo relies on the Java language and virtual machine; our learning algorithms and Smart Grid simulator are developed in Python, we thus needed a way to bridge the two different codes. To do so, we took inspiration from an existing work that tries to bridge the gap between BDI agents, especially in JaCaMo, and RL (Bosello & Ricci, 2020), by adding a web server on the simulator and learning side. The CArtAgo environment then communicates with the simulator and learning algorithms through regular web requests: to ask whether the simulation is finished, whether a new step is ready for judgment, to obtain the data for judgment, and for sending back the rewards.

We now describe the scenarii that we designed, and which principally depend upon the configuration of judging agents. Indeed, we want to evaluate the ability to adapt to changing environment dynamics, especially in the reward function. Agentifying this reward function offers a new, flexible way to provoke changes in the reward function, by modifying the set of judging agents, e.g., adding, or removing. We thus developed the judges so that they can be activated or disabled at a given time step, through their initial beliefs, which are specified in the JaCaMo configuration file. The following configurations were used:

- *Default*: all judging agents are activated for the whole experiment: the rewards aggregate all moral values.
- *Mono-values*: 4 different configurations, one for each moral value, in which only a single judging agent is enabled for the whole experiment. For example, in

the *affordability* configuration, only the *affordability* agent actually judges the behaviours. These scenarii serve as baselines.

- *Incremental*: At the beginning, only *affordability* is activated and produces feedbacks. We then enable the other agents one-by-one: at $t = 2000$, we add *environmental sustainability*, then *inclusiveness* at $t = 4000$, and finally *supply security* at $t = 6000$. From $t = 6000$ and up, all judging agents are activated at the same time.
- *Decremental*: Conversely to the *incremental* scenario, at the beginning, all judging agents are enabled. We then disable them one-by-one: at $t = 2000$, *environmental sustainability* is removed, then *inclusiveness* at $t = 4000$, and finally *supply security* at $t = 6000$. Afterwards, only the *affordability* agent remains.

5.4.2 Learning with AJAR judging agents

Similarly to the previous experiments, we designed several scenarii that are based on the following variables:

- The judgment function $\mathtt{J}_j$ used to transform a grounded extension, and more specifically its $\mathcal{F}_p$ and $\mathcal{F}_c$ sets of arguments, into a single number.
- The aggregation function $\mathtt{g}_{\mathtt{agr}}$ used to transform the different rewards returned by the judging agents into a single reward.
- The configuration of judges, i.e., which judges are activated and when.

The *environment size* and *consumption profile* are also used, as in the previous chapter and in the logic-based experiments. The other variables are described below with their possible values.

The judgment function

We recall that the judgment function $\mathtt{J}_j$ is the last step of the judging process, before the aggregation. It takes as input the computed grounded extension of the argumentation graph, i.e., the graph that contains only arguments that are both *alive* and *acceptable*. Some of these arguments are in favour of the decision "The learning agent's action was moral with respect to the moral value" and are said to be *pros* ($\mathcal{F}_p$), whereas other counter this decision and are said to be *cons* ($\mathcal{F}_c$). The rest of the arguments are neutral. From this grounded extension grd and the sets $\mathcal{F}_p$, $\mathcal{F}_c$, the $\mathtt{J}_j$ function must output a

reward, i.e., a single number, that corresponds to which degree the agent respected the moral value.

Contrary to the logic-based LAJIMA model, where the number of moral valuations is fixed, in the argumentation-based AJAR framework, the number of moral valuations can be any number, only restricted by the total number of arguments. Indeed, the moral valuations are given by the $\mathcal{F}_p$ and the $\mathcal{F}_c$, and any argument can be a pro or con argument. Thus, many methods can be imagined for this judgment function. We have designed the following functions, which basically rely at some point on counting the number of $\mathcal{F}_p$ and $\mathcal{F}_c$, and comparing them. However, they differ in the exact details, which impact the gradient offered by the function, i.e., the possible values. Some functions will, for example, only be able to return $\frac{0}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}$, which has a low gradient (only 4 different values).

simple Function that simply compares the number of $\mathcal{F}_p$ with the sum of $\mathcal{F}_p$ and $\mathcal{F}_c$ in grd.

$$\mathrm{J}_{simple} = \begin{cases} \frac{|grd_{[\mathcal{F}_p]}|}{|grd_{[\mathcal{F}_p]}|+|grd_{[\mathcal{F}_c]}|} & \text{if } |grd_{[\mathcal{F}_p]}| + |grd_{[\mathcal{F}_c]}| \neq 0 \\ \frac{1}{2} & \text{else} \end{cases}$$

This function is simple and intuitive to understand, however, it fails to take into account difference when there are no arguments in $\mathcal{F}_p$ or in $\mathcal{F}_c$. To illustrate this, let us assume that, at a given step t, there were 3 arguments in $grd_{[\mathcal{F}_p]}$, and 0 in $grd_{[\mathcal{F}_c]}$. The result is thus $\frac{3}{3} = 1$. This correctly represents the fact that, at t, the action only had positive evaluations (for this specific moral value). Let us also assume that, at another step t', there were still 0 arguments in $grd_{[\mathcal{F}_c]}$, but this time 4 arguments in $grd_{[\mathcal{F}_p]}$. One would intuitively believe that the action at t' was better than the one at t: it received more positive evaluations. However, this function returns in this case $\frac{4}{4} = 1$, which is the same reward. There is virtually no difference between an action with 3 or 4 pros arguments, as long as there is 0 cons arguments. The exact same problem applies with 0 arguments in $\mathcal{F}_p$, which would invariably return $\frac{0}{x} = 0$.

diff Function that compares first the number of $\mathcal{F}_p$ in the grounded with the total number of $\mathcal{F}_p$ in the unfiltered, original, argumentation graph, then does the same for the number of $\mathcal{F}_c$, and finally compares the $\mathcal{F}_p$ and the $\mathcal{F}_c$.

$$\mathrm{J}_{diff} = \frac{|grd_{[\mathcal{F}_p]}|}{|AF_{[\mathcal{F}_p]}|} - \frac{|grd_{[\mathcal{F}_c]}|}{|AF_{[\mathcal{F}_c]}|}$$

Thus, this function is able to take into account the existence of non-activated arguments, either pros or cons, when creating the reward. For example, if only 3 of the 4 possible $\mathcal{F}_p$ arguments are activated, the function will not return 1. Intuitively, this can be thought as "You have done well... But you could have done better." In this sense, this function seems better than the previous one. However, a problem arises if some arguments are too hard to activate, and therefore (almost) never present in the grd, or if 2 arguments cannot be activated at the same time. The agent will thus never get 1, or conversely 0 in the case of $\mathcal{F}_c$, because the function relies on the assumption that all arguments can be activated at the same time.

ratio Function that compares the sum of the number of $\mathcal{F}_p$ and $\mathcal{F}_c$ with the maximum known of activated $\mathcal{F}_p$ and $\mathcal{F}_c$ at the same time.

$$
\begin{aligned}
top &= |grd_{[\mathcal{F}_p]}|^2 - |grd_{[\mathcal{F}_c]}|^2 \\
down &= |grd_{[\mathcal{F}_p]}| + |grd_{[\mathcal{F}_c]}| \\
max\_count_t &= \max(max\_count_{t-1}, down) \\
\mathrm{J}_{ratio} &= \frac{top}{max\_count_t}
\end{aligned}
$$

This function thus avoids the pitfalls of the previous one, by only comparing with a number of activated arguments that is realistically attainable. However, it still suffers from some drawbacks. For example, we cannot compute *a priori* the maximum number of activated arguments; instead, we maintain and update a global counter. This means that the rewards' semantics may change over the time steps: a $\frac{1}{2}$ might be later judged as a $\frac{1}{3}$, simply because we discovered a new maximum. Taking this reasoning further, we can imagine that maybe, in a given simulation, the maximum known number will be 2 for most of the time steps, and finally during the later steps we will discover that in fact the maximum could attain 4. Thus, all previous rewards would have deserved to be $\frac{x}{4}$ instead of $\frac{x}{2}$, but we did not know that beforehand.

grad Function that takes into account the total possible number of $\mathcal{F}_p$ and $\mathcal{F}_c$ arguments.

$$
\mathrm{J}_{grad} = 0.5 + |grd_{[\mathcal{F}_p]}| \times \left(\frac{0.5}{|AF_{[\mathcal{F}_p]}|} \right) - |grd_{[\mathcal{F}_c]}| \times \left(\frac{0.5}{|AF_{[\mathcal{F}_c]}|} \right)
$$

This function creates a gradient between 0 and 1, with as many graduations between 0.5 and 1 as there are possible pros arguments, and conversely as many graduations between 0.5 and 0 as there are possible cons arguments. Each activated pro argument advances one graduation towards 1, whereas each activated con argument

advances one graduation towards 0, with a base start of 0.5. This function is similar to *diff*, except that it stays within the $[0, 1]$ range.

offset Function that avoids division by 0 by simply offsetting the number of activated arguments.

$$\mathsf{J}_{offset} = \min\left(1, \frac{1 + |grd_{[\mathcal{F}_p]}|}{1 + |grd_{[\mathcal{F}_c]}|}\right)$$

Since neither the numerator nor the denominator can be 0, adding one $\mathcal{F}_c$ argument without changing the number of $\mathcal{F}_p$ would effectively change the resulting reward. For example, assuming $|\mathcal{F}_p| = 0$, and $|\mathcal{F}_c| = 3$, the reward would be $\frac{1}{4}$. Adding one $\mathcal{F}_c$ argument would yield $\frac{1}{5}$, thus effectively reducing the reward. However, to avoid having rewards superior to 1, the use of the $\min$ function prevents the same effect on $\mathcal{F}_p$: if $|\mathcal{F}_c| = 0$, the reward will be 1 no matter how many $\mathcal{F}_p$ are present.

The aggregation function

As for the logic-based judgment, we have several judges that each provides a reward for every learning agent. Yet, the current learning agents expect a single reward: we thus need to aggregate.

We retain the *average* aggregation that was proposed for the logic-based, as it is a quite simple, intuitive, and often used method. Another option is to use a *min* aggregation that is based on the Aristotle principle. The Aristotelian ethics recommends to choose the action for which the worst consequence is the least immoral (Ganascia, 2007b). Adapting this reasoning to our judgment for learning, we focus on the worst value so that the agent will learn to improve the worst consequence of its behaviour. The final reward is thus the minimum of the judges' rewards. The reward signal thus always focuses on the least mastered of the moral values. For example, assuming the agent learned to perfectly exhibit the inclusiveness value and always receives a reward of 1 for this moral value, but has not yet learned the value of environmental sustainability, and receives a reward of 0.2 for this second moral value, the reward will be 0.2 so that the agent is forced to focus on the environmental sustainability to improve its reward. When the agent's mastery of the environmental sustainability exceeds that of other values, its associated reward no longer is the minimum, and another reward is selected so that the agent focuses on the new lowest, and so on. Thus, it emphasizes the learning of all moral values, not strictly at the same time, but in a short interval. The agent should not be able to "leave aside"

one of the moral values: if this value's reward becomes the lowest, it will penalize the agent. Formally, this function is simply defined as:

$$\mathtt{g}_{\mathtt{min}}\left(\mathtt{F}(l)\right) = \min\left(\{\forall f_{l,j} \in \mathtt{F}(l)\}\right)$$

where $\mathtt{F}(l)$ is the set of rewards for the learning agent l, of size $|\mathcal{J}|$, and $f_{l,j}$ is the specific reward for learning agent l determined by judging agent j.

However, I noticed that in some cases this function could become "stuck". For example, if one of the argumentation graph is ill-defined, and it is impossible to get a high reward for one of the moral values, e.g., more than 0.2. In this case, as long as all other moral values have an associated reward $\gneqq 0.2$, they will not be the minimum, and thus not selected by the $\mathtt{g}_{min}$ aggregation. The agent therefore cannot receive feedback on its behaviour with respect to the other moral values. It would be difficult to improve further these moral values without feedback: even if the agent tries an action that improves another moral value, the "stuck" one will remain at 0.2, and thus be selected again, and the final reward will be 0.2. The received signal would seem to (wrongly) indicate that this action did not improve anything. It would not be a problem in the case of a few steps, because the agent would have later steps to learn other moral values. However, in this example scenario, as one of the moral values is "stuck", even later steps would rise the same issue.

To potentially solve this concern, I proposed to introduce stochasticity in the aggregation function: the function should, most of the time, return the lowest reward so as to benefit from advantages of the $\mathtt{g}_{\mathtt{min}}$ function, but, from time to time, return another reward, just in case we are stuck. This would allow agents to learn other moral values when they fail to improve a specific one, while still penalizing this "stuck" value and avoid skipping the learning of this value. To do so, we propose the *Weighted Random* function, in which each reward has a probability of being selected, which is inversely proportional to its value, relatively to all other rewards. For example, if a reward is 0.1, and the other rewards are 0.8 and 0.9, then the 0.1 will have an extremely high probability of being selected. Yet, the two others will have a non-zero probability. Formally, we define the function as follows:

$$\mathtt{g}_{\mathtt{WR}}\left(\mathtt{F}(l)\right) = \text{draw } f_{l,X} \sim P(X = j) = \frac{1 - f_{l,j}}{\sum_{k \in \mathcal{J}} (1 - f_{l,k})}$$

where X is a random variable over the different judging agents, $\mathcal{J}$, and $f_{l,X}$ is the reward determined by judging agent X.

Configuration of judges

Finally, we propose, similarly to the logic-based judges, several configurations of argumentation judges. Judges can be enabled or disabled at a given time step, and we thus use it to evaluate the adaptation to changing reward functions. The 3 following configurations have been tested:

- *Default*: all judging agents are activated, all the time.
- *Incremental*: Only the affordability agent is enabled at the beginning, then the environmental sustainability agent is activated at $t = 2000$, followed by the inclusiveness agent at $t = 4000$, and finally the supply security agent at $t = 6000$. All agents are thus enabled after $t = 6000$.
- *Decremental*: Basically the opposite of *incremental*, all agents are initially activated. At $t = 2000$, the environmental sustainability agent is disabled, the inclusiveness agent is disabled at $t = 4000$, and finally the supply security agent at $t = 6000$. Only remains the affordability agent after $t = 6000$.

5.5 Results

We first present the results for the LAJIMA model, using logic-based agents, and then for the AJAR framework, using argumentation-based agents. As for the previous chapter, the *score* here is defined as the average global reward at the end of the simulation, where the global reward is the mean of rewards received individually by learning agents, at a given step.

5.5.1 Learning with LAJIMA judging agents

Figure 5.6 shows the results for the experiments on each scenario. Results show that, on most scenarii, Q-SOM and Q-DSOM algorithms perform similarly, except on the *env_sustain* mono-value configuration, i.e., when only the judge associated to the Environmental Sustainability value is activated. Some moral values seem easier to learn, e.g.,

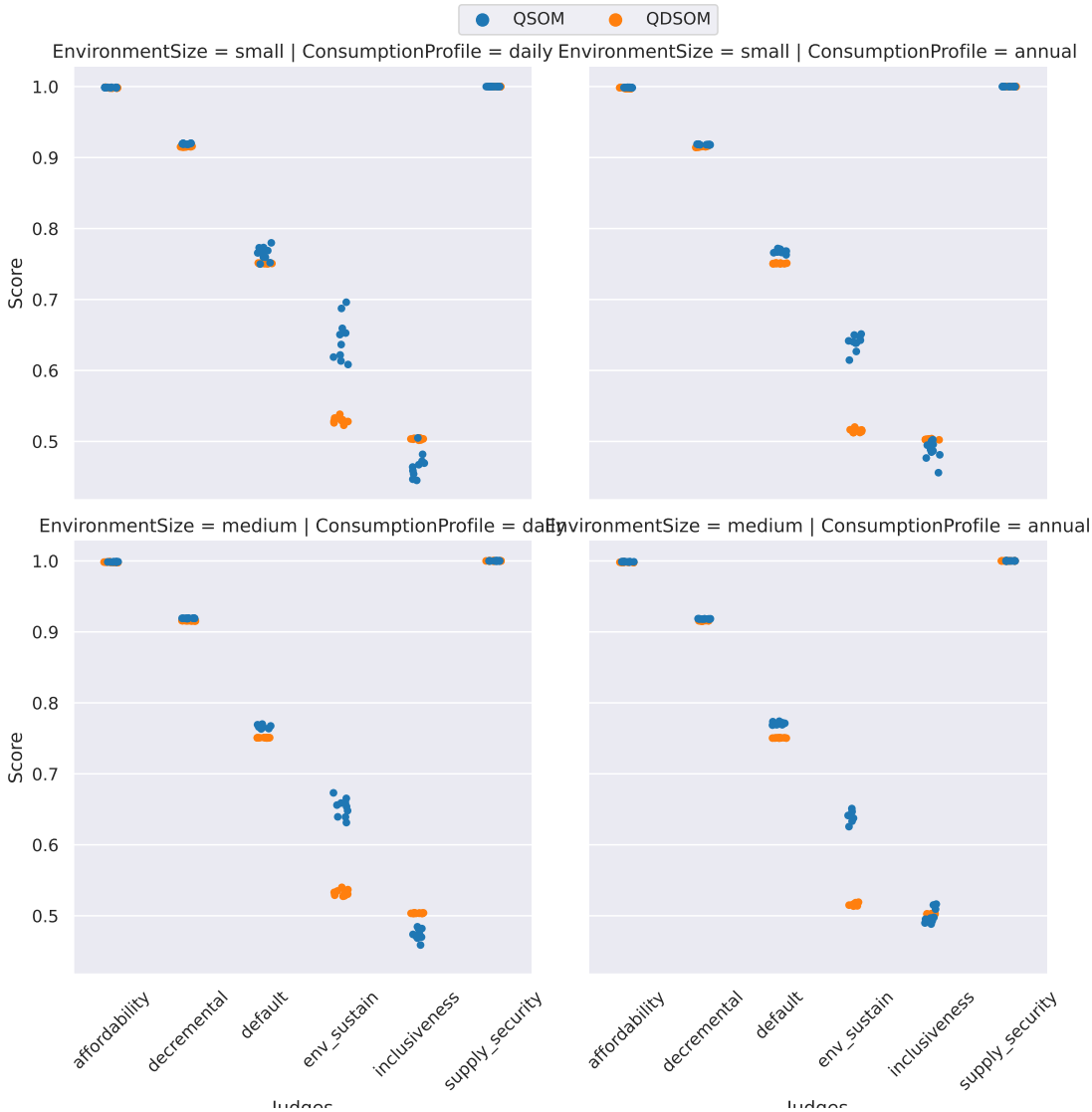


Fig. 5.6.: Results of the learning algorithms on 10 runs for each scenario, when using the LAJIMA agents.

the *affordability* and *supply_security* scenarii, which consistently have a score of nearly 1. Others, such as the *inclusiveness*, seem harder to learn, but nevertheless attain a score of 0.5.

A statistical comparison between the *small* and *medium* sizes of environments, using the Wilcoxon test, does not show a significant difference in the obtained difference (p-value = 0.73751; we cannot accept the alternative hypothesis of a non-0 difference). This shows the scalability of our approach.

5.5.2 Learning with AJAR judging agents

Figure 5.7 shows the scores for the argumentation-based experiments. As we have detailed, in the AJAR framework, 2 additional parameters must be set: the judgment function J_j that returns a single number from a set of moral evaluations, and the aggregation function g_{agr} that returns a final reward from the set of judgments for each judge. We can see that, for every choice of J_j and g_{agr}, and for every scenario, the learning algorithms managed to attain a very high score, superior to 0.9. This means that the judgment functions offer a good enough gradient to allow learning algorithms derive the correct behaviour from the reward signal.

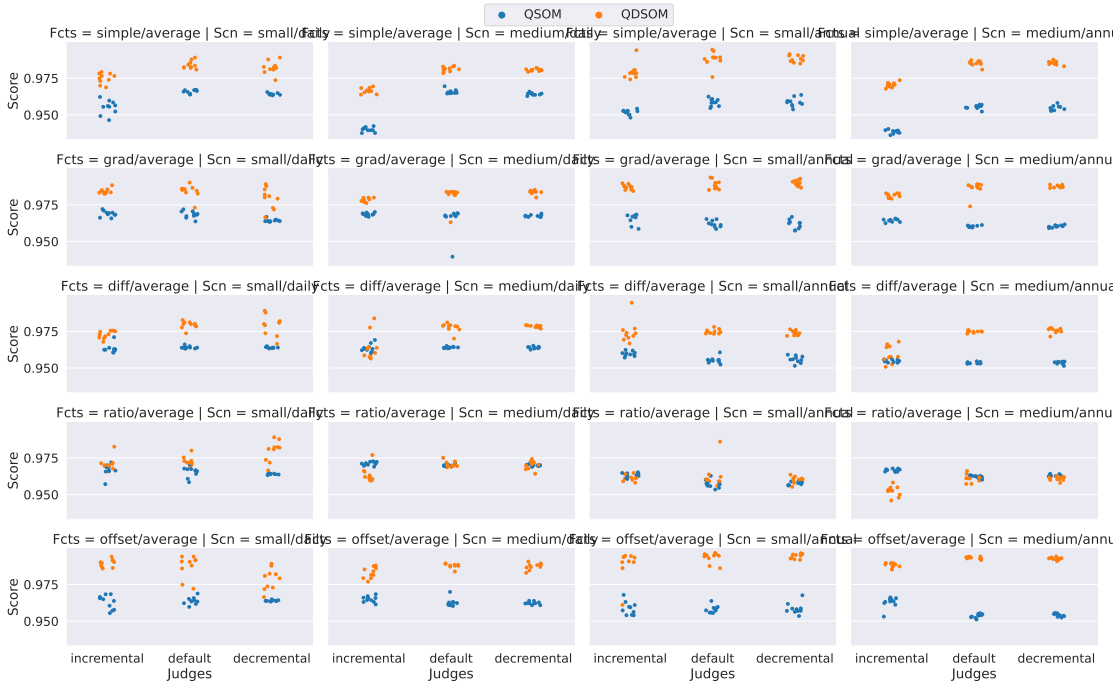


Fig. 5.7.: Results of the learning algorithms on 10 runs for each scenario, when using the argumentation-based judging agents. To simplify the plot, *Scn* (Scenario) regroups EnvironmentSize and ConsumptionProfile; *Fcts* (Functions) combines the choice of judgment and aggregation functions.

However, they are differences between different judgment functions. The *offset*, *grad*, and *simple* seem indeed to yield slightly higher rewards.

Interestingly, we can note that the Q-DSOM algorithm performs better on most of the experiments, compared to the Q-SOM algorithm. This is confirmed by a Wilcoxon test, using the "greater" alternative (p-value = < 2.22e-16).

5.6 Discussion

In this chapter, we presented a new line of research for constructing reward functions, through the use of symbolic judgment. More specifically, we proposed to introduce new judging agents to the multi-agent system, which judge the behaviour of the learning agents to compute their rewards, by leveraging symbolic reasoning, through either logic-based rules or argumentation frameworks. To the best of our knowledge, this line of research has not been studied, especially within Machine Ethics.

We recall that the principal objectives we identified from the state of the art were:

- How to correctly judge a learning agent's behaviour, and particularly avoiding reward gaming?
- How to include a diversity of moral values?

By using symbolic judgments, we can leverage domain experts' knowledge to correctly judge the learning agents. This is even more true with argumentation-based judgments, as the attack relationship can be leveraged to disable arguments in a specific context. We have given the fictional example of an agent that learned to hoard then give energy, to demonstrate how this kind of reward gaming could be fixed, by adding a new argument that checks whether the agent hoarded energy at the previous step. When this argument is true, the arguments that would tend to reward the agent for giving energy are disabled through the attacks. Using argumentation also helps with identifying a behaviour that exhibits reward gaming, by looking at arguments' activation throughout the time steps.

Multiple moral values can be represented by different judging agents; using distinct agents instead of a single one offers several advantages. The agents can be individually added, updated, or removed, thus giving the designers control over the represented moral values in the system at runtime. Additionally, this paves the way to more complex interactions between the judging agents, which we will detail later.

The methods proposed in this chapter, logic-based and argumentation-based, have various specificities, and we now compare them, both to the traditional mathematical reward functions, and between them. We note that, as we have 2 different implementations, some differences, advantages or drawbacks, emerge from the technical implementation details, whereas others stems from more fundamental elements. The implementation differences are still important to note, even though they do not reflect a fundamental flaw or inherent advantage of the method; we will therefore particularly emphasize whether

our remark is related to a technical implementation choice, or a fundamental conception, in the sequel. We also highlight the limitations and perspectives that arise from these methods.

First, we compare the two methods to the traditional mathematical functions, as used in the previous chapter. As we saw in the state of the art, multiple works have tried to implement ethical principles or moral values as symbolic rules. Such formalization seems therefore appropriate, when comparing to mathematical formulas, or more general programming languages. However, we note that one disadvantage of our chosen symbolic methods, when applied to the problem of generating rewards for learning agents, is to offer a poorer gradient than mathematical formulas, as we have already mentioned. By gradient, we mean that a difference in the situation and/or action should result in an equivalent difference in the reward, no matter how small the initial difference may be. The thinnest the gradient is, the more agents will learn when trying new behaviours: by exploring a new action, the result will be different, and this difference will be reflected in the reward, either positively or negatively. If the reward is better than before, the agent will then learn that this new action was a good one, and will retain it, rather than the old one. On the other hand, if the gradient is too coarse, the action difference induced by the learning agent risks not being correctly captured by the judgment. This is particularly visible when we think of thresholds for example: let us assume that we evaluate the action as moral if the agent did not consume more than 10% more energy than the average. At a first step t_0, the agent may consume 20% more, and thus not being positively rewarded. At a second step t_1, the agent may randomly consume 12% more than the average: the action is clearly better, however it is still above the threshold of 10%, and thus the agent is still not positively rewarded. It therefore has no feedback indicating that its action was better, yet not sufficient, but that it could continue in this direction. On contrary, mathematical functions offer a gradient by principle: let us take for example, $R = consumed - (average \times 1.1)$, which represents roughly the same idea: the agent should not consume more than 110% of the average. We can clearly see that the reward will be different when the agent consumes 20% more, or 12%. It would even be different if the agent consumed 19.99999% more. This problem might be mitigated by other techniques, such as fuzzy logic (Zadeh, 1965), or weighted argumentation frameworks (Coste-Marquis et al., 2012), for example, which handle more naturally continuous values in addition to symbols.

This problem of gradient is linked with the notion of *moral evaluation* that we proposed in our methods. The symbolic rules produce these evaluations, which are in the form of

$moral$ and $immoral$ symbols in the LAJIMA model, or represented by pros ($\mathcal{F}_p$) and cons ($\mathcal{F}_c$) arguments in AJAR. Then, the reward is determined based on these evaluations, the number of positive evaluations compared to the negative ones, etc. We note a first difference, which is purely an implementation detail, between our two methods: in LAJIMA, we have chosen to produce exactly as much evaluation as there are dimensions in an action; thus, each action dimension is judged and associated with a moral evaluation. On the contrary, in AJAR, we can have as many pros and cons arguments, and thus moral evaluations, as we desire. These 2 choices have both advantages and disadvantages: on the first hand, LAJIMA simplifies the problem by setting a fixed number of evaluations. Producing a reward can be as simple as dividing the number of $moral$ evaluations by the sum of $moral$ and $immoral$ evaluations. On the other hand, AJAR is more permissive, and offer more flexibility, in particular for the reward designers, who do not have to exactly set 6 evaluations, and may instead create the arguments as they see fit. It also opens the way for different methods to produce rewards: we proposed a few non-exhaustive options in our experiments to illustrate this. This may be seen as a disadvantage, since it requires an additional parameter for selecting the judgment function, which implies fine-tuning. Yet, this disadvantage seems justified, as the experiments tend to show a better gradient for the AJAR framework.

We note a first perspective here, about this judgment function; indeed, designing and selecting such a function to transform a set of symbols into a reward is not a trivial task. There is maybe more work to perform here, and we identify 2 potential (non-exhaustive) research avenues. First, the judgment aggregation theory domain may be leveraged, in order to correctly choose a number to represent a set of diverging evaluations. Secondly, an idea that is more specific to the AJAR framework, the argumentation graphs could be replaced by weighted ones. Indeed, we used a simple argumentation framework in our proof-of-concept work: the attack relationship is a simple, unidirectional one. If an argument is attacked, and not defended by another, it is considered killed. However, other argumentation frameworks exist in the literature, such as the weighted argumentation graphs (Coste-Marquis et al., 2012 ; Amgoud, Ben-Naim, Doder, & Vesic, 2017), where arguments and attacks are augmented by a notion of weight. An argument that has a significant weight will not be killed if it is attacked by an argument with a smaller weight. However, it might be killed if several arguments with small weights attack it, and so on. The idea here, to ultimately improve the judgment function, would thus be to rely on the remaining arguments' weights to produce the reward. If an argument is attacked but not killed, its weight will be reduced, so that we can say the agent "did well, but not enough": this avoids the pitfalls of thresholds mentioned earlier. To compute the

final reward, the judgment function could, e.g., sum the weights of pros arguments and subtracts the weights of cons ones.

Another difference that comes from a technical implementation detail is the kind of inputs used by judging agents. Indeed, the logic-based ones in LAJIMA rely on the individual learning agents' observations and actions: this is inspired by the Ethicaa work, which describes this as a partially informed judgment, i.e., a judgment where the judge uses knowledge from the judged agent, especially about the situation and the agent's action. Intuitively, we can see this kind of judgment as "Would I have done the same thing, had I been at this agent's place?". On contrary, in AJAR, we described the input of the argumentation graphs as the whole knowledge of the current state, e.g., including data about other agents. This allows for example to compute the average consumption, and compare the judged agent's consumption to the entire society. Since all data are included, the judging agents could also compare it to the median, or the 1st quartile, etc. This makes judging agents sort of "omniscient", thus making more accurate judgments, but potentially impairing the privacy, since more data are shared. However, we note that the judging agents are meant as some sort of central controller, and are external to the learning agents. Thus, a Smart Grid participant still cannot access their neighbors' data. There is most certainly an important question to ask and answer here, about the correct tradeoff between the accuracy of judgments, and the privacy of the human users. Some of them will probably not want to share data. We leave this question aside, as our work does not concern the acceptability of Smart Grid (or more generally, AI) systems, but we acknowledge that the ability to correctly judge, and ultimately learn, ethical behaviours may be limited by this acceptability concern.

One of the fundamental differences between our 2 proposed models is the kind of symbolic elements that are leveraged. In LAJIMA, logic rules are used, along with beliefs, and plans, whereas AJAR uses arguments and graphs. This has an impact on the ability to read and understand the reward function, i.e., the expected behaviour, by designers first, but more importantly by non-experts people. For example, the well-known Mycin (Buchanan & Shortliffe, 1984) example has shown that expert systems, which rely on similar rules to our LAJIMA model, were difficult to understand by doctors, although it could theoretically "explain" its reasoning by showing the trace of rules that were triggered. More importantly, it was difficult for users to integrate their own knowledge in the system. Arguments' activation allows visualizing the reasoning as a graph and see the attacks as a map, which may be easier to understand for lay users rather than plans and beliefs. Graphs are updated by adding (or removing) arguments (nodes), and

specifying the attacks on other arguments, visually represented by edges; this, again, may be easier to use rather than rules laid out in a textual format, for domain experts who are not versed in AI development. However, we did not test this hypothesis with a Human-Computer Interaction experiment, and more work is required on this subject.

Both our models have a small limitation linked to the simplification of their definition. Indeed, we consider a set of judging agents, with each judge representing a moral value, and we define a symbolic judgment with respect to this moral value. However, one might argue that moral values are not the only important objectives: there might be some others, that we denote "technical aims", or aims for short, which are important for the system but not especially linked to a moral value. An example of such aims may be, for example, to correctly balance the production and consumption within a Smart Grid. If these aims can be computed as a symbolic judgment, they can be added to the system exactly as the moral values are, without loss of generality. However, some aims may be easier to design as a mathematical function, as the example above, which can be simply formulated as a difference between the production and the consumption. The current model does not allow this. The formal definitions of the model can be extended to consider not only the set of judging agents, but rather the union of judges with aims functions. Thus, the aggregation function takes the symbolic-produced rewards for moral values and the rewards for aims. Considering this union is not difficult from a technical point of view, but complicates the definition and diverts attention from the important part of the contribution, we thus chose to leave it aside.

Finally, another important element is the choice of the aggregation function. Indeed, the learning algorithms, in their current form, require a single reward per learning agent, whereas we have multiple judging agents representing various moral values. We thus need to aggregate, and we proposed several methods to do so: the average function, which is probably the most classical and intuitive way, the min function, which is almost as classical, and finally our custom Weighted Random, which mimics the behaviour of min while adding stochasticity to avoid "stuck" situations. Perhaps other functions could be explored, e.g., in computational social choice (Chevaleyre, Endriss, Lang, & Maudet, 2007), and more specifically in multi-criteria decisions, several approaches have been proposed to replace min-based aggregation, such as *ordered weighted min*, or *leximin* (Dubois, Fargier, & Prade, 1997). Basically, the idea of these approaches is to introduce some kind of preference or priority over the various criteria. They have been thoroughly explored within the field of social choice, especially for fair division, for decision-making purposes. However, in our case, the "decision" is not to select an action,

e.g., a way to distribute resources among agents, but rather to select a reward that will be used to learn the correct action. This (subtle) difference calls for more research and experiments to apply such approaches to the problem of reward selection, in order to avoid pitfalls. For example, who should set the preferences over moral values? Should it be the designers, using the same preferences for all agents, or the users, using their own (different) preferences for their own agent? In this second case, would users be able to make their agent "ignore" of the moral values by carefully crafting their preferences? Should we allow them this possibility? And how can we prevent them from doing so?

The choice of the aggregation function is important for the resulting behaviour, as it determines how the moral values are learned, in which order, how the system handles values of different "difficulties", and situations where values are in conflict. Yet, the scalarization idea is not ideal: by principle, scalarizing makes us lose information. Whether we take an average, a min, or a random reward among those proposed, we still send less feedback to the agent than with the full set of rewards. We have identified 2 groups of ideas to replace it: 1) An improved interaction between judging agents, e.g., through negotiation. In a sense, the judgment aggregation theory mentioned earlier could belong to this group. Thus, the final reward is itself constructed by reasoning. If, for example, one of the judging agents detects that its moral value is not correctly learned, because the learning agent keeps repeating the same mistakes, it can negotiate with other judges to emphasize its reward, or on contrary to ignore it, in order to focus on other moral values and avoid preventing their learning. This can be achieved, e.g., by negotiation between the judging agents themselves, or by creating a meta-reasoning agent, which takes the results of the lower-level judges as inputs, and outputs the final reward. This meta-judge could therefore include ethical principles to decide, whereas the other judges would include rules that concern only their moral value. This idea could be extended to create rewards that do not only depend on the current judgments at step t, but on the whole history of the learning agent. For example, the meta-judge could decide that, although the learning agent's action was good at this step, it could have been better, and thus award only a small amount to the learning agent, to encourage it to improve. Or, on the contrary, the meta-judge could give a higher reward even though some moral value is not respected, to help the agent learn the other values first, in some sort of adversarial or active learning loop. 2) Instead of aggregating the judgments per moral value into a single reward for each learning agent, we could simply send all feedbacks to them. Thus, the learning agents get more information, and can identify situations where 2 moral values are in conflict. This new ability can be exploited to inject human users'

preferences inside the learning agents to resolve these conflict situations, in a way that is specific to each learning agent, and thus to each human user.

We note that, by aggregating the moral values, we settle potential dilemmas *a priori*. This prevents the learning agents from correctly handling these dilemmas, and even recognizing there was a dilemma in the first place. We will therefore, in the next chapter, explore this 2nd family of ideas, which leads to a multi-objective approach for the learning algorithm. Scalarizing was necessary as a first step, in order to evaluate the contribution of agentifying the reward function and leveraging symbolic judgments on its own. Then, we "open up the box", by sending directly multiple rewards, one for each moral value, to each learning agent. This way, they can compare the rewards, recognize situations of dilemmas, and each learning agent can settle dilemmas in a different way, depending on both the context and their user's preferences. For example, user A may favor the inclusiveness value, and thus the learning agent will choose actions that yield maximal rewards for this value when it cannot optimize all of them. Another user, B, may favor on contrary the affordability value, and thus its own learning agent will learn differently from user A's agent. In a different context, e.g., in winter instead of summer, perhaps user A may also favor the affordability value, and the learning agents will have to learn this.

6 Identifying and addressing dilemmas with contextualized user preferences

We present in this chapter the final and third contribution, which relates to the "dilemma management" part from the conceptual architecture in Figure 3.2. Section 6.1 begins with an overview of the contribution, explains what is missing in the previous contribution, and why it is necessary to focus on dilemmas. Then, Sections 6.2, 6.3, and 6.4 describe the 3 steps of our contribution. These steps are summarized in Section 6.5, along with two algorithms that formalize and recapitulate the operations performed in these steps. The experimental setup is presented in Section 6.6, and the results reported in Section 6.7. We finally discuss this contribution's benefits, limitations, and perspectives in 6.8.

6.1 Overview

In the previous chapters, we have first proposed reinforcement learning algorithms to learn behaviours aligned with moral values, and then a new way of constructing reward functions through symbolic reasoning by judging agents, with respect to several moral values. However, the reward, as a scalar number, does not detail the agents' performances for each moral value. This is due to the reinforcement learning algorithms themselves, which expect a scalar reward, and is true whether we consider the mathematical functions in Chapter 4, such as *multi-objective sum*, or the symbolic-based judgments in Chapter 5, which were aggregated. This is problematic, as it necessarily impoverishes the feedback sent to learning agents, and hides some details, such as conflicts between moral values.

Any aggregation function has some "collisions", i.e., different feedbacks that yield the same reward. These collisions result in virtually no difference in the reward between two very different situations, with strong ethical significance. The question of which feedbacks collide and result in the same reward depends on the aggregation function

itself. For example, using an average, $\texttt{average}\,(\{0.5, 0.5\}) = \texttt{average}\,(\{1, 0\})$. Similarly, using a min, $\texttt{min}\,(\{0.2, 0.2\}) = \texttt{min}\,(\{0.2, 0.8\})$. In these examples, the sets of numbers are feedbacks that represent specific rewards for two different moral values. We can see that a first feedback with equal satisfaction (or defeat) of both moral values can have the same aggregated reward as another, very different feedback, where there is a significant difference between the moral values' rewards. As we illustrate just below, such differences between feedbacks that can be obfuscated by aggregation can be very significant on the ethical point of view. We may need to keep track of them and handle the tensions they reflect, with respect to moral values.

We recall that rewards are the signal that is used to learn the interests, or Q-Values, of actions in given states. The interests, in turn, are used to select the most promising action in the current state. Let us consider the following example: in a given situation, we have learned the interests of 3 different actions, which are all 0.5. The actions therefore seems comparable, and it would be as interesting to select the first as the second or third. The agent may simply roll a die to choose one of them. However, had we learned interests *per moral value*, instead of aggregated ones, perhaps we would have a completely different story. The actions' vector interests could be, e.g., $\texttt{Q}(s, a_1) = [0.5, 0.5]$, $\texttt{Q}(s, a_2) = [1, 0]$, and $\texttt{Q}(s, a_3) = [0, 1]$, indicating that the first action a_1 has the same, medium interest for both moral values. On the contrary, the second action a_2 has a high interest for the first moral value, and a low interest for the second moral value, and a_3 mirrors a_2 with a low interest for the first moral value, and a high interest for the second. Thus, there truly is an important choice to be made: do we want to satisfy all moral values equivalently? Or do we prefer one of the moral values, at the expense of the other? This decision should be made explicit, and deliberate, which is only possible if we have access to this knowledge. In the aggregated scenario, we did not even know that there was a trade-off in the first place.

This question of a deliberate decision also opens up the subject of human preferences. We have mentioned "do *we* want to satisfy all moral values equivalently?": in this sentence, the *we* should refer, we believe, to the human users' preferences. Or, more specifically, to the agent's preferences, which should match the human ones. As we mentioned in Chapters 1 and 3, the agents learn ethics according to an ethical intention that we, humans, inject in the system one way or another. Up to now, this *ethical injection* was limited to a feedback upon actions compliance to given moral values. We must now extend it to also include the ethical preferences over moral values, when a choice has to be made, because the agent does not know how to satisfy all moral values at once.

We argue that these preferences differ between users: we do not have the same priority order over moral values. They also differ between contexts, where context refers to the current situation: we may not prioritize the same moral value in summer than in winter, for example, depending on our resistance to heat or cold, as well as the cost of heating or cooling, or the country's climate. In this contribution, we thus propose to take into account this multi-objective aspect and to learn contextualized human preferences. This is related to our third research question:

How to learn to address dilemmas in situation, by first identifying them, and then settling them in interaction with human users? How to consider contextualized preferences in various situations of conflicts between multiple moral values?

To do so, we propose a Multi-Objective extension of the Q-SOM and Q-DSOM reinforcement learning algorithms, which we name *Q-(D)SOM-MORL*. This extension follows 3 steps, each one addressing a different need.

1. The first step is to learn "interesting" actions in all situations, so that we can identify dilemmas and make an informed decision, or choice, when presented with a dilemma.
2. The second step is to correctly identify dilemmas. We propose definitions, inspired from the literature, and refine them to better suit our continuous domains case.
3. The third and final step is to learn the user preferences, based on the interesting actions from step 1, in the dilemmas identified at step 2. We refine the vague notion of "context" mentioned earlier, and describe what is a user preference, and how we can map them to the different contexts so that agents learn to settle dilemmas using the correct, expected preferences. Dilemmas in the same context are settled in the same way, so as to reduce the amount of required interactions with the human users.

Figure 6.1 represents these 3 steps, and details the conceptual architecture proposed in Figure 3.2. Learning agents, on the right side of the figure, still receive observations from the environment, and output actions to the environment. However, their decision and learning algorithms now integrate additional data structures and processes, and learning agents are associated to a human user.

The *exploration profiles* focus on the 1st step, learning interesting actions, based on the observations of the current situation, and the multiple rewards received by the learning agent. Note that, instead of aggregating them and receiving a single scalar $\in \mathbb{R}$, as in the previous chapter, we now send directly all judgments from judging agents, and thus

the learning agents obtain multi-objective rewards $\in \mathbb{R}^m$, where m is the number of objectives. In our case, objectives correspond to the respect of moral values, and we thus equate "moral values" with "objectives" in this chapter. The 2nd step aims at identifying whether the current situation is a dilemma, either known or unknown, by leveraging exploration profiles, and a *human profile* which is learned based on interactions with a human user. Finally, the 3rd step selects an action: if the situation is not a dilemma, then the learning agent takes the best action; otherwise, it learns and re-uses the human contextualized preferences to settle the dilemma and choose an action, according to human preferences. These 3 steps are detailed in the following sections.

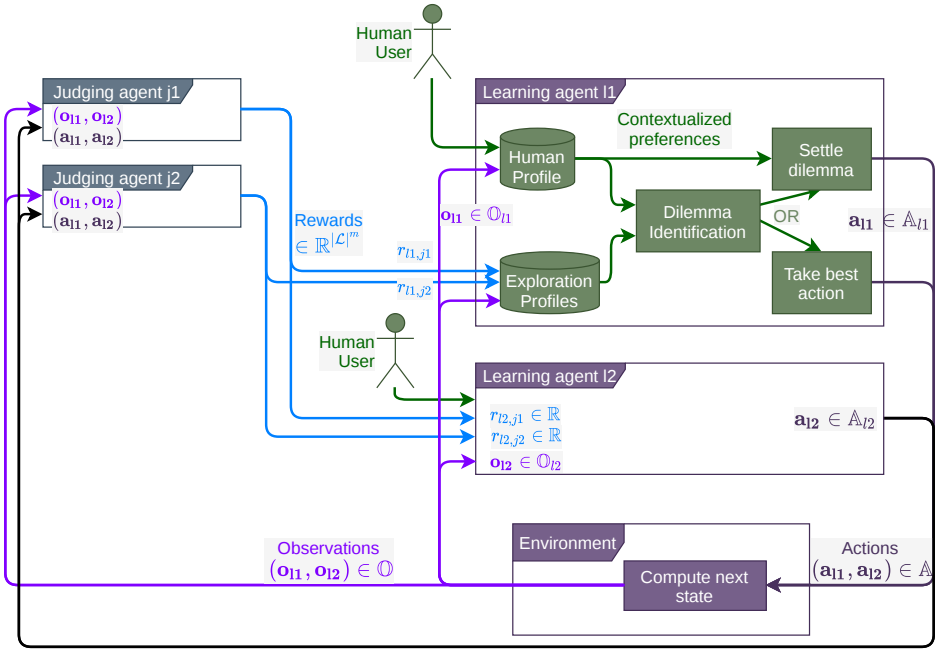


Fig. 6.1.: Architecture of the multi-objective contribution on the identification and settling of dilemmas by leveraging human contextualized preferences.

6.2 Learning interesting actions for informed decisions

The shift from single-objective RL to multi-objective RL (MORL) brings several changes to the Q-(D)SOM algorithms presented in Chapter 4. Mainly, the interests, or Q-Values, become vectors instead of scalars, and some equations are no longer defined for this kind of input, especially for selecting and exploring actions. In addition, as we want to leverage

human preferences, we need to present a comparison measure for the different actions. Specifically, 2 problems arise, which relate to the notion of "interesting actions":

1. How to correctly learn the actions' interests and select an action, compared to others, in the exploration-exploitation dilemma?
2. How to determine whether an action perturbed by a random noise, for exploration purpose, was interesting, with respect to previously learned action?

These 2 problems are detailed below; then, a solution involving the introduction of two distinct phases and *exploration profiles* is proposed and commented.

The first problem relates to the learning of actions in each situation, and more specifically of "interesting actions", which, in this case, means an action that is a suitable alternative as part of a dilemma. This is necessary, as the ultimate goal is to make agents settle dilemmas, i.e., making a choice in a trade-off between interesting actions, based on human users' preferences. To make this decision, both from the agents' and the humans' point of view, we need to know the actions' interests. We also recall that a reinforcement learning algorithm learns interests as actions are tried in the environment. Thus, during the learning, the interests at a given step might not reflect the "true" interests of the action. For example, let us assume that we have an action a_1 with interests $\texttt{Q}(s, a_1) = [0.8, 0.7]$. This action seems pretty good with respect to both moral values. Another action a_2 has other interests $\texttt{Q}(s, a_2) = [1, 0.3]$. This action seems better on the first objective, or moral value, but worse on the second objective. We might think we are in a dilemma, as we have to choose between prioritizing the first objective, or the second, by selecting one of these 2 actions. However, let us also consider that the first action was well explored, it was selected many times by the agent, and we are quite certain that the learned interests have converged very close to the true interests. On the other hand, the second action was almost not explored, the agent only tried it once. It might be possible that the received reward was a bit exceptional, due to some circumstances that do not often happen, and that, in fact, the true interests are closer to $[0.75, 0.3]$. In such case, action a_2 would not be interesting, but rather *dominated*, in the Pareto sense, by a_1, as $\texttt{Q}(s, a_1)$ would be strictly greater on each dimension.

This is what we mean by "interesting" actions in this first problem: to effectively compare them and determine whether there is a trade-off to be made, we need to have correctly learned their interests. If the interests are not known, we risk considering an action as a potential candidate, whereas in fact it should not be, or conversely ignoring an actually good action. In the previous example, perhaps a_1 in fact dominates a_2, or conversely

a_2 is strictly better than a_1. An action that is strictly dominated by another cannot be "interesting", as the other action would yield a better result for each moral value. Asking users for their preferences in an uncertain case like this would only bother them. Indeed, as the interests will be updated afterwards, it is unlikely the agent will have to retain the same preferences the next time it arrives in this situation. We thus need to learn the interests of all actions, as close as possible to their true interests, before we can begin identifying and settling dilemmas, and asking users for their preferences.

The second problem that emerges when considering multiple objectives, is to answer whether the explored, randomly noised action is "interesting", i.e., yields better interests than its original action. Indeed, we recall that, in the Q-SOM and Q-DSOM algorithms, in order to explore the actions space, a proposed action is first identified from the Q-Table. The action's parameters are taken from the prototype vector of the associated neuron in the Action-(D)SOM. Then, to explore and potentially find an even better action, a random noise is applied on these parameters. The explored, or "perturbed" action, is thus enacted in the environment, and the agent receives an associated reward. If this perturbed action was better than the proposed, learned action, the agent should update the Action-(D)SOM: to determine this, we used Equation (4.5), that we recall below:

$$r_t + \gamma \max_{j'} \mathtt{Q}(s_{t+1}, j') \overset{?}{>} \mathtt{Q}(s_t, j)$$

where j is the index of the proposed action. Basically, this equation means that if the received reward, and the maximum interest obtainable by taking an action in the new, resulting state, is higher than the learned interest of the proposed action, then the perturbed action is better than the proposed one, and the Action-(D)SOM should be updated towards the perturbed action.

However, this equation does not work any more in a multi-objective setting. Indeed, we replaced the 2-dimensional Q-Table by a 3-dimensional table, where the 3rd dimension is the moral value. In other words, we previously had $\mathtt{Q}(s, a) \in \mathbb{R}$, but we now have $\mathtt{Q}(s, a) \in \mathbb{R}^m$, and $\mathtt{Q}(s, a, k) \in \mathbb{R}$, where k is the moral value index $\in [[1, m]]$, with m the number of moral values. Similarly, the reward r previously was $\in \mathbb{R}$ but is now $\in \mathbb{R}^m$. The equation relied on taking the maximum interest and comparing 2 scalar values, which is trivial, but these relations are no longer defined in a vectorial space.

To solve these problems, we propose to change the Q-SOM and Q-DSOM algorithms, by introducing 2 distinct phases: a *bootstrap* phase, and a *deployment* phase. In the bootstrap phase, agents are tasked with learning "interesting" actions, i.e., both the

parameters that yield the best interests, and the true interests that correspond to these parameters, without focusing on dilemmas or user preferences. These, on the other hand, are focused on during the deployment phase, where the agents leverage their knowledge of "interesting" actions to identify dilemmas and learn to settle dilemmas according to contextualized user preferences.

Concerning the first problem more specifically, we propose to change the action selection mechanism, and to prioritize exploration. Indeed, since we now consider a bootstrap phase separated from the deployment, we do not need to maximize the expected sum of rewards during exploration. We can instead focus solely on exploring and learning the actions' interests. Taking inspiration from the Upper Confidence Bound method (Auer, Cesa-Bianchi, & Fischer, 2002), we memorize the number of times an action has been enacted, and we use this information as our new criterion. A simple and intuitive way to then ensure that all actions are correctly learned, is to always select the action with the minimum number of times enacted. Thus, at the end of the bootstrap phase, even if some actions have been less enacted, e.g., because the number of steps was not a multiple of the number of actions, we ensure that the "unfairness" is minimal. We detail in Section 6.6 several specific methods that build upon $\min$ and compare them.

The second problem of determining whether a perturbed action is more interesting than the previously learned action can be solve by making the formula valid again. One such way is to scalarize the vector components, but this would favour exploration in specific sub-zones of the action space. For example, if we use an average aggregation, we would consider an action interesting, i.e., better than the learned one, only if the perturbed actions has higher interests on average, on all moral values. If an action has a better reward on one specific moral value, but lower rewards on all other dimensions, it will not be learned. This would prevent discovering some actions that can still be part of a trade-off. To avoid this, during the bootstrap phase, we introduce the notion of *exploration profiles* to the learning agents. Each exploration profile contains a vector of weights, which intuitively tells us which zone of the action space will be explored by this profile. For example, the weights of an exploration profile might be $[0.9, 0.033, 0.033, 0.033]$, which will focus on actions that yield high interest on the first moral value. To also discover actions that yield high interest on the second moral value, and so on, we create multiple exploration profiles, thus partitioning the action space between these profiles.

Additionally, as different exploration profiles will learn differently, notably updating their Action-(D)SOMs at various time steps, we also place the data structures in exploration profiles.

Definition 6.1 (Exploration profile)**.** An exploration profile $p \in \mathbb{P}$ is defined as the following data structures and functions:

- State-(D)SOM: the "map" of situations to discrete states, which contains a fixed number of neurons, and thus of possible states. We define the set of possible states identifiers as $\mathcal{S} = [[0, \cdots, |\mathbf{U}|]]$, where $|\mathbf{U}|$ is the number of neurons.
  - $\texttt{States}_p : \mathbb{O} \to \mathcal{S}$ is the function that returns a discrete state identifier for any observation vector, based on the profile p.
- Action-(D)SOM: the "map" of action identifiers to action parameters, which also contains a fixed number of neurons, and thus of possible actions. The set of possible action identifiers is $\mathcal{A} = [[0, \cdots, |\mathbf{W}|]]$, where $|\mathbf{W}|$ is the number of neurons. Each neuron is associated to a prototype vector, which is the action's parameters.
  - $\texttt{Actions}_p : \mathcal{A} \to \mathbb{A}_l$ is the function that returns the action's parameters for a learning agent l from an action discrete identifier in profile p.
- Q-Table: the 3-dimensional table that learns and memorizes the interests of each (discrete) action identifier in each (discrete) state identifier. The interests are themselves a vector, indexed by the moral values.
  - $\texttt{Q}_p : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^m$ is the function that returns these interests for an action a in profile p, in a state s, where m is the number of moral values.
- ρ : the weights used to scalarize rewards and interests to determine whether a perturbed action is interesting, a vector $\in \mathbb{R}^m$.

We note that, contrary to existing approaches that use scalarization at some point, our exploration profiles are only used to explore, and never to choose an action. This is a crucial difference; we want to use the human users preferences to select actions. The exploration profiles are combined during the deployment phase so that agents have at their disposal all actions learned in the different sub-zones of the action space. We note an exploration profile's vector of weights as ρ; the formula that determines whether a perturbed action is interesting thus becomes:

$$\mathbf{r_t} \cdot \rho + \gamma \max_{j'} \left(\rho \cdot \texttt{Q}(s_{t+1}, j')\right) \overset{?}{>} \rho \cdot \texttt{Q}(s_t, j) \tag{6.1}$$

where $\cdot$ denotes the dot product between 2 vectors, i.e., $\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$.

Example 6.1 (Exploration profiles weights)**.** Let us consider 4 moral values, as described in the Smart Grid use-case in Section 3.4.5 and in the experiments of the previous chapter

in Section 5.4. We propose 5 different exploration profiles: 4 of them each focus on a different moral value, while the last one focuses on learning actions that yield interests "good on average". The generalist exploration profile simply is set to $[\frac{1}{m}, \frac{1}{m}, \frac{1}{m}, \frac{1}{m}]$, where $m = 4$ is the number of moral values. Thus, this profile considers all moral values equivalently. For the other, specialized profiles, we propose to use a weight of 0.9 for their specific moral value, and a weight of $\frac{0.1}{m-1} = 0.033$ for all other objectives. For example, let us consider the previously learned interests $\mathtt{Q}(s, j) = [0.8, 0.3, 0.3, 0.3]$, and the received reward for the perturbed action $\mathbf{r} = [0.8, 0.4, 0.4, 0.4]$. To simplify, let us ignore the $\max_{j'}$ part of the equation, and instead focus on comparing the reward with the previously learned interests. The perturbed action thus did not manage to improve the interest on the first moral value, but did improve on the other objectives. Had we used a weight of 0, we would ignore this improvement and determine that the perturbed action is not interesting, which would be counter-intuitive and counter-productive. We use a very low weight on these other objectives, so that an eventual decrease on the targeted moral value cannot be compensated by an improvement on the other dimensions. Other exploration profiles, such as $[0.75, 0.25, 0, 0]$ could also be used, and we detail in Section 6.8 perspectives on this topic.

Note that, in addition, the Bellman equation must be adapted as well, where k is used to iterate on the various moral values:

$$\forall k \in [[1, m]] : \mathtt{Q}_{t+1}(s_t, a_t, k) \leftarrow \alpha \left[r_{t,k} + \gamma \max_{a', \rho} \mathtt{Q}_t(s_{t+1}, a', k) \right] + (1 - \alpha) \mathtt{Q}_t(s_t, a_t, k) \quad (6.2)$$

The Q-Value was previously a scalar, updated by adding the reward, which was also a scalar; they are know both vectors. We adapt the Equation (2.1) presented earlier by simply using element-wise addition of vectors. In other words, the first dimension of the Q-Value is updated by taking into account the first dimension of the reward, and so on. We also need to obtain the interest of the next state, which was computed as the maximum Q-Value of any action in the next state s_{t+1}. As we previously mentioned when adapting the "interesting criterion", the max operator is not defined when comparing vectors: we thus propose to use the interests of the action that maximizes the dot product with the exploration weights ρ.

6.3 Identifying dilemmas

Once the interesting actions have been learned, they can be leveraged to identify dilemma situations when the agent is deployed. First, explorations profiles are merged into agents, and "frozen", i.e., their data structures, especially actions, are not learned any more. When deployed, the learning agents will, at each step, compare the actions proposed by each exploration profile for the current situation, represented by the received observations from the environment. To choose the action to execute, the agent first needs to determine whether there is a dilemma. If there is, it cannot directly choose an action, and must rely on the human user's preferences; otherwise, there is no dilemma, thus the best action can be clearly defined, and the agent simply selects this action.

We pose a few definitions to formalize the dilemma identification process. To better explain our reasoning behind the algorithm we propose, we start from a naïve application of an existing definition of dilemma, and point out the problems that arise and how we overcame them. Let us recall that a situation is described by a vector of observations $\mathbf{o} \in \mathbb{O}$, according to the DecPOMDP framework described in Definition 4.1.

We start by adapting a definition of *dilemma* proposed by Bonnemains (2019):

> A situation is considered as a dilemma if there is at least two possible decisions, and every decision is unsatisfactory either by nature or by the consequences.

We refer the interested reader to Bonnemains' thesis for the formal definition. However, Bonnemains used a symbolic model, whereas our actions are defined in terms of continuous interests and parameters. Thus, we adapt it to better fit our conditions:

Definition 6.2 (Naïve dilemma)**.** A situation is considered as a dilemma if, among the proposed actions, all actions are unsatisfactory. An action is unsatisfactory if there is another action with a higher interest on at least one dimension.

This definition takes into account the fact that actions have continuous interests, and we always have a choice between at least 2 actions, provided that the number of neurons in the Action-(D)SOM is greater than or equal to 2, or that there are at least 2 exploration profiles. It echoes the notion of *regret* also mentioned by Bonnemains. For example, considering an action a_1 with interests $\mathtt{Q}(s, a_1) = [0.5, 0.5]$ and another action a_2 with interests $\mathtt{Q}(s, a_2) = [1, 0]$, we can say that taking action a_1 would result in a regret with respect to the first moral value, as a_2 would have yielded a better result. Conversely,

taking a_2 would result in a regret with respect to the second moral value, as a_1 would have yielded a better result. Thus, for each of these actions, there is "another action with a higher interest on at least one dimension": if these are the only possible actions, the situation is a dilemma.

To formalize this definition, we note that it is in line with the notion of Pareto Front (PF), which is a well-known tool in multi-objective reinforcement learning. We first define the Pareto-dominance operator:

$$\mathbf{x} >_{Pareto} \mathbf{y} \Leftrightarrow (\forall i : x_i \geq y_i) \text{ and } (\exists j : x_j > y_j) \tag{6.3}$$

In other words, a vector Pareto-dominates another if all its dimensions are at least equal, and there is at least one dimension on which the vector has a strictly superior value. Applying it to our problem of identifying dilemmas, we can compute the Pareto Front (PF), which is the set of all actions which are not Pareto-dominated by another action:

$$\texttt{PF}(\mathbf{o}) = \{(p,a) \in (\mathcal{P},\mathcal{A}) \mid \nexists (p',a') \in (\mathcal{P},\mathcal{A})\, \texttt{Q}_{p'}(\texttt{States}_{p'}(\mathbf{o}),a') >_{Pareto} \texttt{Q}_p(\texttt{States}_p(\mathbf{o}),a)\} \tag{6.4}$$

Note that we compare *all* actions from *all* profiles together to compute this Pareto Front (PF). An action is unsatisfactory if there is another action with higher interests on at least one moral value: in other words, if it does not dominate all other actions. Actions not in the PF are dominated by those in the PF and are unsatisfactory. However, actions in the Pareto Front cannot dominate each other, by definition. Thus, if the PF contains more than 1 action, it means that, for each action, there exists in the PF another action with a higher interest on at least one dimension, and thus the situation is a dilemma.

However, first experiments have demonstrated that this definition was not well-suited, because there was not a single situation with only 1 action in the PF. This is due to 3 reasons: 1) As the interests are continuous, it is possible to have an action which is non-dominated by only a small margin, e.g., compare $[0.9, 0.9, 0.9, 0.9]$ to $[0.900001, 0, 0, 0]$. Clearly the first action seems better, nonetheless, because of the small difference on the first dimension, it does not dominate the second action. 2) We explore from different sub-zones of the action space, and we combine different exploration profiles which all search for different interests, e.g., one exploration profile tries to obtain actions with high interests on the first dimension, whereas another profile tries to obtain actions with high interests on the second dimension. Thus, it seems natural to obtain actions that cannot dominate each other. 3) We do not impose a limit on the number of moral values: we use 4 in our Smart Grid use-case, but we could theoretically implement dozens. To

dominate another action, an action needs first and foremost to be at least equally high on all dimensions: if we increase the number of dimensions, the probability to find at least one dimension for which this is not the case will increase as well, thus preventing the second action to be dominated, and adding it to the PF.

Thus, the first and rather naive approach does not work in our case; we extend and improve this approach, by keeping the ideas of Pareto-dominating and Pareto Front, but adding new definitions and data structures, namely *ethical thresholds* and *theoretical interests*. First, we build upon one of our general assumptions with respect to the Socio-Technical System: ethics comes from humans. Thus, to determine whether an action is acceptable, from an ethical point of view, we choose to rely on the "source of truth" for ethics, i.e., the human users, and we introduce an *ethical threshold*, reflecting their judgment upon the point at which an action satisfies a given moral objective or value. The ethical threshold is set by a human user for each learning agent, and is used as a target for the actions' interests. Moreover, we note that learned actions' interests may be different between the various moral values. This is partially due to the fact that some moral values may be harder to learn than others, or because of the way the reward function (or judgment) for this specific value is designed. Human users may have different requirements for these moral values, for example someone who is not an ecologist might accept actions with a lower interest for the environmental sustainability value than the inclusiveness value. For these 2 reasons, we choose to have a multi-objective ethical threshold: in other words, the thresholds for each moral value may differ.

Definition 6.3 (Ethical threshold)**.** An *ethical threshold* is a vector $\zeta \in \mathbb{Z} = [0, 1]^m$, where m is the number of moral values. Each component ζ_i, $\forall i \in [[1, m]]$, can be read as a threshold between 0% and 100%, relative to the interest associated with moral value i.

Note that we define the ethical thresholds as values between 0 and 1. This is indeed, we argue, rather intuitive for the human users, and easy to understand: 0 represents an action completely uninteresting, whereas 1 represents an action perfectly interesting. However, this poses a problem with the actual actions' interests: they are updated at each time step using the modified Bellman equation (6.2), which leaves them unbounded. Interests depend both on the action's correctness, and the number of times they have been explored. In other words, an action with a low interest could be explained either by its non-compliance with the moral values, or because it was not often explored. For example, running a simulation for $5,000$ steps could yield an interest of 6, whereas running the simulation for $10,000$ steps could yield an interest of 11. These absolute interests are

not comparable, especially by lay users, as there is no reference: what does 6 mean? Is it good? We thus propose to introduce an anchor, or point of comparison, in the form of a *theoretical interest*. The theoretical interests are computed using the same Bellman equation as the interests, with a small difference: we assume the received reward was the maximum possible, as if the action was "perfect".

Theoretical interests are updated at the same time as the interests, and thus grow similarly. If the action is poorly judged, its interest will be lower than the theoretical interests; if the action is judged as adequate, its interests will converge close to the theoretical ones. The number of steps impacts interests and theoretical interests exactly in the same manner: thus, the ratio between the two can be considered time-independent, e.g., if we train actions for only $5,000$ steps and get an effective interest of 6, the theoretical interests will reflect the fact that the maximum will be near 7. Thus, an action with a ratio of $\frac{6}{7}$ can be considered as quite good. On the other hand, if we train actions for $10,000$ steps but still get an effective interest of 6, the maximum indicated by theoretical interests will be near 11, and an action with a ratio of $\frac{6}{11}$ will be considered as less satisfactory. We therefore offer a reference to compare unambiguously actions' interests that does not depend on the number of steps. To compute and memorize the theoretical interests, we introduce a new data structure to the agents' exploration profiles in the bootstrap phase, which we name the *Q-theoretical table*. As its name indicates, it is very similar to the Q-Table for interests; the only difference is the update equation, as we mentioned earlier. In the sequel, we assume the maximum reward to be 1, the equation for updating theoretical interests is thus:

$$\mathtt{Q}_{t+1}^{theory}(s_t, a_t) \leftarrow \alpha \left[1 + \gamma \max_{a'} \mathtt{Q}_t^{theory}(s_{t+1}, a')\right] + (1-\alpha)\mathtt{Q}_t^{theory}(s_t, a_t) \tag{6.5}$$

Note that, to simplify, we did not consider the multi-objective aspect in this equation. The actual formula adds a third dimension to the Q-theoretical table, but the update formula stays the same, as we use 1 as the "theoretical reward", regardless of the moral value.

Now that we have the theoretical interests, and the ethical thresholds, we may define what is an *acceptable action*.

Definition 6.4 (Acceptable action)**.** An action $(p, a) \in (\mathcal{P}, \mathcal{A})$, where p is an exploration profile and a an action identifier, is deemed *acceptable* if its interests, compared to the theoretical interests, in a given situation represented by the observations $\mathbf{o}$, attain the ethical thresholds on all moral values. Formally, (p, a) is acceptable if and only if

$\forall i \in [[1, m]] : \frac{\mathtt{Q}_p(\mathtt{States}_p(\mathbf{o}),a)_i}{\mathtt{Q}_p^{theory}(\mathtt{States}_p(\mathbf{o}),a)_i} \geq \zeta_i$. In this formula, $\mathtt{Q}_p$, $\mathtt{States}_p$ are the functions from exploration profiles given in Definition 6.1; $\mathtt{Q}_p^{theory}$ values are computed through Equation (6.5); ζ are the ethical thresholds as specified in Definition 6.3.

We see that acceptable actions depend on the user-specified ethical thresholds; additionnally, as the theoretical interests are by construction superior or equal to the interests, the ratio is a value $\in [0, 1]$ that can be easily compared with the thresholds. Thus, an ethical threshold of $\zeta = [0.8, 0.75]$ might be read as: "An action is acceptable if its interest with respect to the first moral value is at least 80% of the maximum attainable, and its interest for the second moral value at least 75%".

From this, we can finally define a *dilemma*.

Definition 6.5 (Dilemma)**.** A situation is said to be in a dilemma if none of the actions in the Pareto Front is *acceptable* with respect to a given ethical threshold. More formally, we define a *dilemma* as a tuple $(\mathbf{o}, \zeta, optimal) \in \left(\mathbb{O}, \mathbb{Z}, 2^{\mathcal{P}\times\mathcal{A}}\right)$, where $\mathbf{o}$ is the observation vector representing the situation, ζ is the user-given ethical threshold, and $optimal$ is the Pareto Front of actions for the given situation, such that $optimal = \mathtt{PF}(\mathbf{o})$ as defined in Equation (6.4).

This formal definition of *dilemmas* can be computed and used automatically by learning agents, while relying on human users' preferences for the ethical threshold. An advantage is to offer configurability: some might consider that most actions are acceptable, thus letting agents choose in their place most of the time, whereas others might specify a higher threshold, thus identifying more situations as dilemmas and forcing agents to ask for their preferences concerning which action to take. As the ethical thresholds are specified individually by user, several agents with various behaviours may coexist in the system.

6.4 Learning user preferences

Once we know how to identify dilemmas, the next and final step mentioned in Section 6.1 is to settle them, i.e., to choose an action. This action cannot be the best, otherwise we would not be in a dilemma, and it reflects some trade-off between several, conflicting moral values. We believe and defend that these trade-offs must be settled by human

users, and we thus ask for their preferences. However, asking for preferences adds mental charge to the humans: it is not a trivial task. If we ask too often, it might become a burden, and the system becomes unusable: one of the goals of implementing a system of artificial agents is to automate some of our tasks, so as to relieve us; if the system asks us for the correct decision at each step, this completely negates the benefits.

To avoid this, we want to learn the human preferences, so that artificial agents will solicit humans less often, while still exhibiting a behaviour that corresponds to the human user's preferences. Learning the correct preferences for dilemmas could be as simple as maintaining a map of dilemmas to preferences; yet, we recall that we are in continuous domains. In particular, the situation in which a dilemma occurs is represented by an observation vector, which is composed of continuous values. Thus, even a small difference in only one of the observation's components would yield, strictly speaking, a different dilemma. This seems counter-intuitive: surely, not all dilemmas are unique? We may consider that some conflicts in various situations correspond to similar dilemmas. For instance, a choice of consumption at 2:00 AM or 2:01 AM can be considered close, and settled equivalently. Yet, a choice between the same actions, with the same conflicts between moral values, may constitute a different dilemma when occuring at 7:00 AM, i.e., the choice may be different. To reduce the burden on human users, agents could "group" dilemmas that are close; we thus need a way of grouping them. To do so, we propose the notion of *context*, which we vaguely define, for now, as such: "A context is a group of dilemmas identified as being similar, such that they can be settled in the same manner, i.e., with the same action selection".

We now need to propose a formal definition for these contexts, such that artificial agents can automatically identify them and group dilemmas by contexts. This is what we first describe in this section. Then, we explain how the unknown dilemmas are presented to human users to create new contexts when asking for their preferences, and how the preferences are learned. As previously, we start with a naïve definition to highlight its issues, and explain the reasoning behind the actual definition.

First, we can notice that contexts are somewhat to dilemmas what discretized states are to observations. A simple and intuitive idea can be to leverage the notion of states to define a context: if dilemmas appear in the same state, they may belong to the same context. However, agents now use several exploration profiles when deployed, and each of the profiles has its own State-(D)SOM: in other words, each exploration profile discretizes observations into states differently, even though they receive the same observations vectors. For example, profile p_1 might say that the current situation corresponds to state

s_3, whereas profile p_2 might say it corresponds to s_1. As we combine proposed actions from all exploration profiles, and proposed actions are determined from the Q-Table, based on the discrete state, we need to consider the states discretized by *all* exploration profiles, thus leading to a combination of states. This combination of states effectively describe the current situation, for each exploration profile. An unambiguous combination could be, e.g., $[s_3, s_1, s_{12}, s_{44}, s_5]$, where s_3 is the discrete state from profile p_1, s_1 from profile p_2, etc., with a total of 5 profiles. Under this definition, another combination $[s_3, s_1, s_{12}, s_{44}, s_{22}]$ would be a different context, because one of the exploration profiles deemed the situation as a different state, even though all other states are the same. Note that, if the discrete states are exactly the same for all exploration profiles, then we are guaranteed to have the same proposed actions, and thus the same Pareto Front of optimal actions. Two dilemmas that have exactly the same combination of states would have the same proposed actions, which makes it easier to "settle them in the same manner", as our vague definition of context puts it. Regardless of the action selection learned as a preference for this context, we are guaranteed to find the same action in both dilemmas.

More formally, this definition of context can be represented as $\texttt{Context}(\mathbf{o}) = \langle \forall p \in \mathcal{P} \,|\, \texttt{States}_p(\mathbf{o}) \rangle$. The 2 most important advantages are its simplicity to compute, and the guarantee that dilemmas in a same context have the same actions, as we mentioned. However, earlier experiments have demonstrated a few flaws to this simple definition. Indeed, with 5 exploration profiles, and thus lists of 5 elements, the probability that at least one of these elements differ is quite high, and increases with the number of possible discrete states, i.e., the number of neurons in the State-(D)SOM. In one of our experiments, over $10,000$ steps and using 144 neurons in the State-SOM, we obtained $1,826$ unique lists: there is still a $5\times$ decrease factor, which is interesting for such a simple definition, but it is not enough. Asking the users $1,826$ times in a simulation seems way too much in our opinion. The main problem is that the vast majority of lists appear only a few times: 55% of time appear exactly 1 time, only 45% appear more than 2 times, 30% more than 3 times, 10% more than 10 times, etc.

Another attempt tried to leverage the AING (Bouguelia, Belaïd, & Belaïd, 2013) algorithm to automatically group dilemmas through distance between situations they occur in, in the observation space. AING is a clustering algorithm, based on the Growing Neural Gas idea (Fritzke, 1995): each time step, a data point, i.e., a dilemma in our case, is presented to the neural gas. The algorithm compares the point with the existing neurons: if it is sufficiently close to one of the neurons, according to an automatically determined

threshold, the data point is associated with this neuron, and the neuron is updated slightly towards the data point. If no neuron is sufficiently close, a new one is created at the exact position of the data point: this mechanism represents the creation of a new context in our case, when existing ones do not suffice to describe the current situation. However, it failed to work, perhaps because dilemmas appear in a seemingly random order: we may have first a dilemma in the bottom-left quadrant, and then another in the upper-right quadrant, and so on. Thus, the algorithm creates lots of neurons because all these dilemmas seem so far away from each other. When finally a dilemma appears that seem close to another one, there are so many neurons around that the algorithm computes an infeasible distance threshold, and this dilemma is therefore also assigned to a new neuron. We could have tweaked the distance formula to force the creation of fewer neurons, however, we feared that this may artificially create a topology that does not initially exist, and thus making our agents learn "garbage" contexts. Another disadvantage is that neurons use a single distance threshold, relative to their center, which makes a sphere of attraction around their prototype. However, we have no certainty that the bounds between any two different contexts can be represented by such spheres. Perhaps, in some cases, we might have an abrupt bound, on a specific dimension, because at this specific point, there is a clear change of contexts for human users. In other words, the same distance traveled does not have the same significance on two different dimensions. A one-minute shift between 2:00 and 2:01 may not change the context, but a 1% shift in comfort between 49% and 50% could change the context, as seen by the user.

Reflecting on this, we thought that contexts exist in the human eye. We thus propose to leverage human users to identify contexts, instead of a purely automated tool. An additional advantage of this approach is that humans can bring their own baggage, and thus their own mental state, to the identification of contexts, which the artificial agents do not have access to. More specifically, we define a *context* as a set of bounds, i.e., minimal and maximal values, for each dimension of the observation vector. This is a quite simple definition that still allows for arbitrary bounds around the context.

An interface is created to allow human users to choose the bounds for each non-recognized dilemma, thus creating a new context. When a dilemma is identified by a learning agent, the dilemma's situation is compared to the bounds of known contexts: if the situation is entirely inside the bounds of a given context, we say that this context "recognizes" the dilemma, and we apply the action that was learned for this context. In other words, the dilemma belongs to the context.

Definition 6.6 (Context)**.** A *context* is a set of bounds, both minimal and maximal, for each dimension of the observation space $\mathbb{O}_l$. Formally, a context is defined as $c \in \mathcal{C} = \langle (b_1, B_1), \cdots, (b_g, B_g) \rangle$, where g is the number of dimensions of the observation space ($\mathbb{O}_l \subseteq \mathbb{R}^g$), b_k is the lower bound for dimension k, and B_k is the upper bound for dimension k. A context *recognizes* a situation in a dilemma, represented by its observation vector $\mathbf{o_l}$ if and only if $\forall k \in [[1, g]] : b_k \leq o_{l,k} \leq B_k$.

Now that we can define contexts, we can learn the users' preferences in each context about which action should be taken in dilemmas that are recognized by this context. We also propose to filter actions by comparing their parameters and removing those with similar parameters, as they would have a similar effect on the environment. This allows reducing the number of actions proposed to the user. Indeed, because of the way we combine exploration profiles and the use of Pareto-domination, we end up with many proposed actions that do not dominate each other. When simply merging the profiles together, and taking the Pareto Front of all proposed actions, we obtained most of the time between 4 and 16 optimal actions in the PF. Although it might not be difficult to choose an action among 4, the cognitive load seems too high when we have to compare 16 actions. For each pair of actions among the PF, if, for every dimension, their parameters differ by less than a given threshold, we consider the actions to be equivalent, and we only retain one of the two. This threshold can be an absolute or relative value, and we detail in our experiments and results the choice of this threshold, and the resulting number of actions. To give an order of magnitude, we managed to reduce from maximum 16-20 actions to about 4-6 actions each step, while using what seemed to us reasonable thresholds.

Remark. Let us emphasize that we only compare actions on their parameters. Indeed, actions are sometimes similar on their interests as well; however, we argue it would not be a good idea to remove actions that have similar interests, if they have different parameters. This is precisely what we want to avoid, by giving the control back to the user and asking them their preferences: to make the trade-offs explicit. Using the interests is a great pre-filtering tool: if an action is clearly better than another, we do not want to keep the second one, as the first one was, in all likelihood, judged better with respect to the moral values, and thus would have a better impact. This is why we use the interests to compute the Pareto Front. Once we have the actions that cannot be compared to each other, because none of them dominates any other, i.e., the Pareto Front, we need to focus on the parameters instead. If an action proposes to consume 600Wh while another proposes to consume 590Wh, the difference is perhaps not that important, and we can

remove one of them. On the contrary, if two actions have almost equal interests, but different parameters, this means there is a compromise to be made: although they would have a comparable impact in terms of rewards, by their very nature they are different.

With the number of proposed actions reduced, and thus more manageable for lay users, we present these alternatives to the users when a dilemma occurs, and they accordingly define a new context. The actions are compared both on their interests and their parameters, so that users may make an informed decision. The association between contexts and chosen action, i.e., user's preferences, is simply memorized by an associative table.

6.5 Summarizing the algorithm

In this section, we summarize the 3 previous steps, which are formally described in 2 algorithms, one for the bootstrap phase, and another for the deployment phase:

1. Bootstrap phase (Algorithm 6.1): learning interesting actions
2. Deployment phase (Algorithm 6.2)
    1. Identifying dilemmas in situations
    2. Learning contextualized user preferences

6.5.1 Bootstrap phase

We recall that, in the first phase, we introduce new exploration profiles: each agent learns a separate profile. Each exploration profiles contains a State-(D)SOM, an Action-(D)SOM, a Q-Table, and a new Q-theoretical table, which they learn throughout the time steps by receiving observations, selecting actions to explore them, and receiving rewards. Algorithm 6.1 describes the process that takes place during the bootstrap phase, although in a pseudo-algorithm manner, where some functions such as the random perturbation (noise) over actions are left out or the update of (D)SOMs, as they have been previously described in Chapter 4.

The bootstrap phase is set for a fixed number T of steps (line 1). At each time step t, learning agents receive observations (line 4), and determine the state hypothesis as in the Q-SOM and Q-DSOM algorithms (line 5), by leveraging the State-SOM and finding

the Best Matching Unit. They select actions based on the number of times they were already chosen (lines 6 and 7), and noise the action parameters to explore the action space (line 8). The selected action is then executed in the environment (line 9). Learning agents receive the new observations and the reward (lines 12 and 13), and use them to update their data structures (State-(D)SOM, Action-(D)SOM, Q-Tables), similarly to the Q-SOM and Q-DSOM algorithms (lines 14-21). However, this updated algorithm differs on 2 aspects:

1) The perturbed action is deemed interesting or not based on the exploration profile's weights and Equation (6.1) (line 16).

2) In addition to the Q-Values, which are updated using Equation (6.2), the Q-theoretical values are updated thanks to Equation (6.5) (line 20).

Algorithm 6.1: Learning process during the bootstrap phase

Function `learning()`:
  Data:
    $\mathcal{L}$ the set of learning agents
    T the number of time steps
    $N_{l,s,j}$ the number of times action j was selected in a state s
    ρ the agent's exploration profile $\mathcal{U}, \mathcal{W}$ State-(D)SOM and Action-(D)SOM neurons
    $\mathbf{U_i}, \mathbf{W_j}$ vector associated to neuron i (resp. j) of the State-(D)SOM (resp. Action-(D)SOM)
  for $t = 1$ *to* T **do**
    /* All agents choose an action to explore */
    for $l \in \mathcal{L}$ **do**
      $\mathbf{o_{l,t}} \leftarrow \texttt{observe}(env, t, l)$
      /* Discretize state */
      $s_t \leftarrow \text{argmin}_i \, ||\mathbf{o_{l,t}} - \mathbf{U_i}||$
      /* Choose action based on number of times enacted instead of interests */
      $j \leftarrow \texttt{choose}(\mathbf{N_{l,s}})$
      $N_{l,s,j} \leftarrow N_{l,s,j} + 1$
      /* Random noise to explore the action space */
      $\mathbf{a_{l,t}} \leftarrow \texttt{noise}(\mathbf{W_j})$
      $\texttt{execute}(env, \mathbf{a_{l,t}})$
    end
    /* Agents learn and update their data structures */
    for $l \in \mathcal{L}$ **do**
      $\mathbf{o_{l,t+1}} \leftarrow \texttt{observe}(env, t+1, l)$
      $\mathbf{r_{l,t}} \leftarrow \texttt{reward}(env, t+1, l)$
      /* Compute neighborhood of the (D)SOMs */
      $\psi_U \leftarrow \texttt{neighborhood}(\mathcal{U}, s, \mathbf{o_{l,t}})$
      $\psi_W \leftarrow \texttt{neighborhood}(\mathcal{W}, j, \mathbf{a_{l,t}})$
      /* If the noised action is interesting */
      if $\mathbf{r_t} \cdot \rho + \gamma \max_{j'} (\rho \cdot \mathtt{Q}(s', j')) > \rho \cdot \mathtt{Q}(s, j)$ **then**
        Update the Action-(D)SOM using ψ_W
      end
      /* Update Q-table and Q-theoretical */
      $\forall u \in \mathcal{U} \; \forall w \in \mathcal{W} \quad \mathtt{Q}(u, w) \leftarrow$
      $\alpha_Q \psi_U(u) \psi_W(w) \left[r + \gamma \max_{j'} (\mathtt{Q}(i', j')) - \mathtt{Q}(u, w) \right] + \mathtt{Q}(u, w)$
      $\forall u \in \mathcal{U} \; \forall w \in \mathcal{W} \quad \mathtt{Q}^{theory}(u, w) \leftarrow$
      $\alpha_Q \psi_U(u) \psi_W(w) \left[r + \gamma \max_{j'} \left(\mathtt{Q}^{theory}(i', j') \right) - \mathtt{Q}^{theory}(u, w) \right] +$
      $\mathtt{Q}^{theory}(u, w)$
      Update the State-(D)SOM using ψ_U
    end
  end
end

6.5.2 Deployment phase

On the other hand, Algorithm 6.2 describes the process in the deployment phase. Note that the algorithm considers a fixed number of time steps T (line 2), but in practice, nothing prevents from setting $T = \infty$. At each time step, learning agents receive observations (line 3). To select an action, agents compute the Pareto Front (PF) of optimal actions (line 4), as described in Equation (6.4). They then determine whether each of them is an acceptable action, according to Definition 6.4, leveraging the ethical thresholds ζ set by the human user (line 5). If at least 1 acceptable action can be found, the situation is not a dilemma, and the agent automatically selects one of the acceptable actions (lines 7-9). We propose to choose based on the sum of interests per action, so that the selected action has the best impact on the environment (line 8). Otherwise, if no acceptable action can be found, the situation is a dilemma (lines 10-26), and the agent looks for a context that would recognize this dilemma, as defined in Definition 6.6. This means that, for each context, and for each dimension k of the observation space, we check whether the observation $o_{l,t,k}$ is within the bounds $c_{[b_k]}$ and $c_{[B_k]}$ defined by the context (line 13). When such a context is found, the action that was associated to this context by the user is selected. In the case where no context corresponds (lines 18-24), an interface asks the user for which action to be enacted, and the definition of a context in which the same action should be selected again. We filter the proposed actions to remove those which have too similar parameters (lines 20-21): if another action is found with parameters differing by less than 3% on each dimension, the action is removed from the set of proposed actions. For example, $[1, 1, 1, 1, 1, 1]$ and $[0.99, 0.98, 0.99, 0.99, 0.99, 1]$ have less than 3% difference on each dimension, and thus one of them will be removed. Finally, the association of this context to the selected action is memorized (line 23), so that it may be re-used in future time steps. The selected action (whether from an existing or a new context) is executed in the environment (line 25).

Algorithm 6.2: Decision process during the deployment phase

Function `decision()`:
  Data:
    $\mathcal{L}$ the set of learning agents
    T the number of time steps
    $Contexts_l$ the map of contexts to action learned by agent l
  for $t = 1$ *to* T **do**
    /* All agents choose an action */
    for $l \in \mathcal{L}$ **do**
      $\mathbf{o_{l,t}} \leftarrow \texttt{observe}(env, t, l)$
      /* Get optimal actions from the profiles */
      $optimal \leftarrow \texttt{PF}(\mathbf{o_{l,t}})$
      /* Filter eventual acceptable actions */
      $acceptables \leftarrow \left\{(p, a) \in optimal \mid \forall i \in [[1, m]] : \frac{\texttt{Q}_p(\texttt{States}_p(\mathbf{o_{l,t}}),a)_i}{\texttt{Q}_p^{theory}(\texttt{States}_p(\mathbf{o_{l,t}}),a)_i} \geq \zeta_i\right\}$
      if $|acceptables| \geq 1$ **then**
        /* Not a dilemma, let us take the acceptable action with maximum sum of interests */
        $\mathbf{a} \leftarrow \max_{\sum}(acceptables)$
      end
      else
        /* This is a dilemma, try to find an existing suitable context */
        $context \leftarrow null$
        forall $(c, \mathbf{a}) \in Contexts_l$ **do**
          if $\forall k \in [[1, g]] : c_{[b_k]} \leq o_{l,t,k} \leq c_{[B_k]}$ **then**
            /* Context c recognizes the current situation */
            $context \leftarrow c$
            break
          end
        end
        if $context == null$ **then**
          /* No context has been found, ask the user */
          $context \leftarrow \texttt{ask\_bounds}(\mathbf{o_{l,t}})$
          /* Remove actions which have parameters closer than 3% on each dimension */
          $optimal \leftarrow \{(p, a) \in optimal \mid \nexists (p', a') \in optimal :$
            $\forall k \in [[1, d]] \, |\texttt{Actions}_p(a)_k - \texttt{Actions}_{p'}(a')_k| \leq$
            $0.03 \times |\texttt{Actions}_{p'}(a')_k|\}$
          $\mathbf{a} \leftarrow \texttt{ask\_action}(optimal)$
          /* Memorize this couple context and selected action */
          $Contexts_{l,context} \leftarrow \mathbf{a}$
        end
        $\texttt{execute}(env, \mathbf{a})$
      end
    end
  end
end

6.6 Experiments

In this section, we describe the experiments' setup we used to evaluate our contribution on the multi-objective aspect, with a focus on learning interesting actions and identifying dilemmas to, ultimately, learn the contextualized users' preferences in dilemmas. These experiments are split into 2 sets: the bootstrap phase, and the deployment phase. Note that the latter leverages the data structures, i.e., exploration profiles, learned from the former. The implementation details for some mechanisms, such as the action exploration selection during the bootstrap, which were left out in previous sections, are also detailed and compared here.

The implementation was done in the Python 3 language, so that it could be integrated easily with the Q-(D)SOM algorithms. The Graphical User Interface (GUI) used to interact with users was developed using the Tkinter library, and Ttk (Themed Tk) widgets, as it is a simple library for creating GUIs. Tkinter also has the advantages of being part of standard Python, available on most platforms (Unix, macOS, Windows), and capable of easily integrating Matplotlib plots, which are used to present actions' interests graphically.

The rewards came from argumentation-based judging agents (AJAR) as described in Section 5.4.2, using the "min" aggregation and the "simple" judgment function. They were slightly modified to avoid aggregating the reward and instead sending the vector to the RL algorithms.

6.6.1 Learning profiles from bootstrapping

We recall that the objective of the bootstrap phase is to learn interesting actions, in all situations. To do so, we introduced exploration profiles that include a weight vector for scalarizing rewards and interests, so as to focus on different sub-zones of the action space. Agents need to explore the actions so that they can be compared fairly: they do not yet need to exploit at this point, as the bootstrap phase is separated from the deployment phase. Thus, the action selection mechanism has been modified with a focus on the number of times they have been selected, instead of their interests.

Action selection

Initially, we chose to simply select actions randomly: according to the law of large numbers, every action having the same probability of being drawn, they should all be drawn a similar number of times as the number of draws increase. This simple method also avoids memorizing the number of times each action is selected in each state, which saves computational resources, especially memory. However, first experiments have shown empirically that, in some states, the distribution of actions' selection was not uniform. States that were more visited had higher standard deviations of the number of times each action was chosen. This is a problem, as we want the actions' interests to be fairly comparable, so that they can be shown to users later, and they can make an informed choice. Whereas it was part of the exploitation-exploration dilemma in the Q-(D)SOM algorithms, we now want the actions to be fairly compared by having been selected, and thus explored, roughly the same number of times. To reduce the standard deviation of actions' selection within states, and thus achieve fairer comparisons, a first improvement was to replace the random method with a min method, by memorizing $N_{l,s,a}$ the number of times an agent l has chosen action a in state s, and by always choosing the action with the minimum $N_{l,s,a}$.

This method indeed managed to reduce the disparities, within a state. There was still a disparity between states, because some states are more rarely visited than other, but this stems from the environment dynamics, on which we cannot act. Within a given state, all actions were chosen exactly the same number of times, with a margin of ± 1. Indeed, if a state is visited a total number of times that is not a multiple of the number of actions, it is impossible to select every action the same number of times. Yet, by construction, the min method does not allow the difference to be higher than 1: if all actions but one have been selected x times, and the last one $x-1$, min will necessarily choose this one. At the next step, all actions have therefore been selected x times, any of them can be chosen, thus resulting in one action selected $x+1$ times and all other x times. If this state is never visited any more after that, we cannot compensate this difference and end up with a margin of 1 between actions. We computed the standard deviation for each state, and then compared a few statistics: when using the random method, standard deviations ranged from 0.7 to 6.6, with an average of 2.4, whereas, when using the min method, they ranged from 0 to 0.5, with an average of 0.4.

The min method thus offers more theoretical guarantees than the random method, at the expense of higher computational needs. However, there is still a slight bias in this

method, due to an implementation detail of the `argmin` method that returns the action a with the lowest $N_{l,s,a}$: when multiple actions are candidates, i.e., have the same lowest $N_{l,s,a}$, `argmin` returns the first one. Thus, the order of action selection will always be $a_1, a_2, a_3, a_1, a_2, a_3, \cdots$ for 3 actions in a given state. If the state is not visited a number of steps that is a multiple of the number of actions, the first actions will be advantaged compared to the others, e.g., a_1, a_2, a_3, a_1. This is only a small bias, as the difference will only be 1 in the worst case, but still, we can wonder why choosing always the first actions, perhaps other ones would be better. We propose to use the "min+random" method, which consists in taking a random action among those which have the lowest $N_{l,s,a}$ only. It has, again by construction, the same guarantees as the min method, but is not biased towards the first actions.

Agents and exploration profiles

As mentioned in the previous sections, we propose to use a "generalist" profile granting equal importance to all moral values, and several "specialized" profiles that focuses on the compliance with a particular moral value. Concretely, the weights vector for each profile are:

- $\rho_1 = [0.25, 0.25, 0.25, 0.25]$
- $\rho_2 = [0.9, 0.033, 0.033, 0.033]$
- $\rho_3 = [0.033, 0.9, 0.033, 0.033]$
- $\rho_4 = [0.033, 0.033, 0.9, 0.033]$
- $\rho_5 = [0.033, 0.033, 0.033, 0.9]$

Each of these exploration profiles is learned by a separate learning agent. In addition, as the agents' learned actions will be used in a deployment phase, where many agents will impact the same environment, we added several other learning agents, that do not learn exploration profiles but rather try to optimize their behaviour as previously, using the Q-SOM algorithm. Thus, the exploration profiles are learned in a quite realistic scenario.

We ran 1 simulation using the *annual* consumption profile, with our 5 "exploration profile" agents, and 26 other agents (20 Households, 5 Offices, 1 School). At the end of the simulation, the exploration profiles were exported to be reused later, and particularly in the deployed experiments.

6.6.2 Deployment phase

Once the exploration profiles were learned, we created new agents that contained the 5 previously learned exploration profiles, instead of having a State-(D)SOM, an Action-(D)SOM. There are 2 goals for the experiments of this phase:

1. Show a proof-of-concept interface that is usable by human users to learn the contexts and preferences.
2. Demonstrate that agents learn behaviours that are aligned with the given preferences over moral values.

User interaction through a graphical interface

In order to allow users to interact with the learning agents, we created a prototype Graphical User Interface (GUI), which is used when the agent detects a dilemma in an unknown (new) context. We recall that, from the presented algorithm, the agent asks the user for the solution of the dilemma, i.e., which action should be chosen, and the definition of a context in which other dilemmas will be considered similar, i.e., the lower and upper bounds.

The GUI thus presents the current situation, and a set of sliders so that the user can set up the lower and upper bounds, for each dimension of the observation space. This is depicted in Figure 6.2.

Remark. Note that the environment simulator and the learning algorithms manipulate vectors of values $\in [0,1]^g$. This is indeed easier to learn representations: if a dimension shows higher absolute values than others, it might become (falsely) preponderant in the decision process. Having all dimensions in $[0,1]$ mitigates this. However, human users do not have this requirement; on contrary, it is sometimes more difficult to understand what such a value means, e.g., for the time (hour). We understand immediately what 3 o'clock means, but $\frac{3}{24} = 0.125$ is not immediate. Thus, we transformed the value of the "hour" dimension to use a 24-hour format rather than the machine-oriented $[0,1]$ format. Similar transformations could be applied to other dimensions as well, by injecting a priori knowledge from the designers.

The user must select both the context bounds and the action that should be taken when this context is identified, in any order. The bounds' tab is initially shown, but the user

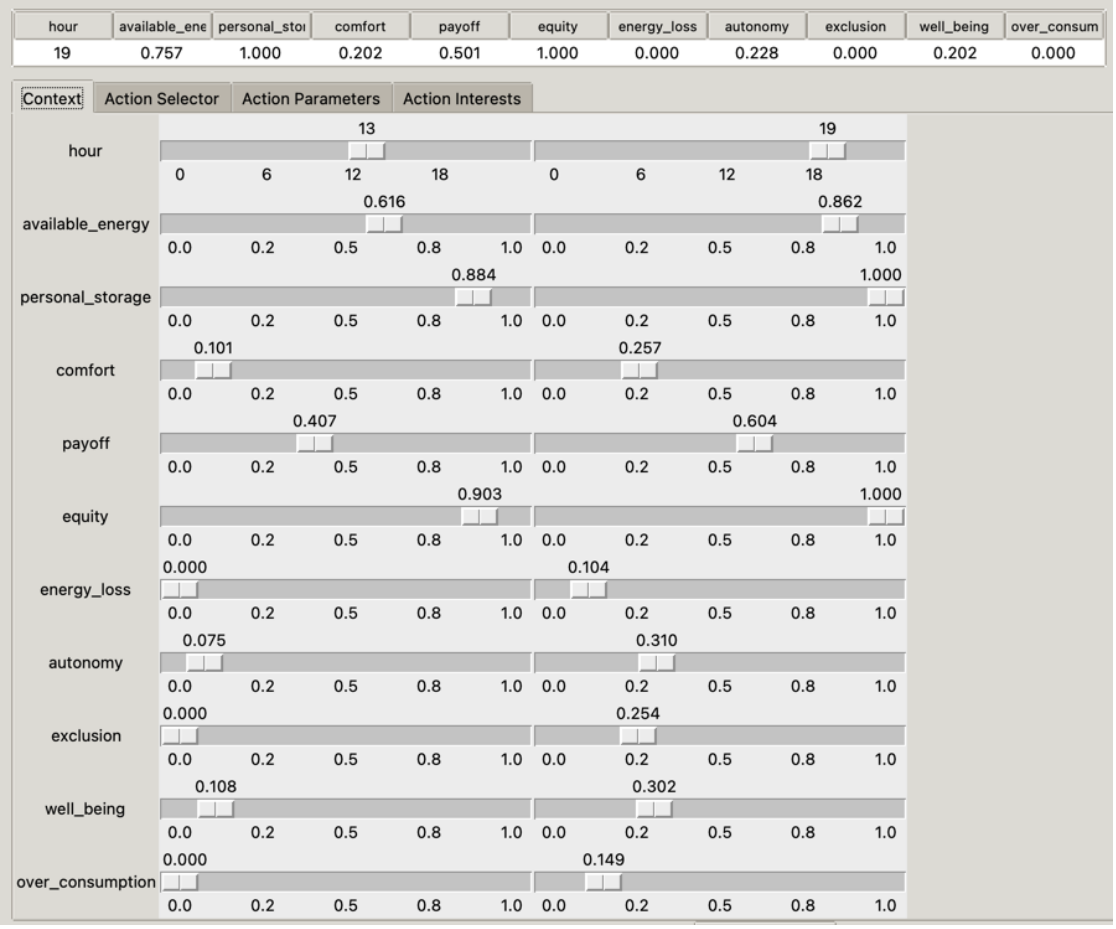


Fig. 6.2.: The graphical interface used to create a new context. The current situation, represented by an observation vector's values, is shown at the top. The bottom part contains sliders that the user must set up to define the desired bounds.

may change the tab freely. We note that it may also be interesting to first ask the user the dilemma's solution before asking for the bounds, as the conflicts between values and actions might guide its ethical reasoning. To do so, the interface presents the different available alternatives, i.e., the optimal actions inside the Pareto Front (PF), after filtering the actions that are too close in parameters. To describe the actions, we choose to plot all actions' respective parameters on the same plot, as several histograms, such that users can compare them quickly. To improve the visualization, we also compute and plot the mean of each parameter for the proposed actions: this allows distinguishing and comparing actions. Figure 6.3 gives an example of these plots. The first two actions, ID = 0 and ID = 1, can be compared, e.g., as follows: "action #0 proposes to consume from the grid twice as energy as the mean of proposed actions; action #1 on the contrary, consumes from the grid less than the average". Similarly, we also plot and compare the actions' respective

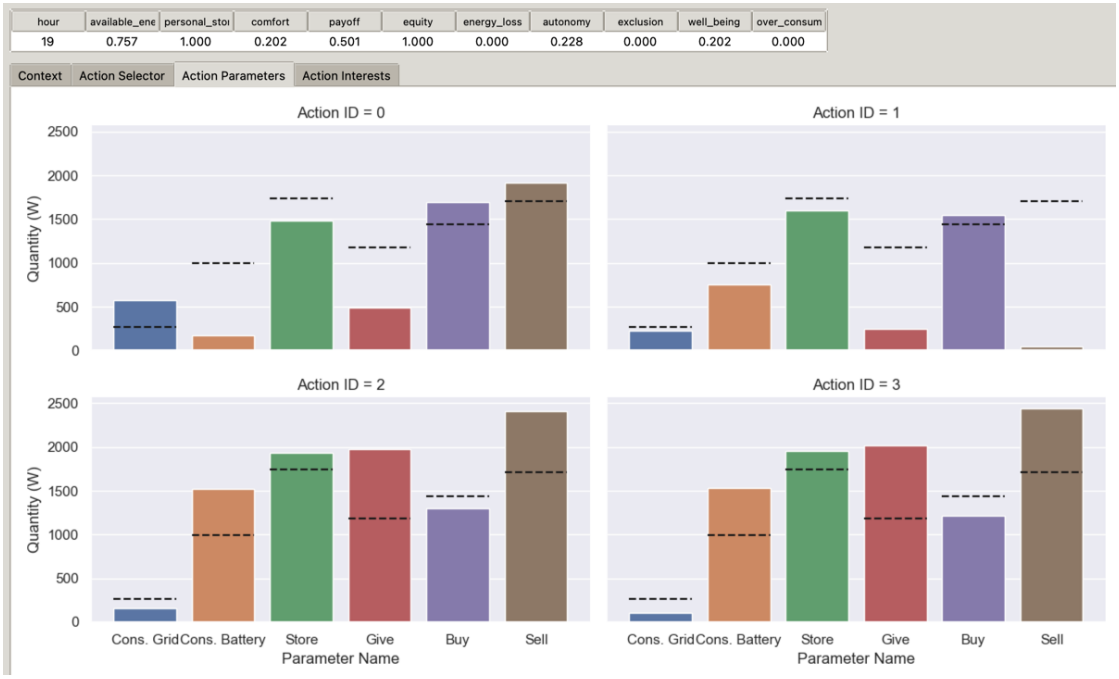


Fig. 6.3.: Interface to compare actions by their respective parameters. The dashed line represents the mean, for each parameter, of the proposed alternatives.

interests, in a separate tab. These interests correspond to the expected satisfaction of each moral value, for every proposed action. As for the parameters, we show the mean on each dimension to facilitate comparison, and, in addition, we also show the theoretical maximum interest. Figure 6.4 shows an example of this interface. As we did not have time nor resources for making an experiment with a panel of lay users, I made a single

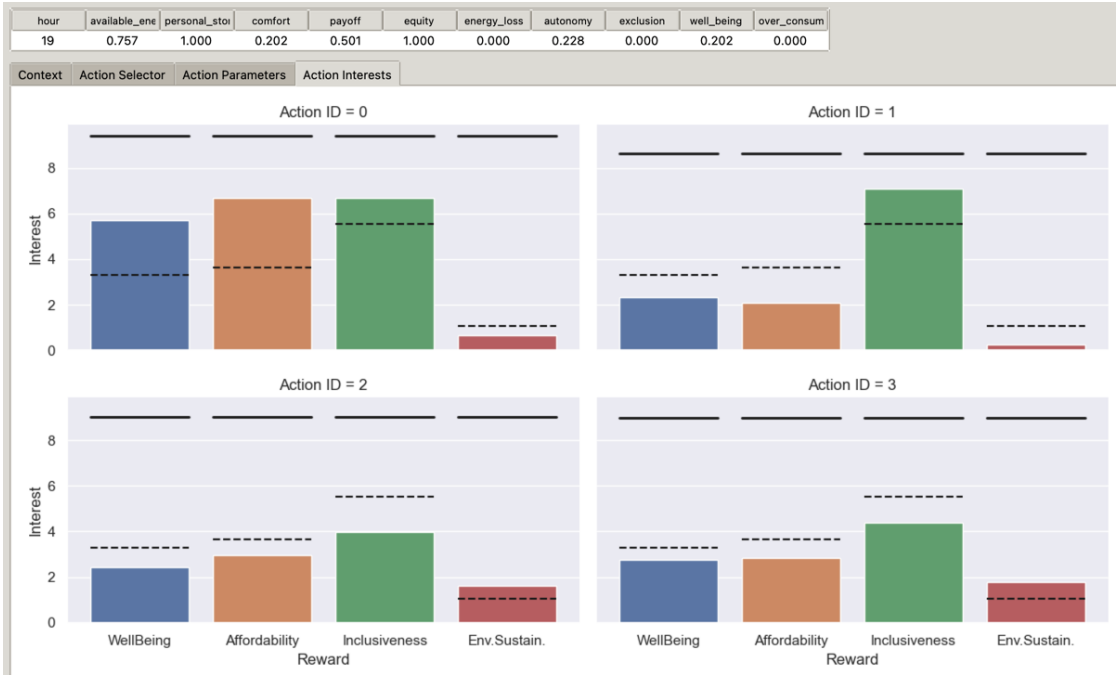


Fig. 6.4.: Interface to compare actions by their respective interests. The dashed line represents the mean, for each interest, of the proposed alternatives. The thick black line represents the theoretical maximum for this specific action on the specific interest.

experiment, placing myself as a user of the system. Thus, the experiment is somewhat limited, with only 1 single agent, and we cannot determine how usable the interface is for non-expert users; however, this experiment still proves the ability to interact with agents, and how agents can learn the contexts. One important aspect is the number of identified contexts, as we do not want the interface to be overwhelming to users. If agents ask their users every 2 steps, the system would quickly be deemed unusable. We expect the agents to quickly identify dilemmas that they cannot settle, as they have initially no knowledge of the preferences, and to reduce the number of required interactions as the time steps increase, because more and more dilemmas will be associated with a known context.

Learning behaviours aligned with contextualized preferences

The second goal of these simulations is to demonstrate that the learned behaviours are aligned with the preferences. For example, if a human user prefers ecology, represented by the *environmental sustainability* value, the agent should favour actions with high interests for this value, when in an identified dilemma. Again, we did not have resources for making experiments with a sufficient number of users. Thus, we propose a proxy experiment, by implementing a few synthetic profiles that are hardcoded to make a decision in a dilemma by using simple rules. We make a proof-of-concept experiment and do not assume that such profiles accurately represent the diversity of human preferences in our society, but we make the hypothesis that, if the system correctly learns behaviours that correspond to these hardcoded preferences, it will learn appropriate behaviours when in interaction with real human users.

We propose the following profiles:

- The “Flexible ecologist” represents humans who want to support the *environmental sustainability* value, as long as their comfort is above a certain threshold. In other words, they filter the actions for which the *well-being* interest is above the threshold, and take the one that maximizes the *environmental sustainability* interest among them. However, if no such action is proposed, then they resort to maximizing their comfort through the *well-being* interest.
- The “Activist ecologist” is the contrary of the “flexible” one. As long as there are actions for which the *environmental sustainability* is above an acceptable threshold, they take the action in this sub-group that maximizes their comfort through the *well-being* interest. When no action satisfies this threshold, they sacrifice their comfort and take the action that maximizes *environmental sustainability*.

- The "Contextualized budget" focuses on the notion of *contextualized* preferences, and makes different choices based on the context, i.e., the current situation. When a dilemma is identified during the day, this profile chooses the action that avoids paying too much, by favouring the *affordability* value. On contrary, when a dilemma is identified during the night, the profile chooses to improve its comfort by maximizing the *well-being* value. This example echoes some contracts where the energy price is time-dependent (peak/off-peak hours), and where the price is usually cheaper at night.

6.7 Results

In this section, we present the results for the experiments described above.

6.7.1 Learning profiles

We first show the learned interests, from all profiles, relatively to their theoretical interests, i.e., the interests actions they propose would have if they received the maximum reward 1 at each time step, on all moral values. The closer an action is to the theoretical interests, the better its impact on the environment, as judged by the reward function. Figure 6.5 shows the ratios of interests over theoretical interests. From this figure, we note two important results. First, actions manage to attain high interests, w.r.t. the theoretical ones, on all moral values, although some can be more difficult to satisfy than others, particularly the *environmental sustainability* in our case. Thus, we demonstrate that the proposed algorithm, during the new *bootstrap* phase, can learn "interesting" actions, which was one of our goals for the experiments. Yet, we see that the satisfaction of moral values are not learned equivalently. The second point is that, in order to select the ethical thresholds, having prior and expert knowledge about the learned interests might be an advantage. Providing human users with this knowledge could help them to choose appropriate ethical thresholds. On the other hand, a potential risk is that it would encourage users to lower their expectations too much, and reduce the desirable to the automatable, by settling for what the machine can do.

As we mentioned in the previous sections, it is important that actions are selected a similar number of times, to ensure a fair comparison, before presenting the alternatives to human

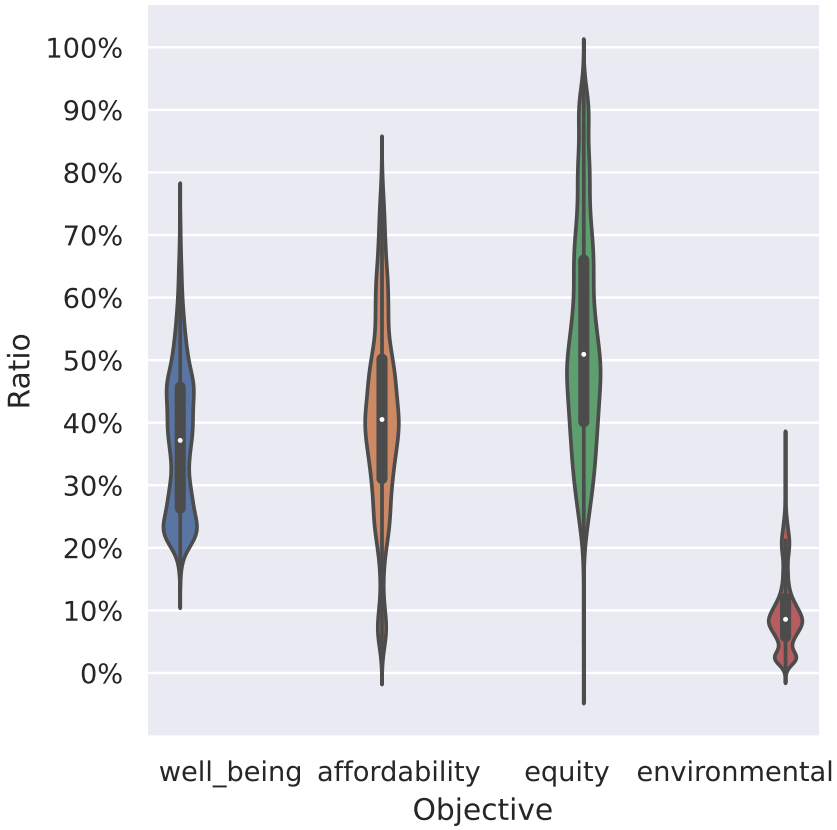


Fig. 6.5.: Ratio between learned interests and theoretical interests, for all actions in all states, from all exploration profiles.

users. Otherwise, we would risk proposing two actions that are in fact incomparable, because one of them has higher uncertainty over its interests. We thus memorized the number of times each action was selected, in each state: Figure 6.6 shows the results. In

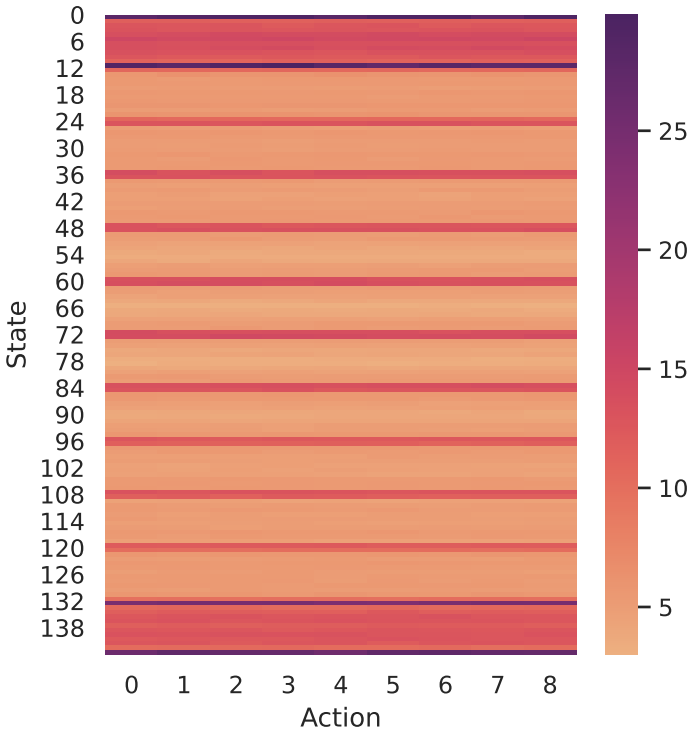


Fig. 6.6.: Number of times each action was selected in each state.

this figure, each state of the State-(D)SOM is represented as a row, whereas each action of the Action-(D)SOM is represented as a column. The color indicates the number of times an action was selected in a given state. We remark from the figure that, although rows have various colors, each line is, individually, roughly uniform. For example, the first row, i.e., state #0, is entirely purple, there is no red, orange, or yellow that would indicate an action less often selected. This demonstrates that our proposed action selection mechanism effectively explores actions similarly, without focusing too much on one specific action within a given state. Thus, we can, when in a dilemma, compare the actions' interests fairly, as we know they were given the same chances.

To verify this mathematically, in addition to visually, we computed the standard deviation of the number of times the actions were enacted, within each state. Figure 6.7 shows

the distribution of the found standard deviations, for every state. As we can see, the vast majority of states have a low deviation, which means actions were explored similarly.

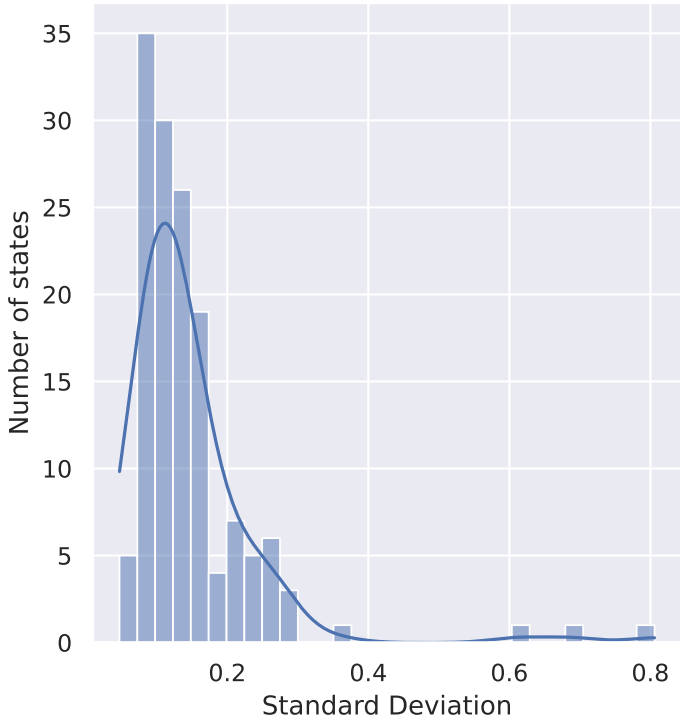


Fig. 6.7.: Distribution of the standard deviations of actions' number of times selected, for each state. The curve represents a kernel density estimation.

6.7.2 Learning human preferences

As mentioned previously, in this experiment I placed myself as the human user of a learning agent, and settled dilemmas that were presented to me through the prototype interface. Our hypothesis was that the agent would, initially, identify many dilemmas, as it had not learned yet my preferences; as the time steps goes by, most identified dilemmas should fall within one or another of the already known contexts, and the number of interactions should decrease. To verify this, we have, during the experiment, memorized each dilemma the agent encountered, and the time steps at which each context was created. The contexts also memorize which dilemmas they "recognize", i.e., the dilemmas that correspond to this context and are automatically settled by the agent. After the

experiment, we computed the total number of identified dilemmas the agent encountered: knowing at which time step a context is created, and how many situations it recognized during the simulation, we can plot the number of "remaining" dilemmas to be settled. Figure 6.8 shows this curve: we can see that, at the beginning of the simulations, we have still $10,000$ dilemmas to settle.

At the first step, a first dilemma is identified: this is natural, as the agent does not know anything about my preferences at this point. This first situation is somewhat of an exception: as the simulation just started, the agent's observations are mostly default values, e.g., 0. For example, the agent did not have a comfort at the previous time step, because there is no previous time step. Thus, this situation is not realistic, and we cannot expect something similar to happen ever again in the simulation. This is why the bounds I have chosen at $t = 0$ defined a context that recognized only a single situation during the whole simulation, i.e., this first exceptional step. At the next time step, $t = 1$, the agent once again asks for my input, and this time the new context covers more dilemmas (about $\sim$ 400); similarly for $t = 2$ (about $\sim$ 800). We can see that, at the beginning, many time steps present new, unknown dilemmas, and thus impose asking the user to choose the proper action and define a new context. This behaviour is in line with our hypothesis. The "interaction time steps" are dense and close together until approximately time step $t = 50$, after which interactions happened only sporadically. On the curve, these interactions are identified by drops in the number of remaining dilemmas, as the new contexts will recognize additional dilemmas.

This supports our hypothesis that the number of interactions would quickly decrease, as the agent is able to settle automatically more and more dilemmas, based on the preferences I gave it. The total number of interactions, in this experiment was 42, which is appreciable, compared to the $10,000$ time steps.

Remark. Note that we have in this experiment as many dilemmas as time steps: this indicates that the ethical thresholds I have set may have been too high, my expectations were not met by the agent, and no acceptable actions could be found. This is not a problem for this experiment, as we wanted to demonstrate that the number of interactions decrease with the time: as every situation was recognized as a dilemma, it was a sort of "worst-case" scenario for me. Had the ethical thresholds been set lower, I would have had fewer dilemmas to settle, and thus, fewer interactions.

On the other hand, this remark also emphasizes a point made previously: lay users might have some difficulties choosing the ethical thresholds, if they have no prior knowledge

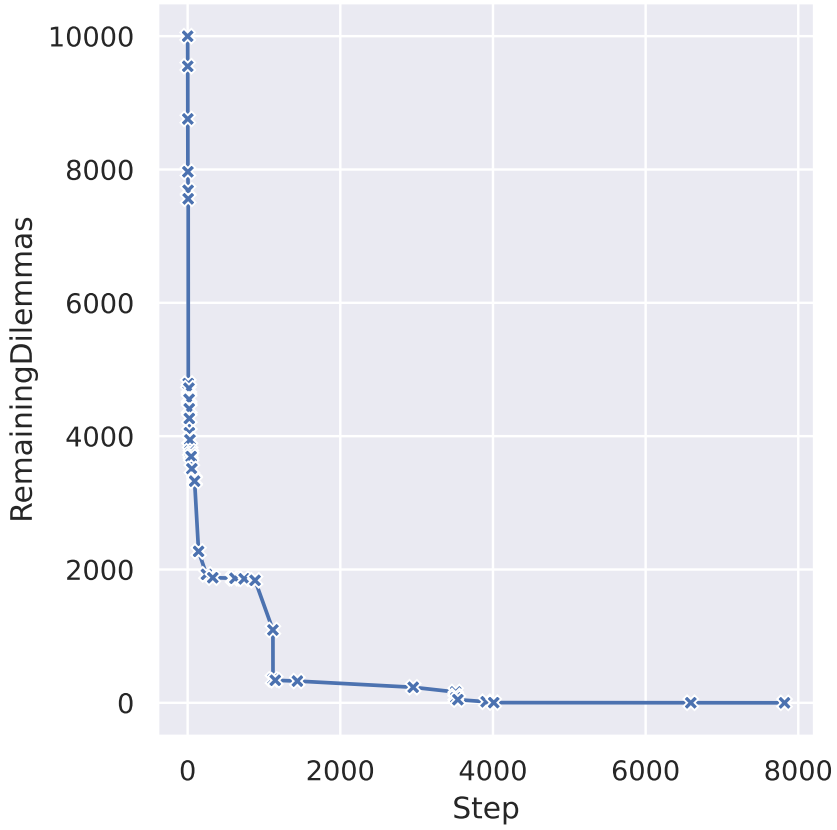


Fig. 6.8.: Number of "remaining" dilemmas to be settled, at each time step of the simulation. When a context is created, we subtract the number of dilemmas this context will identify during the entire simulation. A cross indicates a time step at which a context is created.

about the agents' learning performances. It would be better to correctly explain to users, and to give them information or data so that they can set the thresholds in an informed manner, or perhaps to learn these ethical thresholds from a dataset of dilemma solutions given by users.

A second result is the number of filtered actions proposed in each interaction, which imply an input from the user to select an action and to create a context. We recall that human users have to choose the action when they settle a dilemma; however, the combination of different exploration profiles and the use of a Pareto Front yield many actions, which can be difficult to manage for a lay user. We have proposed to filter actions that are sufficiently close in terms of their parameters, because they represent the same behaviour, in order to reduce the number of actions, and we set the threshold to 3% of relative difference. In other words, if two actions have a difference of 3% or less on every parameter, we remove one of them. Figure 6.9 shows the distributions of the number of proposed actions, in each dilemma, and the number of filtered actions. We can see that, in the original set of *proposed* actions, the number of actions ranged from 8 to 25, whereas the number of *filtered* actions ranges from 2 to 6, which is simpler to compare, as a user.

6.8 Discussion

In this chapter, we proposed an extension to the *Q-SOM* and *Q-DSOM* algorithms that focuses on the multi-objective aspect. By doing so, we make explicit the (eventual) conflicts between moral values, and the trade-offs that ensue, which aims to answer our third research question :

How to learn to address dilemmas in situation, by first identifying them, and then settling them in interaction with human users? How to consider contextualized preferences in various situations of conflicts between multiple moral values?

The objectives, as defined in Chapter 1, were to capture the diversity of moral values and ethical preferences within a society, and especially through multiple human users (O1.1), and to learn behaviours in dilemma situations (O2.2). Dilemmas are situations where no single action is able to optimize all moral values at once; the goal is to make artificial agents able to identify them, and to learn to settle dilemmas according to contextualized human preferences from their respective users.

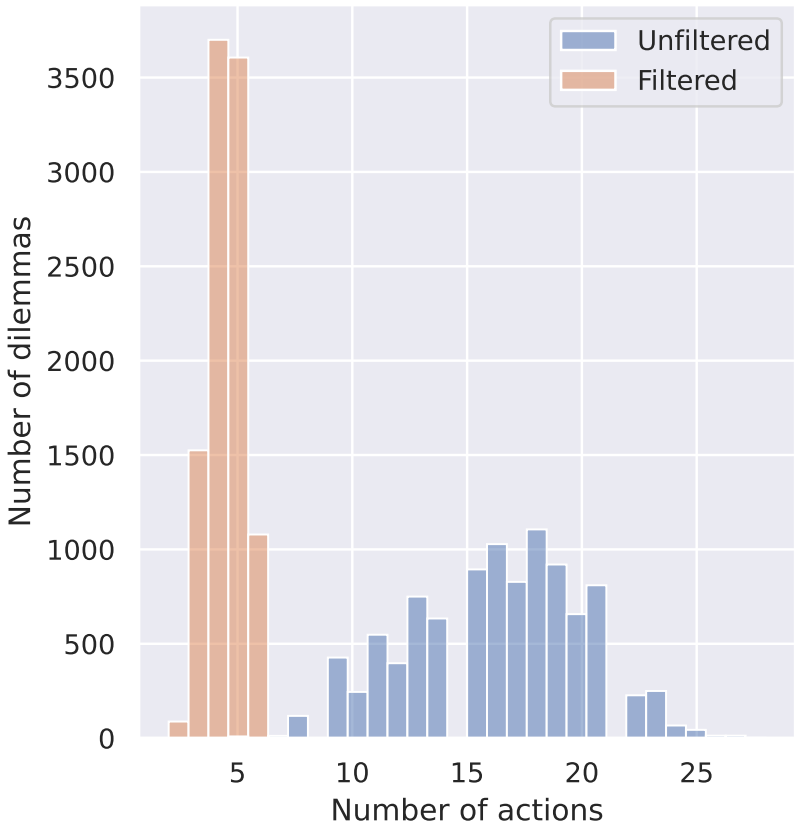


Fig. 6.9.: Distributions of the number of proposed actions in each dilemma of the simulation. The Unfiltered distribution shows the total number, before the filter, whereas the Filtered distribution shows the number of remaining actions, after actions too similar are removed.

O1.1 was attained by considering several human users, and their various contextualized preferences, i.e., preferences are different from a user to another, and from a situation to another. Specifying ethical thresholds for each agent, and thus each user, also improves diversity: an agent could consider most situations as non-dilemmas, whereas another agent would classify the same situations as dilemmas, depending on their users' ethical thresholds. To solve O2.2, we separated the contribution into 3 main parts: 1) learning the "interesting" actions, 2) identifying dilemmas in situation based on interesting actions and ethical thresholds of acceptability, and 3) learning the human preferences. These 3 steps effectively learn to settle dilemmas according to human preferences.

An important advantage of our approach is its high configurability, by putting the user back in the loop. Indeed, several definitions rely on the human user, such as the ethical thresholds, and the contexts. This means that the system will be aligned to the human users. For example, as the dilemmas are recognized based on the ethical thresholds, users could have a very low amount of dilemmas, and let the system handle most of the actions, whereas other users could ask for higher expectations, getting more dilemmas and thus taking a more active role. Similarly, as the contexts' definition rely on users, they can bring their own mental state, which the machine does not have access to.

An additional goal was to make the system usable to human users. Even though we could not demonstrate this result, due to the lack of resources for a panel experiment, we have kept this in mind when designing the algorithms' extension. For example, whereas some MORL algorithms from the state of the art propose to use vectors of preferences as a criteria to choose the optimal strategy, and thus actions, we argue it might be complicated for a non-expert user to choose such a vector. The relationship between preferences and outcomes is not always clear nor linear (Van Moffaert et al., 2014). Instead, human users need to specify a vector of thresholds that determines which actions are acceptable. This might still be difficult to specify, for both technical and ethical reasons: users need to understand what these thresholds entail, in terms of ethical demands. However, we argue it has a more limited impact: if the thresholds are not appropriately set, two cases may arise. On the one hand, if the thresholds are set unrealistically too high, then the agents will not manage to propose actions for which interests attain the required thresholds. Thus, almost all situations will be labelled as dilemmas, and the system will ask the user for the correct action to choose. This is a problem, as the system will become a burden for the user, but at least it will not lead to undesired actions being taken, as would be the case if a vector of preferences was incorrectly set. On the other hand, if the thresholds are set too low, then the agent will automatically solve some dilemmas that should have been

presented to the user. The taken action will be the one that maximizes the average of moral values: thus, it might not correspond to the human preferences. For these reasons, it would probably be safer, in order to limit a potential impact non-aligned with human preferences from the machine, to use high rather than low thresholds.

Surely it is important for human users to understand how the system works, what the ethical thresholds mean, what are the learned actions' interests, etc. We have already mentioned this point in the results, when we presented the learned actions' interests. As some moral values might be harder than other to learn, it is probable that, on the corresponding dimensions, the interests will be lower. If human users are not aware of this, they might set similar thresholds for all dimensions, which would fail to find acceptable actions. It is essential to remember that the system may very well have limits, either because of some imperfection, e.g., in the learning process, or because it was naturally infeasible in the given situation. Users should not follow the system "blindly".

Our system has also a few limitations, possible short-term improvements, and longer-term research perspectives, all of which we discuss below.

The first potential limitation concerns the moral values themselves, and more specifically their number. In our experiments, we used 4 moral values, as described in our use-case, yet, the proposed algorithms are theoretically not limited in terms of moral values. However, intuitively it seems that, the more moral values we implement, the more difficult it will be to learn actions that satisfy all of them, especially if they conflict. It follows that, if actions cannot satisfy all moral values at once, the majority of situations will be considered as dilemmas. In addition, the more dimensions we use for the interests, the more the Pareto Front will yield many actions. Indeed, it is more likely that, for each action, we will find at least one dimension on which it is not dominated, as the number of dimensions increases. Thus, with more actions in the Pareto Front, human users will have more actions to compare in order to choose the one to be executed in this dilemma, which would increase their cognitive load. Based on this reasoning, we wonder whether there is a relation between the number of moral values and the system's performance; if such a relation exists, what is the maximum, reasonable number of moral values before the system becomes too "chaotic" to be used? 4 moral values seems already a high number to us, as many works in the MORL literature focus only on 2 objectives; nevertheless, it would be interesting to try to apply our proposition to use cases with even more moral values, e.g., 6, 8, or even 10.

The second limitation is the manual partitioning of the action space, when exploring and learning the interesting actions, namely, the exploration profiles. As the profiles are manually set by the designers, it means that some gaps may exist: two different exploration profiles may find actions in two different sub-zones of the action space, but perhaps they may be actions between these two sub-zones, which we cannot find. A better exploration could rely on intrinsic motivation, such as curiosity, to fully explore this action space. For example, the algorithm could begin with similar weights as the ones we defined, and then try to explore in a different zone. If no interesting actions, i.e., actions that are not Pareto-dominated, can be found in this zone, the algorithm could preemptively stop its exploration, and resort to another zone. However, we left this aspect aside, as the exploration of the action space was only a pre-requirement for the rest of contribution. It was not the main part, contrary to the identification of dilemmas and more importantly, the contextualized preferences. Our manual partitioning is already sufficient for the rest of contribution, although it could be improved: this is thus an important perspective.

A more important limitation, which is directly linked with the rest of the thesis, concerns the ability to adapt to changes. As we introduced a bootstrapping phase, which is separated from the deployment phase in which agents are expected to actually exhibit their behaviour, it is harder to adapt to changes, either in the environment's dynamics, or in the reward functions, as we have shown in previous chapters. Indeed, actions, including their parameters, are learned during the bootstrap phase, and "frozen" during the deployment phase, so that they can be compared fairly. If the environment changes during the deployment, in our current algorithm, agents could not adapt any more. We propose two potential solutions to this:

1. The simplest one is to alternate bootstrap and deployment phases regularly. The environment used in the bootstrap phase must be as close as possible to the deployment one, which can be the real world for example. Thus, agents would not be able to adapt immediately to a change, but in the long-term, their behaviour will be "upgraded" after the next bootstrap phase. Note that this still requires some work from the designers, which need to update the bootstrap environment, and to regularly update learning agents as well. It can also be computationally intensive, as the bootstrap phase needs to simulate a lot of time steps.
2. A more complex method could be to make agents able to learn during deployment, similarly to what we have done in the two previous chapters. However, the exploration-exploitation dilemma must be carefully handled, as we need to propose

actions to human users. It will probably require work on the explanation aspect as well: how do we present and compare actions that have not been explored the same number of times? If an action has been less explored than another, a difference of interests between them can be explained either because one is truly better, or because one has more uncertainty. Perhaps works such as Upper-Confidence Bound and similar can be leveraged, to memorize not only the current interests, but also their uncertainty. This approach would require close collaboration with Human-Computer Interaction researchers, so that the correct information can be presented to human users, in a manner that help them make an informed choice.

There is, however, one aspect of adaptation that is separated from the question of learning interesting actions. Indeed, perhaps human users will want to update, at some point, their definition of a context, or to change the chosen action, e.g., because they have discovered new information that made them reflect on their choice. This does not seem technically complicated, based on our representation of a context: we recall that a context comprises a set of lower and upper bounds to recognize situations, and the chosen action. The recognized situations can thus be changed by simply updating the bounds, and the chosen action replaced by another. However, it would require, again, specific work on the interface itself to present the functionality to the users.

Following up on the notion of the user interface, we acknowledge that it is not very clear for the moment, and most certainly confusing for lay users. The tabs that, respectively, present sliders to set the context's bounds, and the actions' plots of interests and parameters, could be improved. Yet, we have chosen to make a simple, prototype interface, as the contribution focuses on the algorithmic capabilities of agents to integrate contextualized preferences from human users. The source of these preferences is, in this regard, a secondary problem of implementation, and the prototype interface suffices to demonstrate, by interacting with the system, that we can indeed settle dilemmas.

Another limitation that is due to a technical implementation detail is that, in the current version, users must choose the bounds and the desired action when a dilemma is presented to them. This is appropriate and sufficiently simple for a laboratory experiment ; however, in a real-world scenario, if a dilemma happens at 3 o'clock in the morning, surely we would not want to force the user to get up and come settle it at this instant. It would be interesting to allow users to interact with the system at a later step, perhaps by making the agent resort to a default selection mechanism, e.g., a random action or taking the one with the maximum average interests, while still memorizing that this dilemma was not settled. Thus, when the user decides to interact with the system, a list of encountered

dilemmas would be presented so that adequate actions can be elicited and corresponding contexts defined.

Finally, we reflect back on some definitions we have proposed, and how different definitions could yield other results. A first definition that can be potentially limiting is the *ethical threshold*: we recall that they correspond to the user-set thresholds, above which the interests of an action are considered acceptable. The definition of these thresholds that we propose is a simple vector of as many dimensions as there are moral values, i.e., objectives in the multi-objective reward, for example $[80\%, 75\%, 90\%, 70\%]$. In this example, the user accepts an action if its interest w.r.t. the 1st moral value attains at least 80% of the theoretical (maximum) interest, *and* at least 75% w.r.t. the 2nd moral value, and so on. Users specify a single threshold; however, we sometimes want to accept different alternatives. For example, a user might say "I would accept an action that satisfies at least 80% of the 1st moral value *and* 75% of the 2nd moral value, *or* an action that satisfies at least 60% of the 1st moral value *and* 90% of the 3rd moral value". As hinted in this example, such preferences can be represented by using disjunctions of conjunctions, i.e., by combining "or" alternatives of "and"-based thresholds. In a pseudo-mathematical format, the previous example could be represented as $(80\% \wedge 75\% \wedge 0\% \wedge 0\%) \vee (60\% \wedge 0\% \wedge 90\% \wedge 0\%)$. Note that it also explicitly allows users to specify thresholds for subsets of values. Whereas this is already feasible with the current definition, it is not recommended as there is a single vector of thresholds: a threshold set to 0% would result in the moral value being completely ignored. When using disjunctions of conjunctions, a value can be ignored in a given vector, but not in another vector. As a result, the ethical thresholds would be more generic, and more flexible. This definition would not be technically difficult to implement, as we simply need to compare the interests to a set of potentially multiple vectors, instead of a several vector. Yet, in order to be effective, more research is needed, both in moral philosophy to ensure that it is an appropriate definition, and in Human-Computer Interaction to ensure that it is easily usable by users.

In the same vein, we now reflect back on the definition of a "user preference": in this contribution, we proposed to simply define a user preference as an action choice. Others definitions could be possible, e.g., "Choose the action that maximizes the *well-being* value", which can be encoded as a formula based on the actions' interests. On the one hand, these definitions would be harder to use for the user, compared to simply selecting the action they prefer. On the other hand, they would allow for more flexibility: if the set of actions changes, e.g., because of adaptation, we can still select an appropriate action,

because the formula is more generic than directly memorizing the desired action. Perhaps a long-term perspective could be to learn such formulas, from the user interactions. For example, based on the chosen actions' interests, the agent could derive a logical formula that mimics the users' choices. This idea appeals again to the notion of symbolic reasoning in a hybrid context that we used in Chapter 5. However, this time the hybrid combination would be endogenous, taking place within the agent itself.

We could even imagine a system that would automatically learn preferences by observing the human user's behaviour and choices in various situations, not necessarily linked to the current application domain, as a matter of fact. This can be linked to the idea of an "ethics bot", which Etzioni & Etzioni (2017) define as follows:

> An ethics bot is an AI program that analyzes many thousands of items of information (not only publicly available on the Internet but also information gleaned from a person's local computer storage and that of other devices) about the acts of a particular individual in order to determine that person's moral preferences.

The idea of ethics bot is a bit far from our work, as we instead propose to directly ask users for their preferences, and we try to teach the agents how to "behave ethically", or, more precisely, to exhibit a behaviour aligned with moral values. However, it illustrates that some parts of our contribution can be linked, and even reused or extended, in different lines of research within the Machine Ethics community.

7 Synthesis, discussion, and perspectives

In this final chapter, we summarize our answers to the problem and research questions, and reflect on the proposed contributions. The first section makes a synthesis of the 3 contributions, and we particularly highlight how they answer the defined objectives. The second section describes the limitations and perspectives offered by our work; we outline here the salient points of each contribution and how they relate to the global architecture, as the details have been discussed in the chapters on each contribution.

7.1 Main results

In the state of the art in Section 2.1, we notably presented a set of properties that we deem important for the implementation of artificial moral agents:

- Continuous domains allow for use-cases that can be difficult to represent using discrete domains.
- Multi-agent is more realistic, as the human society is inherently multi-agent, and artificial agents are bound to be integrated into our society, in one form or another.
- Multiple moral values offer more diversity and better represent human moral reasoning.
- The capacity to adapt to changes is crucial as the social mores constantly evolve.

We have also described our methodology in Section 3.1, for which we emphasize some of the properties:

- Pluri-disciplinarity is important as AI experts do not usually have the necessary domain knowledge, in particular in moral philosophy.
- Putting humans in control and reflecting in terms of Socio-Technical Systems ensures that the (technical) system is aligned with humans, rather than humans being forced to align themselves with the system.

- Diversity, especially of moral values and of human preferences, is a necessity as we have different ethical considerations.

These properties echo the ethical model that we introduced in Section 3.2. This model could be leveraged by other works; one of the main points to retain is that producing systems ethically-aligned, whether they focus on producing "ethical behaviours", or simply take ethical considerations as part of their behaviours, requires a discussion with a large audience, consisting of designers, users, stakeholders, philosophers, etc. As these various people have different knowledge and expertise, this discussion necessitates in itself to reach a common ground of explaining the key choices when constructing a system. These choices, which we name the "ethical injection", reflect what we want the machine to take into account.

Finally, we presented 3 contributions in this manuscript. The first one focused on learning behaviours through reinforcement learning algorithms that particularly emphasize the adaptability of agents to changes in the environment dynamics, including in the reward function, i.e., in the expected behaviour. The second one tackled the construction of the reward function and replaced the traditional mathematical functions with an aggregated multi-agent symbolic judgment, relying on explicitly defined moral values and rules. Finally, the third one opened up the question of dilemmas and replaced the aggregation of the second contribution with a new process that allows learning agents to identify situations of conflicts between moral values, and to address such dilemmas according to human preferences.

We presented in Chapter 1 a set of objectives that should be targeted in this thesis. Table 7.1 shows that each of them is handled in at least one of our 3 contributions. We recall that the 3rd objective, implementing a prototype, was tackled throughout each contribution by the means of the Smart Grid simulator, which was used for the experiments. The simulator was updated when necessary, e.g., making a connection to the Ethicaa platform in the LAJIMA model, or adding a simple user interface to present dilemmas and capture human preferences in the last contribution. Thus, our thesis accordingly answers all objectives. In the following subsections, we recapitulate these contributions and their advantages.

Tab. 7.1.: Recapitulative table of objectives; each one is tackled by at least one contribution.

Objective	Contribution(s)
O1.1 Diversity	C1 + C2 + C3
O1.2 Ethical consensus shifting	C1
O2.1 Learning non-dilemmas	C1
O2.2 Learning dilemmas	C3
O2.3 Reward specification	C2

Nevertheless, this does not mean that these objectives can be considered "closed": we presented in each contribution chapter some perspectives, and, in Section 7.2, we provide additional ones that pertain more to the global proposed architecture rather than individual components. Yet, we consider that we have made a substantial contribution to each of these objectives.

7.1.1 Learning behaviours

The first contribution consists in 2 reinforcement algorithms, named *Q-SOM* and *Q-DSOM* (see Chapter 4). These algorithms target the various challenges identified from the state of the art that pertain to our problematic of learning morally-aligned behaviours. In particular, there was a lack of approaches targeting continuous domains; this is a problem, as such domains may be necessary, or at least more practical, to describe some situations or use-cases, with the example of our Smart Grid energy distribution. This is true for both describing situations, e.g., with the inequality measure, and actions, e.g., various amounts of energy to choose. In order to introduce more diversity in the experiments, as per the methodology and objective O1.1, we introduced multiple learning agents: they each encounter different experiences, and thus exhibit various behaviours; they also embed one of several profiles that dictates their needs and available actions. Another aspect that shaped our algorithms was the notion of privacy: as the learning agents represent human users, it would certainly not be acceptable to share their data with every other agent. Some sharing is necessary at some point, at least in a centralized

component which computes the rewards, yet, the learning agents do not have access to other agents' data, neither their observations, actions, nor rewards. Finally, the most important challenge here is the "Continuous Learning", as defined by Nallur (2020), or as we call it, the ability to adapt to changes. This is particularly crucial within Machine Ethics, as the ethical consensus may shift overtime. We note that these algorithms are not exclusive to the learning of ethical considerations, or morally-aligned behaviours, in the sense that they do not explicitly rely on ethical components, such as moral values, moral supports or transgressions, etc.

In order to address these challenges, the Q-(D)SOM algorithms are based on Self-Organizing Maps (SOMs), which allow them a mapping between continuous and discrete domains. Thus, multi-dimensional and continuous observations can each be associated with a discrete state; similarly, a set of discrete action identifiers are linked to continuous action parameters. The interest of performing an action in a situation, i.e., the expected reward horizon from that action and the resulting situation, is learned in a Q-Table, a tabular structure that relies on discrete state identifiers for the rows, and action identifiers for the columns. The Q-Table, along with the State-(D)SOM and Action-(D)SOM, are used to learn the behaviours in various situations, which corresponds to objective O2.1. Learning mechanisms, and in particular the exploration-exploitation dilemma, were specifically designed to allow adaptation to changes, to address objective O1.2. Regarding the notion of privacy, we designed the simulator and the algorithms in such a way that agents do not obtain information about their neighbours, which seems more acceptable. On the other hand, this complicates their learning, as the behaviours of other agents in the environment constitute a kind of "noise" on which they have no information. In order to counterbalance this effect, we use *Difference Rewards*, which calculate the contribution of each agent, by comparing the current state of the environment with a hypothetical state in which the agent would not have acted.

We have evaluated these algorithms by using several "traditional" reward functions, i.e. mathematical formulas, which target various moral values, such as the well-being of agents, or the equity of comforts. Some of these functions target several moral values at the same time, using simple aggregation, or are designed so that their definition evolves over time: new moral values are added as time goes by. The results show that our algorithms learn better than 2 state-of-the-art baseline algorithms, DDPG and MADDPG. It also seems that the size of the environment, i.e., the number of learning agents, has an impact on the learning performance: it is more difficult for the agents to learn the

dynamics of the environment when there are more elements that impact these dynamics. However, the results remain satisfactory with a high number (50) of agents.

7.1.2 Constructing rewards through symbolic judgments

The second contribution consists in the construction of rewards through the agentification of symbolic judgments (see Chapter 5). The need for this contribution stems mainly from objective O2.3: the need of correctly specifying the expected behaviour to express the significance of moral values, which include preventing the agent from "gaming" or "hacking" the reward function Additionally, a better usability and understandability of the reward function for external users or regulators constitutes a secondary objective for this contribution. Reward hacking is a well-known phenomena in reinforcement learning, in which the agent learns to maximize its received reward by adopting a behaviour that corresponds to the criteria implemented in the reward function, but does not actually represent what the designer had in mind. For example, an agent could circle infinitely around a goal without ever reaching it, to maximize the sum of rewards on an infinite horizon. Within the Machine Ethics community, it would be an important problem if the agent learned to "fake" acting ethically. The reward function should thus be designed to prevent reward hacking, or at least to be easily repaired when reward hacking is detected. The second objective itself encompasses 2 concepts: usability and understandability. Indeed, as the reward function must signal the expected behaviour to the learning agent, and thus appropriately encode the relevant moral values and more largely, ethical considerations, the AI experts are not always the most competent to design it. The specificities of the application domain and the ethical considerations could be best apprehended by domain experts and moral philosophers. Yet, it may be difficult for them to produce a mathematical function that describes the expected behaviour, and this is why another form of reward function is needed. On the other hand, understandability is important for all humans, not only the designers. Human users may be interested in what the expected behaviour is; regulators may want to ensure that the system is acceptable, with respect to some criteria, which can include the law. As we mentioned in the state of the art, this understandability may be compromised when the reward function is a mathematical formula and/or leverages a large dataset.

Whereas the previous contribution used mathematical functions, this contribution proposed to leverage symbolic judgment. We propose a new architecture of an exogenous, multi-agent hybrid combination of neural and symbolic elements. This architecture

agentifies the reward function, where judging agents implement the symbolic judgment that forms the reward signal, and learning agents implement the neural learning process. The two sets of agents act in a decision-judgment-learning loop, and the learning agents learn to capture and exhibit the ethical considerations that are injected through the judgment, i.e., the moral values. An advantage of this learning-judgment combination is that the resulting behaviour can benefit from the flexibility and adaptation of the learning: assuming that, in an example situation, it is infeasible for the agent to satisfy all moral values, the agent will find the action that provides the best trade-off. This computation relies on the Q-Values, which take into account the horizon of expected rewards, i.e., the sequence of received rewards from the next situation. Thus, human designers can simply design the judgment based on the current step, and do not necessarily need to consider future events, yet, learning agents will still take them into account.

Two different models, and their implementation, were proposed. The first one, LAJIMA (Logic-based Agents for JudgIng Morally-embedded Actions), relies on Beliefs-Desires-Intentions (BDI) agents, and more specifically on simplified Ethicaa agents (Cointe et al., 2016), which implement a set of logic rules. On the other hand, the second one, AJAR (Argumentation-based Judging Agents for ethical Reinforcement learning), uses argumentation graphs to determine the correctness of an action with respect to moral values. In both cases, several moral values are explicitly considered, either through different BDI agents, or several argumentation graphs, which adds diversity to the use-case, as per objective O1.1. We note that logic and argumentation are very similar, and, in definitive, there is no theoretical difference in their expressiveness. However, there is a difference in their ease of use: we argue that argumentation graphs are easier to grasp for external users or regulators, thanks to their visual representation. In addition, the attack relationship of argumentation allows to easily prevent a specific behaviour, which further answers our objective O2.3 of avoiding "gaming" behaviours or mis-alignments.

The experiments have shown that learning agents are still able to learn from symbolic judgments. However, the results could have been better, especially for the LAJIMA model. The rules that we defined were perhaps too naive; they suffice as a proof-of-concept first step, but they certainly need to be improved to provide more information to the learning agents. In this regard, a technical implementation advantage of AJAR was that any number of arguments could be used as *positive* or *negative* feedbacks, whereas we fixed this number to the number of dimensions of the action space in LAJIMA, e.g., $d = 6$ for our Smart Grid use-case. Thus, the gradient of the reward was better in AJAR, but at

the expense of an additional hyperparameter: the judgment function, which transforms a set of arguments into a reward, i.e., a scalar.

The main idea to retain from this contribution is to leverage symbolic elements and methods to compute the rewards. We proposed BDI agents with logic rules, and argumentation graphs, as such symbolic methods, but perhaps others could be used as well. Many other forms of logics have been used within Machine Ethics, e.g., non-monotonic logic (Ganascia, 2007a), deontic logic (Arkoudas, Bringsjord, & Bello, 2005), abductive logic (Pereira & Saptawijaya, 2007), and more generally, event calculus (Bonnemains, Saurel, & Tessier, 2018). Still, we argue that argumentation in particular offers interesting theoretical advantages, namely the attack relationship and the graph structure that allow both repairing reward "gaming", and an easy visualization of the judgment process. Our idea hopefully opens a new line of research with numerous perspectives, and an empirical and more exhaustive comparison of symbolic methods for judgment would be important.

7.1.3 Identifying and addressing dilemmas

The third and final contribution focuses on the notion of dilemma, i.e., a situation in which multiple moral values are in conflict and cannot be satisfied at the same time. We want the learning agents to exhibit behaviours that take into account all defined ethical considerations, mainly represented by moral values, but when they cannot do so, how should they act? This is related to our objective O2.2, which states that "Behaviours should consider dilemmas situations, with conflicts between stakes". In the state of the art, we saw that numerous multi-objective reinforcement learning (MORL) approaches already exist, and several of them use preferences to select the best policy. However, their preferences are defined as a set of weights on the objectives: this is, in our opinion, an easily usable definition for machines, but quite hard for humans. It also assumes the existence of numeric tradeoffs between moral values, in other terms, the commensurability of moral values. Thus, we have proposed an alternative definition of "preferences", based on selecting an action. To avoid forcing the user to make a choice every time a dilemma is identified, which can become a burden as the number of dilemmas increase, the learning agents should learn to apply human preferences and automatically select the same actions in similar dilemmas.

To solve these tasks, we extended the Q-(D)SOM algorithms to receive multi-objective rewards, and proposed a 3-steps approach. First, interesting actions are learned, so that, when a dilemma is identified, we can offer various interesting alternative actions to the user, and we know that we can compare them fairly, i.e., that they have similar uncertainty on their interests, or Q-Values. This is done by separating the learning into a bootstrap and a deployment phases, and creating exploration profiles to explore various sub-zones of the action space.

The second step identifies dilemmas, by computing a Pareto Front of the various actions, i.e., retaining only the actions that are not dominated by another action. To determine whether the situation is a dilemma, we compare the remaining actions' interests to *ethical thresholds*, chosen by the human user, which are a set of thresholds, one for each moral value. Thus, different human users may have different expectations for the moral values. To simplify the definition of such ethical thresholds for lay users, we also learn the *theoretical* interests of actions, i.e., the interests they would have if they had received the maximum reward each time they were selected. The actions' interests are compared to the theoretical ones, such that the ratio yields a value between 0 and 1. If at least one of the proposed actions in the Pareto Front, for a given situation, is deemed acceptable, i.e., if all its interests are over their corresponding ethical threshold, the situation is not a dilemma, and the agent takes one of the acceptable actions. Otherwise, the situation is a dilemma, and we must resort to the human preferences. Thus, our proposed definition of a dilemma depends on the human user: the same situation can be considered as a dilemma for one human user, and not a dilemma for another user. We consider this an advantage.

Finally, when a situation is an identified dilemma, we take an action based on the human preferences. We assume that human preferences depend on the situation: we may not always prefer the same moral values. To avoid asking the user too much, e.g., each time a dilemma is encountered, we introduce the notion of *contexts* to group similar dilemmas. Users define a context as a set of bounds, a lower and an upper, for each dimension of the observation space. Intuitively, it is similar to drawing a polytope in the observation space: a dilemma belongs to this context if the situation's observations are inside the polytope. When a context is created, the user also specifies which action should be taken; the same action is automatically selected by the artificial agent whenever an identified dilemma falls within the same context.

An important aspect of our contribution is that we focus on giving control back to the human users. This is done through several mechanisms: the selection of ethical thresholds,

the creation of contexts, and the definition of a *preference* as an action choice rather than a weights vector. The ethical thresholds allow controlling the system's sensibility, i.e., to which degree situations will be flagged as dilemmas. By setting the thresholds accordingly, the human user may choose the desired behaviour, e.g., always taking the action that maximizes, on average, the interests for all moral values, and thus acting in full autonomy, or, at the opposite extreme, recognizing all situations as dilemmas and thus always relying on the human preferences, or anything else in between. This is in line with the diversity targeted by objective O1.1. We note that our approach can therefore be used as a step towards a system that acts as an *ethical decision support*, i.e., a system that helps human users make better decisions by providing them with helpful information (Tolmeijer et al., 2020 ; Etzioni & Etzioni, 2017).

7.2 Limitations and perspectives

Our approach is firstly limited by its proof-of-concept nature. As such, it demonstrates the feasibility of our contributions, but is not (yet) suited for real-world deployment. The experiments happened in a simulated environment, and the moral values and rules we implemented were only defined by us, in our role of "almighty" designers, as mentioned in Section 3.2. We explained in the same section that, in our opinion, such design choices, which have ethical consequences, should be on the contrary discussed in a larger group, including users, stakeholders, moral philosophers, etc. The implementation would also require field experiments to validate that agents' decisions are considered correct by humans, for example through a Comparative Moral Turing Test (cMTT), as defined by Allen et al. (2000). A cMTT is comparable to the Turing Test, except that it compares descriptions of actions performed by two agents, one artificial and one human, instead of discussions. The evaluator is then asked whether one of the two agents appears "less moral", in the sense that its actions are less aligned with moral values, than the other. If the machine is not identified as the "less moral" one significantly more often than the human, then it is deemed to have passed the test. In a similar vein, the simulation and the proposed interface could also be improved by focusing more on the Human-Computer Interaction (HCI) aspects. This would enhance the intelligibility of the system, which would make it easier to use and to accept. It is particularly important for our last contribution that focuses on putting users back in the loop and interacting with them to learn their preferences.

Several interesting lines of research have been left out in this thesis, by lack of time and to focus on the contributions that have been presented in this manuscript. One of the perspectives could be to improve the explainability aspect of agents through another hybrid combination. For example, some works focus on the idea of causal models, or more generally on the notion of consequences, within the Machine Ethics community (Winfield, Blum, & Liu, 2014). This echoes the so-called "model-based" reinforcement learning algorithms, which aim to first learn a model of the environment, and then leverage this model to select the best decisions, similarly to a planning algorithm. We imagine a step further down this line, and argue that such models could also be appealing for explainability purposes: the agent could then "explain", or at least provide elements of explanations, on why an action was taken, by presenting the learned model, the expected consequences of taking this action, and eventually a few counterfactuals, i.e., comparing the expected consequences of other actions. Furthermore, such a model could be leveraged not only as a tool to explain decisions *after taking them*, but even in a *proactive* manner: human users could explore consequences of their preferences in a simulated environment by querying the learned model. This would impact the co-construction of ethics between artificial agents and human users, with enriched opportunities for humans to reflect upon their ethical principles and preferences. The learned model could also improve the action selection interface when a new dilemma is identified: actions could be compared in terms of expected consequences, in addition to their interests, i.e., expected rewards, and parameters. In our architecture, this model could especially draw on our symbolic elements, namely the judging agents. For example, a first idea could be to provide the argumentation graphs to the learning agents, as well as the individual activation of each argument in addition to the computed reward. Learning agents could thus learn which arguments are activated by an action in a situation. This does not represent a complete model of consequences in the world, as the arguments are themselves typically at a higher level; still, it represents a first step. More generally, proactive exploration of consequences could benefit from the Hybrid Neural-Symbolic research: our current approach leverages some sort of exogenous hybrid, as the symbolic and learning elements are in separated sets of agents. If the learning agents were imbued with a symbolic model of their world, they could learn this model through repeated interactions.

Another perspective of improvement is that judging agents are somewhat limited in their agency, in terms of reasoning abilities principally, but potentially interactions between them as well. By this, we mean that the process that determines a numeric reward from the symbolic judgment is described as a function, which basically counts the number

of positive symbols versus the number of negative ones. Instead, this process could be replaced with a deliberate reasoning and decision-making to determine the reward based on the judgment symbols. For example, the judging agent could yield a reward that does not exactly correspond to the learning agent's merits at this specific time step, but which would help it learn better. Drawing on the idea of curriculum learning proposed by Bengio, Louradour, Collobert, & Weston (2009), the judging agents could choose to start with a simpler task, thus rewarding the learning agent positively even if all the moral values are not satisfied, and then gradually increasing the difficulty, requiring more moral values to be satisfied, or to a higher degree. Or, on the contrary, judging agents could give a lower reward than the learning agent normally deserves, because the judging agent believes the learning agent could have done better, based on previous performances. In the two previous examples, the described process would require to follow a long-term strategy, and to retain beliefs over the various learning agents to effectively adapt the rewards to their current condition with respect to the strategy. This exhibits proper agency, instead of a mere mathematical function; in addition, it would improve the quality of the learning, and potentially reduce the designers' burden, as the curriculum learning for example would be done automatically by the judging agents. Note that this idea is, again, linked with the notion of co-construction between judging and learning agents: indeed, the judges would have to retain information about the learning agents, and to adapt their strategy. For example, if learning agents do not manage to learn a given moral value, judging agents could switch their strategy, focusing on another one.

An important limitation of our approach is that we assume we can implement (philosophical) moral values as computer instructions, in this case the judgment process more specifically. In other words, this limitation means that we expect to have a formal definition, translatable in a given programming language, or "computable" in a large sense. However, we have no guarantee for this assumption. For example, how can we define *human dignity*[1]? Currently, we consider that moral values are symbols, which are linked to actions that may support or defeat them, as defined by the moral rules. However, this might be a reduction of the original meaning of moral values, in moral philosophy. This limitation can be seen, in fact, as the "ultimate" question of Machine Ethics: will we be able to implement moral reasoning in a computer system? An advantage, perhaps, of our approach, is that we do not require to code *how* to act, but rather to specify what is praiseworthy, or on contrary punishable. This means that, even if we cannot define, e.g., human dignity entirely, we may still provide the few elements that we can think of and

[1] Many thanks to one of the researchers with whom I discussed at the Arqus conference for this specific example!

formalize. Thus, the agent's resulting behaviour will not exactly exhibit the human dignity value, but some aspects of it, which is better than nothing. The agent will thus exhibit a behaviour with ethical considerations, and as a consequence have a more beneficial impact on our lives and society.

Finally, we have evaluated our contributions on a Smart Grid case, which we described as an example, and mentioned that the proposed approaches were generic. We now detail the necessary steps to adapt them to another domain; these steps are grouped by contribution to make it clearer. Concerning the reinforcement learning algorithm, the first important step is to have an environment that follows the RL framework: sending observations to agents, querying actions from them, executing actions, computing rewards. As part of this step, a significant effort must be applied to the choice and design of the observations (also called "features" in the RL literature): some will provide better results, e.g., by combining several low-level features together. The presented algorithms, Q-SOM and Q-DSOM, are themselves applicable as-is to any environment that follows this RL framework, as they can be parametrized to any size of input (observations) and output (actions) vectors. However, a second step of finding the best hyperparameters will surely be necessary to optimize the agents' behaviours to this new application use-case. Concerning the symbolic judgments for reward functions, more work will be required, as we only provide a framework, be it through Beliefs-Desires-Intentions (BDI) agents or argumentation-based agents. Designers will have to implement the desired reward function using one of these frameworks: this requires wondering which moral values should be focused, what metrics are used to support them, etc. Note that, whereas argumentation can be easily plugged with the learning algorithms, as they both are implemented in Python, BDI agents in JaCaMo are implemented in Java and require an additional technical work to communicate with Python, e.g., through a Web server that will transfer observations and actions back-and-forth between Python and Java. Concerning the last part, dilemmas and human preferences, the general idea and algorithms we propose are generic, as they do not explicitly rely on the domain's specificities, except for the interface that is used to ask the human user. This interface has to present the current situation, e.g., in terms of observations, and the proposed actions' interests and parameters. These elements are specific to the chosen use-case, for example in Smart Grid the current hour is an important part of observations; actions' parameters concern quantities of energy to transfer, etc. An appropriate interface must be built, specifically for the new domain. This requires both a technical work and a bit of reflection about the UX, so that the interface can be usable to lay users.

References

Alcaraz, B. (2021). *Argumentation-based judging agents for ethical reinforcement learning supervision* (Master's thesis). Université Claude Bernard Lyon 1.

Allen, C., Smit, I., & Wallach, W. (2005). Artificial morality: Top-down, bottom-up, and hybrid approaches. *Ethics and Information Technology*, *7*(3), 149–155.

Allen, C., Varner, G., & Zinser, J. (2000). Prolegomena to any future artificial moral agent. *Journal of Experimental & Theoretical Artificial Intelligence*, *12*(3), 251–261.

Amgoud, L., Ben-Naim, J., Doder, D., & Vesic, S. (2017). Acceptability semantics for weighted argumentation frameworks. In *Twenty-sixth international joint conference on artificial intelligence*.

Amgoud, L., Cayrol, C., Lagasquie-Schiex, M.-C., & Livet, P. (2008). On bipolarity in argumentation frameworks. *International Journal of Intelligent Systems*, *23*(10), 1062–1093.

Amgoud, L., & Prade, H. (2009). Using arguments for making and explaining decisions. *Artificial Intelligence*, *173*(3-4), 413–436.

Anderson, M., & Anderson, S. L. (2011). *Machine Ethics*. Cambridge University Press. Retrieved from https://books.google.com?id=N4IF2p4w7uwC

Anderson, M., Anderson, S. L., & Armen, C. (2004). Towards machine ethics. In *AAAI-04 workshop on agent organizations: Theory and practice, san jose, CA*.

Anderson, M., Anderson, S. L., & Berenz, V. (2018). A value-driven eldercare robot: Virtual and physical instantiations of a case-supported principle-based behavior paradigm. *Proceedings of the IEEE*, *107*(3), 526–540.

Arkoudas, K., Bringsjord, S., & Bello, P. (2005). Toward ethical robots via mechanized deontic logic. In *AAAI fall symposium on machine ethics* (pp. 17–23). The AAAI Press Menlo Park, CA, USA.

Ashley, K. D., & McLaren, B. M. (1995). Reasoning with reasons in case-based comparisons. In M. Veloso & A. Aamodt (Eds.), *Case-Based Reasoning Research and Development* (pp. 133–144). Springer Berlin Heidelberg.

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, *47*(2), 235–256.

Avigad, G., Eisenstadt, E., & Cohen, M. W. (2011). Optimal strategies for multi objective games and their search by evolutionary multi objective optimization. In *2011 IEEE conference on computational intelligence and games (CIG'11)* (pp. 166–173). IEEE.

Barrett, L., & Narayanan, S. (2008). Learning all optimal policies with multiple criteria. In *Proceedings of the 25th international conference on machine learning* (pp. 41–47).

Bellman, R. (1966). Dynamic programming. *Science*, *153*(3731), 34–37.

Bench-Capon, T. (2002). Value based argumentation frameworks. *arXiv Preprint Cs/0207059*.

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48). New York, NY, USA: Association for Computing Machinery. http://doi.org/10.1145/1553374.1553380

Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, *27*(4), 819–840.

Berry, D. A., & Fristedt, B. (1985). Bandit problems: Sequential allocation of experiments (monographs on statistics and applied probability). *London: Chapman and Hall*, *5*(71-87), 7–7.

Boarini, R., Laslier, J.-F., & Robin, S. (2009). Interpersonal comparisons of utility in bargaining: Evidence from a transcontinental ultimatum game. *Theory and Decision*, *67*(4), 341–373.

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., & Santi, A. (2013). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, *78*(6), 747–761.

Bonnemains, V. (2019). *Formal ethical reasoning and dilemma identification in a human-artificial agent system.* (PhD thesis). Institut supérieur de l'aéronautique et de l'espace, Toulouse, France.

Bonnemains, V., Saurel, C., & Tessier, C. (2018). Embedded ethics: Some technical and ethical challenges. *Ethics and Information Technology*, *20*(1), 41–58.

Bordini, R. H., El Fallah Seghrouchni, A., Hindriks, K., Logan, B., & Ricci, A. (2020). Agent programming in the cognitive era. *Autonomous Agents and Multi-Agent Systems*, *34*(2), 1–31.

Bordini, R. H., Hübner, J. F., & Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using jason*. John Wiley & Sons.

Bosello, M., & Ricci, A. (2020). From programming agents to educating agents – a jason-based framework for integrating learning in the development of cognitive agents. In L. A. Dennis, R. H. Bordini, & Y. Lespérance (Eds.), *Engineering multi-agent systems* (pp. 175–194). Cham: Springer International Publishing.

Bouguelia, M.-R., Belaïd, Y., & Belaïd, A. (2013). An Adaptive Incremental Clustering Method Based on the Growing Neural Gas Algorithm. In *2nd International Conference on Pattern Recognition Applications and Methods - ICPRAM 2013* (pp. 42–49). Barcelona, Spain: SciTePress. http://doi.org/10.5220/0004256600420049

Bowling, M., & Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, *136*(2), 215–250.

Bremner, P., Dennis, L. A., Fisher, M., & Winfield, A. F. (2019). On Proactive, Transparent, and Verifiable Ethical Reasoning for Robots. *Proceedings of the IEEE*, *107*(3), 541–561. http://doi.org/10.1109/JPROC.2019.2898267

Buchanan, B. G., & Shortliffe, E. H. (1984). *Rule based expert systems: The mycin experiments of the stanford heuristic programming project*. Addison-Wesley Longman Publishing Co., Inc. Retrieved from https://www.shortliffe.net/Buchanan-Shortliffe-1984/MYCIN%20Book.htm

Caminada, M. (2007). Comparing two unique extension semantics for formal argumentation: Ideal and eager. In *Proceedings of the 19th belgian-dutch conference on artificial intelligence (BNAIC 2007)* (pp. 81–87). Utrecht University Press.

Chaput, R., Duval, J., Boissier, O., Guillermin, M., & Hassas, S. (2021). A multi-agent approach to combine reasoning and learning for an ethical behavior. In *Proceedings of the 2021 AAAI/ACM conference on AI, ethics, and society* (pp. 13–23).

Chevaleyre, Y., Endriss, U., Lang, J., & Maudet, N. (2007). A short introduction to computational social choice. In *International conference on current trends in theory and practice of computer science* (pp. 51–69). Springer.

Cointe, N., Bonnet, G., & Boissier, O. (2016). Ethical judgment of agents' behaviors in multi-agent systems. In *Proceedings of the 2016 international conference on autonomous*

agents & multiagent systems (pp. 1106–1114). Richland, SC: International Foundation for Autonomous Agents; Multiagent Systems.

Coste-Marquis, S., Konieczny, S., Marquis, P., & Ouali, M. A. (2012). Weighted attacks in argumentation frameworks. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

de Wildt, T. E., Chappin, E. J. L., van de Kaa, G., Herder, P. M., & van de Poel, I. R. (2019). Conflicting values in the smart electricity grid a comprehensive overview. *Renewable and Sustainable Energy Reviews*, *111*, 184–196. http://doi.org/10.1016/j.rser.2019.05.005

Dennis, L., & Fisher, M. (2018). Practical challenges in explicit ethical machine reasoning. In *International symposium on artificial intelligence and mathematics, ISAIM 2018*.

Dignum, V. (2019). *Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way*. Springer International Publishing. http://doi.org/10.1007/978-3-030-30371-6

Dubois, D., Fargier, H., & Prade, H. (1997). Beyond min aggregation in multicriteria decision:(ordered) weighted min, discri-min, leximin. In *The ordered weighted averaging operators* (pp. 181–192). Springer.

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

Duval, J. (2020). *Judgment of ethics of behavior for learning* (Master's thesis). Université Claude Bernard Lyon 1.

Etzioni, A., & Etzioni, O. (2017). Incorporating ethics into artificial intelligence. *The Journal of Ethics*, *21*(4), 403–418.

EU Commission Task Force for Smart Grids, E. G. 1. of the. (2010). Functionalities of smart grids and smart meters. Retrieved from https://circabc.europa.eu/ui/group/f5b849d3-26ae-4cba-b9f9-6bc6688c5f58/library/b094eb71-ba9e-4c93-b26c-f2cb8240dc05/details

Floridi, L., & Sanders, J. W. (2004). On the Morality of Artificial Agents. *Minds and Machines*, *14*(3), 349–379. http://doi.org/10.1023/B:MIND.0000035461.63578.9d

Foot, P. (1967). The problem of abortion and the doctrine of the double effect. *Oxford Review*, *5*.

Fritzke, B. (1995). A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, *7*.

Gabriel, I. (2020). Artificial intelligence, values, and alignment. *Minds and Machines*, *30*(3), 411–437.

Ganascia, J.-G. (2007a). Ethical system formalization using non-monotonic logics. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 29).

Ganascia, J.-G. (2007b). Modelling ethical rules of lying with answer set programming. *Ethics and Information Technology*, *9*(1), 39–47.

Garcez, A. d'Avila., Dutra, A. R. R., & Alonso, E. (2018). Towards symbolic reinforcement learning with common sense. *arXiv Preprint arXiv:1804.08597*.

Garnelo, M., Arulkumaran, K., & Shanahan, M. (2016). Towards deep symbolic reinforcement learning. *arXiv Preprint arXiv:1609.05518*.

Gini, C. (1936). On the measure of concentration with special reference to income and statistics. *Colorado College Publication, General Series*, *208*(1), 73–79.

Güth, W., Schmittberger, R., & Schwarze, B. (1982). An experimental analysis of ultimatum bargaining. *Journal of Economic Behavior & Organization*, *3*(4), 367–388.

Haas, J. (2020). Moral Gridworlds: A Theoretical Proposal for Modeling Artificial Moral Cognition. *Minds and Machines*. http://doi.org/10.1007/s11023-020-09524-9

Hadjsaïd, N., & Sabonnadière, J.-C. (2013). *Smart grids*. John Wiley & Sons.

Harmelen, F. van, & Teije, A. ten. (2019). A Boxology of Design Patterns for Hybrid Learning and Reasoning Systems. *Journal of Web Engineering*, *18*(1), 97–124. http://doi.org/10.13052/jwe1540-9589.18133

Hayes, C. F., Rădulescu, R., Bargiacchi, E., Källström, J., Macfarlane, M., Reymond, M., et al.others. (2022). A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, *36*(1), 1–59.

Hernandez-Leal, P., Kaisers, M., Baarslag, T., & de Cote, E. M. (2019). A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. *arXiv:1707.09183 [Cs]*. Retrieved from https://arxiv.org/abs/1707.09183

Hiraoka, K., Yoshida, M., & Mishima, T. (2009). Parallel reinforcement learning for weighted multi-criteria model with adaptive margin. *Cognitive Neurodynamics*, *3*(1), 17–24.

Honarvar, A. R., & Ghasem-Aghaee, N. (2009). An artificial neural network approach for creating an ethical artificial agent. In *2009 IEEE international symposium on computational intelligence in robotics and automation-(CIRA)* (pp. 290–295). IEEE.

Ikenaga, A., & Arai, S. (2018). Inverse reinforcement learning approach for elicitation of preferences in multi-objective sequential optimization. In *2018 IEEE international conference on agents (ICA)* (pp. 117–118). IEEE.

Jobin, A., Ienca, M., & Vayena, E. (2019). The global landscape of AI ethics guidelines. *Nature Machine Intelligence*, *1*(9), 389–399. http://doi.org/10.1038/s42256-019-0088-2

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, *78*(9), 1464–1480.

LaCroix, T. (2022). Moral Dilemmas for Moral Machines. *AI and Ethics*. http://doi.org/10.1007/s43681-022-00134-y

Li, H., Oren, N., & Norman, T. J. (2011). Probabilistic argumentation frameworks. In *International workshop on theorie and applications of formal argumentation* (pp. 1–16). Springer.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv Preprint arXiv:1509.02971*.

Liu, C., Xu, X., & Hu, D. (2015). Multiobjective Reinforcement Learning: A Comprehensive Overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(3), 385–398. http://doi.org/10.1109/TSMC.2014.2358639

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*.

Marcus, G. (2018). Deep learning: A critical appraisal. *arXiv Preprint arXiv:1801.00631*.

Mataric, M. J. (1994). Learning to behave socially. *From Animals to Animats*, *3*, 453–462.

Matignon, L., Laurent, G. J., & Le Fort-Piat, N. (2012). Independent reinforcement learners in cooperative markov games: A survey regarding coordination problems. *The Knowledge Engineering Review*, *27*(1), 1–31.

Milchram, C., Van de Kaa, G., Doorn, N., & Künneke, R. (2018). Moral Values as Factors for Social Acceptance of Smart Grid Technologies. *Sustainability*, *10*(8, 8), 2703. http://doi.org/10.3390/su10082703

Moor, J. (2009). Four kinds of ethical robots. *Philosophy Now*, *72*, 12–14.

Mukai, Y., Kuroe, Y., & Iima, H. (2012). Multi-objective reinforcement learning method for acquiring all pareto optimal policies simultaneously. In *2012 IEEE international conference on systems, man, and cybernetics (SMC)* (pp. 1917–1923). IEEE.

Murukannaiah, P. K., Ajmeri, N., Jonker, C. M., & Singh, M. P. (2020). New foundations of ethical multiagent systems. In *Proceedings of the 19th international conference on autonomous agents and MultiAgent systems* (pp. 1706–1710).

Nallur, V. (2020). Landscape of machine implemented ethics. *Science and Engineering Ethics*, *26*(5), 2381–2399.

Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, *53*(3), 139–154. http://doi.org/https://doi.org/10.1016/j.jmp.2008.12.005

Nofal, S., Atkinson, K., & Dunne, P. E. (2021). Computing grounded extensions of abstract argumentation frameworks. *The Computer Journal*, *64*(1), 54–63.

O'Neill, J. (2017). Pluralism and incommensurability. In *Routledge handbook of ecological economics* (pp. 227–236). Routledge.

Ong, S., & Clark, N. (2014). Commercial and residential hourly load profiles for all TMY3 locations in the united states [dataset]. http://doi.org/10.25984/1788456

Panait, L., & Luke, S. (2005). Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, *11*(3), 387–434. http://doi.org/10.1007/s10458-005-2631-2

Papoudakis, G., Christianos, F., Rahman, A., & Albrecht, S. V. (2019). Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv Preprint arXiv:1906.04737*.

Pereira, L. M., & Saptawijaya, A. (2007). Modelling morality with prospective logic. In *Portuguese conference on artificial intelligence* (pp. 99–111). Springer.

Rădulescu, R., Mannion, P., Roijers, D. M., & Nowé, A. (2019). Multi-objective multi-agent decision making: A utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, *34*(1), 10. http://doi.org/10.1007/s10458-019-09433-x

Rodriguez-Soto, M., Lopez-Sanchez, M., & Rodriguez-Aguilar, J. A. (2021). Multi-objective reinforcement learning for designing ethical environments. In *Proceedings of the 30th international joint conference on artificial intelligence* (pp. 1–7).

Roijers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, *48*, 67–113. http://doi.org/10.1613/jair.3987

Rougier, N., & Boniface, Y. (2011). Dynamic self-organising map. *Neurocomputing*, *74*(11), 1840–1847.

Russell, S. (2021). Human-compatible artificial intelligence. *Human-Like Machine Intelligence*, 3–23.

Saisubramanian, S., Kamar, E., & Zilberstein, S. (2021). A multi-objective approach to mitigate negative side effects. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 354–361).

Shavlik, J. W. (1994). Combining symbolic and neural learning. *Machine Learning*, *14*(3), 321–331.

Shavlik, J. W., & Towell, G. G. (1991). An approach to combining explanation-based and neural learning algorithms. In *Applications of learning and planning methods* (pp. 71–98). World Scientific.

Shteingart, H., & Loewenstein, Y. (2014). Reinforcement learning and human behavior. *Current Opinion in Neurobiology*, *25*, 93–98. http://doi.org/https://doi.org/10.1016/j.conb.2013.12.004

Smith, A. J. (2002a). Applications of the self-organising map to reinforcement learning. *Neural Networks*, *15*(8-9), 1107–1124.

Smith, A. J. (2002b). *Dynamic generalisation of continuous action spaces in reinforcement learning: A neurally inspired approach* (PhD thesis). University of Edinburgh.

Stahl, B. C. (2022). From computer ethics and the ethics of AI towards an ethics of digital ecosystems. *AI and Ethics*, *2*(1), 65–77.

Subramanian, A., Chitlangia, S., & Baths, V. (2022). Reinforcement learning and its connections with neuroscience and psychology. *Neural Networks*, *145*, 271–287. http://doi.org/10.1016/j.neunet.2021.10.003

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tally, H., Rowena, R., & David, W. (2019). Smart Grids and Ethics: A Case Study. *The ORBIT Journal*, *2*(2), 1–28. http://doi.org/10.29297/orbit.v2i2.108

Thomson, J. J. (1976). Killing, letting die, and the trolley problem. *The Monist*, *59*(2), 204–217.

Tolmeijer, S., Kneer, M., Sarasua, C., Christen, M., & Bernstein, A. (2020). Implementations in machine ethics: A survey. *ACM Computing Surveys (CSUR)*, *53*(6), 1–38.

Towell, G. G. (1991). *Symbolic knowledge and neural networks: Insertion, refinement and extraction.* (PhD thesis). University of Wisconsin - Madison.

Van Moffaert, K., Brys, T., Chandra, A., Esterle, L., Lewis, P. R., & Nowé, A. (2014). A novel adaptive weight selection algorithm for multi-objective multi-agent reinforcement learning. In *2014 international joint conference on neural networks (IJCNN)* (pp. 2306–2314). IEEE.

Varela, F. J. (1999). *Ethical know-how: Action, wisdom, and cognition*. Stanford University Press.

Waldrop, M. M. (1987). A question of responsibility. *AI Magazine*, *8*(1), 28. http://doi.org/10.1609/aimag.v8i1.572

Wang, Y., Yao, Q., Kwok, J. T., & Ni, L. M. (2020). Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (Csur)*, *53*(3), 1–34.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292.

Whittlestone, J., Nyrup, R., Alexandrova, A., & Cave, S. (2019). The Role and Limits of Principles in AI Ethics: Towards a Focus on Tensions. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (pp. 195–200). New York, NY, USA: Association for Computing Machinery. http://doi.org/10.1145/3306618.3314289

Whitworth, B., & De Moor, A. (2003). Legitimate by design: Towards trusted socio-technical systems. *Behaviour & Information Technology*, *22*(1), 31–51.

Wiener, N. (1954). *The human use of human beings: Cybernetics and society*. New York: Doubleday Anchor.

Winfield, A. F., Blum, C., & Liu, W. (2014). Towards an ethical robot: Internal models, consequences and ethical action selection. In *Conference towards autonomous robotic systems* (pp. 85–96). Springer.

Wolpert, D. H., & Tumer, K. (2002). Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems* (pp. 355–369). World Scientific.

World Economic Forum. (2015). Value Alignment | Stuart Russell. Retrieved from https://www.youtube.com/watch?v=WvmeTaFc_Qw

Wu, Y.-H., & Lin, S.-D. (2018). A low-cost ethics shaping approach for designing reinforcement learning agents. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 32).

Yliniemi, L., & Tumer, K. (2014b). Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In *Asia-pacific conference on simulated evolution and learning* (pp. 407–418). Springer.

Yliniemi, L., & Tumer, K. (2014a). Multi-objective Multiagent Credit Assignment Through Difference Rewards in Reinforcement Learning. In G. Dick, W. N. Browne, P. Whigham, M. Zhang, L. T. Bui, H. Ishibuchi, ... K. Tang (Eds.), *Simulated Evolution and Learning* (pp. 407–418). Springer International Publishing.

Yu, D., Yang, B., Liu, D., & Wang, H. (2021). A survey on neural-symbolic systems. *arXiv Preprint arXiv:2111.08164*.

Yu, H., Shen, Z., Miao, C., Leung, C., Lesser, V. R., & Yang, Q. (2018). Building ethics into artificial intelligence. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 5527–5533).

Yu, X., Cecati, C., Dillon, T., & Simões, M. G. (2011). The new frontier of smart grids. *IEEE Industrial Electronics Magazine*, *5*(3), 49–63. http://doi.org/10.1109/MIE.2011.942176

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353.

A

Mathematical notations and symbols

We list here both classic mathematical notations, e.g., $\mathbb{R}$, and some symbols specific to our work, e.g., s.

$\mathbb{R}$: the "reals", or set of all real numbers.

t : a time step.

$|X|$: depending on the context, either the cardinality of a set X, or the absolute value of a real X.

$\mathcal{L}$: the set of learning agents.

$\mathbb{A}_l$: the action space of a learning agent l, typically $\mathbb{A}_l \subseteq \mathbb{R}^d$.

d : number of dimensions of the agent's action space.

$\mathbf{a}$: an action in the action space, i.e., a vector of d dimensions.

$\mathbb{O}_l$: the observation space of a learning agent l, typically $\mathbb{O}_l \subseteq \mathbb{R}^g$.

g : number of dimensions of the agent's observation space.

$\mathbf{o}$: observations in the observation space, i.e., a vector of g dimensions.

γ : Discount factor.

$\mathcal{U}$: set of neurons in the State-(D)SOM, i.e., the neurons that learn the observation space.

$\mathcal{W}$: set of neurons in the Action-(D)SOM, i.e., the neurons that learn the action space.

$\mathbf{U_i}$: prototype vector associated to neuron i of the State-(D)SOM.

$\mathbf{W_j}$: prototype vector associated to neuron j of the Action-(D)SOM.

$\mathtt{P}_U(i)$: position of neuron i in the State-(D)SOM.

$\mathtt{P}_W(j)$: position of neuron j in the Action-(D)SOM.

ψ_U : neighborhood of the State-(D)SOM.

ψ_W : neighborhood of the Action-(D)SOM.

$\mathtt{Q}$: the Q-Function, or Q-Table, which stores the interest for every state-action pair.

s : usually, a discrete state identifier.

$\mathcal{J}$: the set of judging agents.

$\mathbb{B}$: the space of all possible beliefs about a current situation.

$\mathcal{B}$: a set of beliefs about a current situation, with $\mathcal{B} \in \mathbb{B}$.

$\mathcal{V}$: the set of possible moral valuations, $\mathcal{V} = \{moral, immoral, neutral\}$.

$\mathtt{ME}_j$: the Moral Evaluation function of a judging agent j, $\mathtt{ME}_j : \mathbb{B} \times \mathbb{R} \rightarrow \mathcal{V}$.

$\mathtt{Judgment}_j$: the Judgment function of a judging agent j, $\mathtt{Judgment} : \mathcal{L} \rightarrow \mathcal{V}^d$.

$\mathtt{F}$: the Feedback function.

$\mathcal{A}rgs$ or $AF_{[\mathcal{A}rgs]}$: a set of arguments in an Argumentation Framework for Judging a Decision (AFJD).

$\mathcal{A}tt$ or $AF_{[\mathcal{A}tt]}$: a binary attack relationship in an AFJD.

$\mathcal{F}_p$ or $AF_{[\mathcal{F}_p]}$: the set of *pros* arguments in an AFJD.

$\mathcal{F}_c$ or $AF_{[\mathcal{F}_c]}$: the set of *cons* arguments in an AFJD.

$\mathtt{J}_j$: the Judgment function of a judging agent j, $\mathtt{J}_j : \mathcal{P}(AF_j) \rightarrow \mathbb{R}$.

$\mathtt{g_{agr}}$: the Aggregation function, $\mathtt{g_{agr}} : \mathbb{R}^{|\mathcal{J}|} \rightarrow \mathbb{R}$.

grd : the grounded extension of an AFJD.

$>_{Pareto}$: the Pareto-dominance operator : $\mathbf{x} >_{Pareto} \mathbf{y} \Leftrightarrow (\forall i : x_i \geq y_i)$ and $(\exists j : x_j > y_j)$.

$\mathcal{P}$: the set of all exploration profiles.

p an exploration profile $\in \mathcal{P}$.

$\mathcal{S}$: the set of discrete state identifiers.

$\mathtt{States}$: function that returns a state identifier from an observations vector, $\mathtt{States} : \mathbb{O} \rightarrow \mathcal{S}$.

$\mathcal{A}$: the set of discrete action identifiers.

$\mathtt{Actions}$: function that returns action parameters from a discrete action identifier, $\mathtt{Actions} : \mathcal{A} \rightarrow \mathbb{A}_l$.

$\texttt{PF}$: function that returns a Pareto Front (PF) from a set of possible actions.

$\mathbb{Z}$: the space of possible ethical thresholds.

$\mathcal{C}$: the set of all learned contexts.

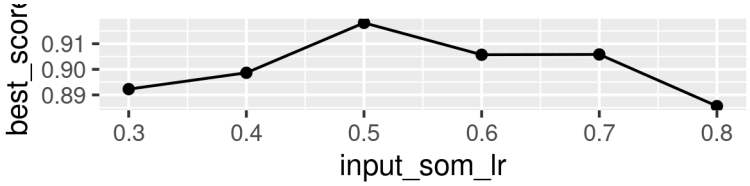
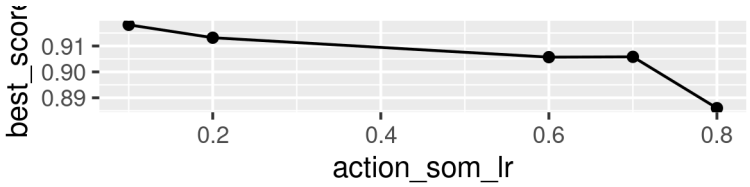
B

Learning algorithms' hyperparameters

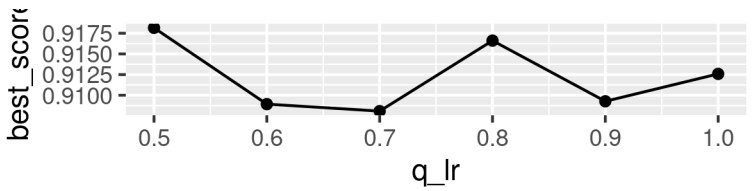
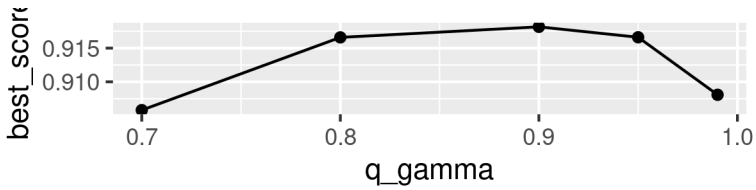
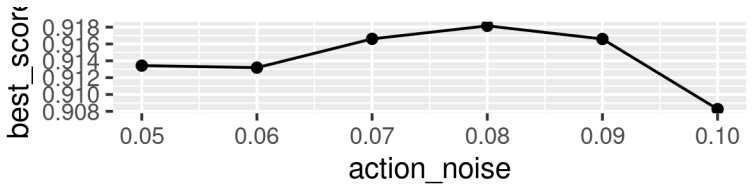
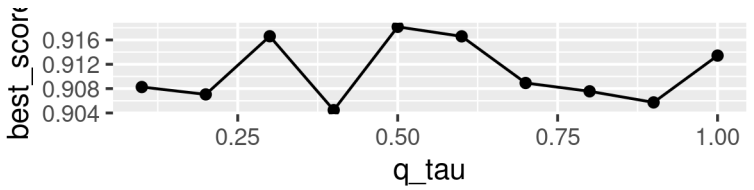
The following tables show a comparison of various hyperparameters for each of the learning algorithms: QSOM, QDSOM, DDPG, and MADDPG. For each of them, numerous combinations of hyperparameters were tried multiple times, and the *score* of each run was recorded. We recall that, in this context, *score* is defined as the average global reward over the time steps, where the global reward is the reward for all agents, e.g., the equity in the environment. The mean score is computed for all runs using the same combination of hyperparameters. Then, for each hyperparameter, we compute the maximum of the mean scores attained for each value of the parameter.

This method simplifies the interaction between hyperparameters, as we take the maximum. For example, a value v_1 of a hyperparameter h_1 could yield most of the time a low score, except when associated when used in conjunction with a value v_2 of another hyperparameter h_2. The maximum will retain only this second case, and ignore the fact that, for any other combination of hyperparameters, setting $h_1 = v_1$ yields a low score. This still represents valuable information, as we are often more interested in finding the best hyperparameters, which yield the maximum possible score.

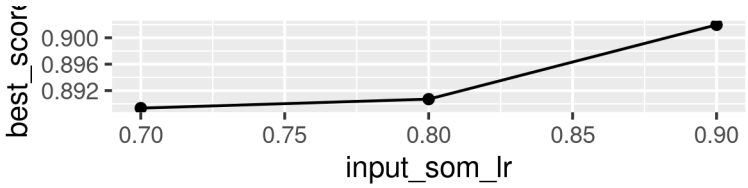
Tab. B.1.: Comparison of hyperparameters' values for the Q-SOM algorithm.

Hyperparameter	Best score per value
input_som_lr	
action_som_lr	

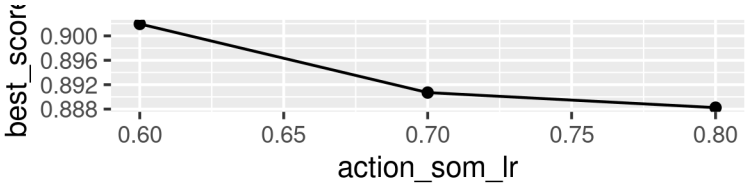
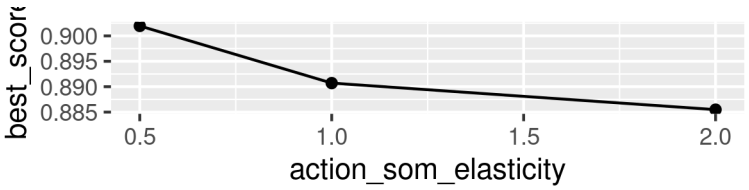
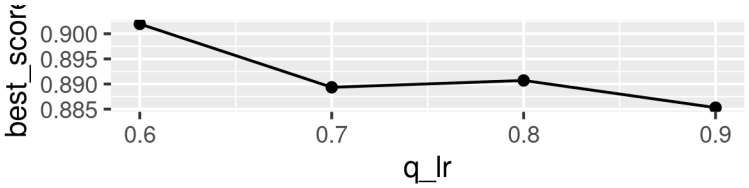
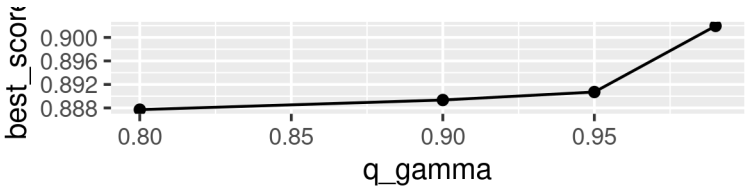
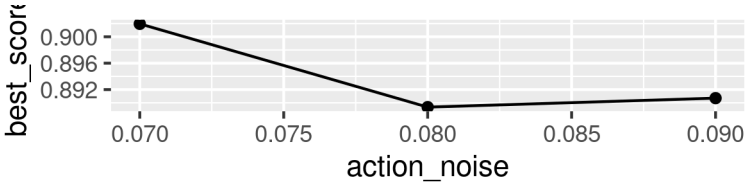
Tab. B.1.: Comparison of hyperparameters' values for the Q-SOM algorithm.

Hyperparameter	Best score per value
q_lr	
q_gamma	
action_noise	
q_tau	

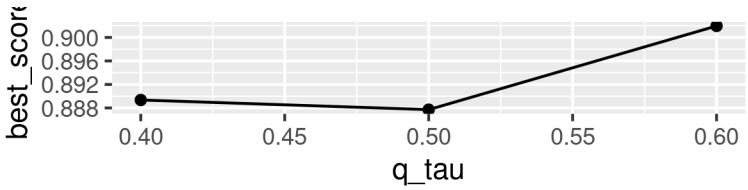
Tab. B.2.: Comparison of hyperparameters' values for the Q-DSOM algorithm.

Hyperparameter	Best score per value
input_som_lr	

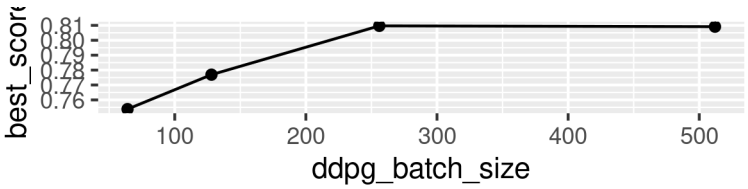
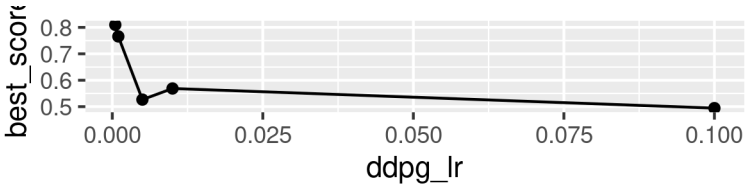
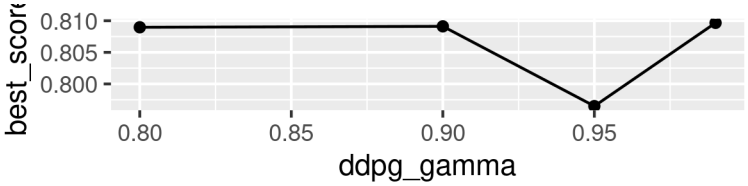
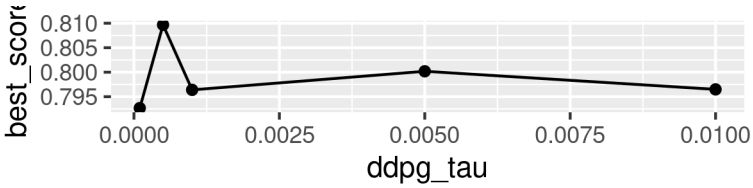
Tab. B.2.: Comparison of hyperparameters' values for the Q-DSOM algorithm.

Hyperparameter	Best score per value
input_som_elasticity	
action_som_lr	
action_som_elasticity	
q_lr	
q_gamma	
action_noise	

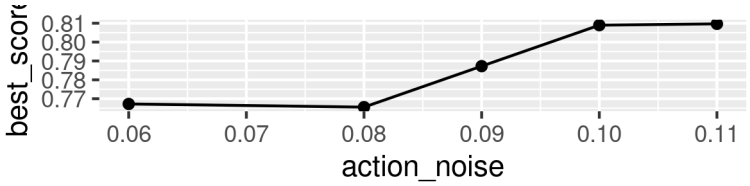
Tab. B.2.: Comparison of hyperparameters' values for the Q-DSOM algorithm.

Hyperparameter	Best score per value
q_tau	

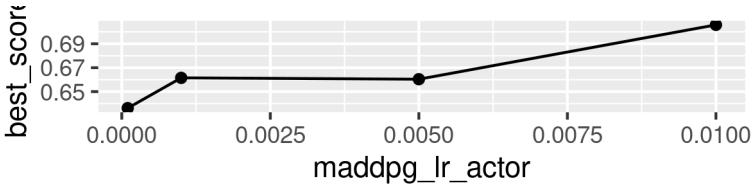
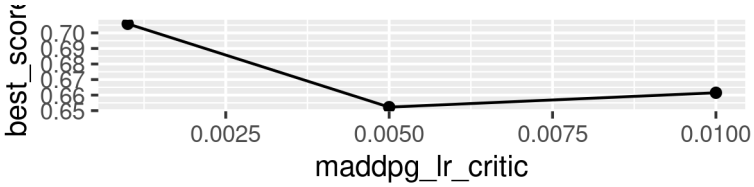
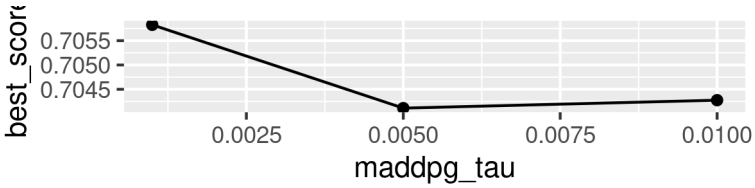
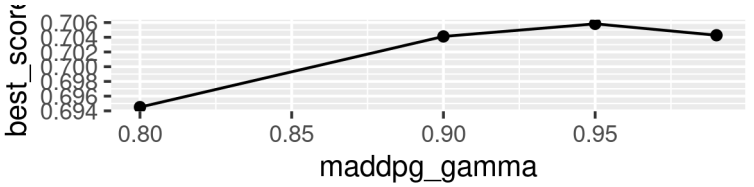
Tab. B.3.: Comparison of hyperparameters' values for the DDPG algorithm.

Hyperparameter	Best score per value
ddpg_batch_size	
ddpg_lr	
ddpg_gamma	
ddpg_tau	

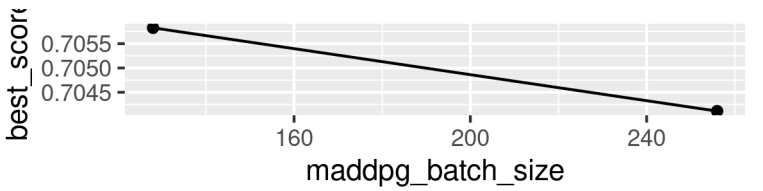
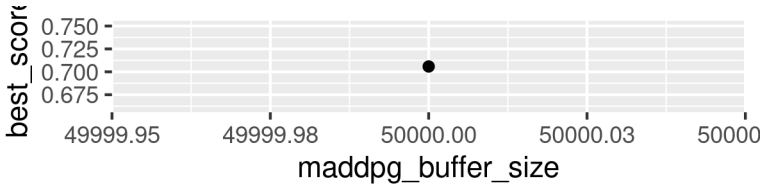
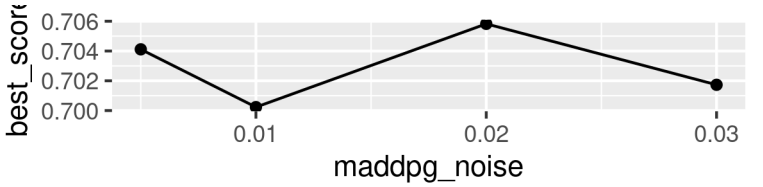
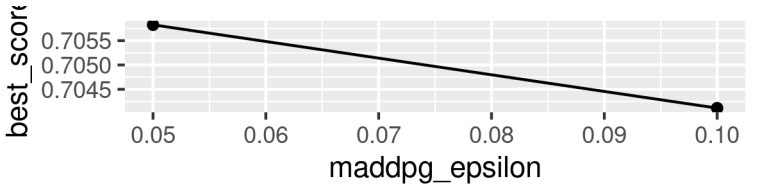
Tab. B.3.: Comparison of hyperparameters' values for the DDPG algorithm.

Hyperparameter	Best score per value
action_noise	

Tab. B.4.: Comparison of hyperparameters' values for the DDPG algorithm.

Hyperparameter	Best score per value
maddpg_lr_actor	
maddpg_lr_critic	
maddpg_tau	
maddpg_gamma	

Tab. B.4.: Comparison of hyperparameters' values for the DDPG algorithm.

Hyperparameter	Best score per value
maddpg_batch_size	
maddpg_buffer_size	
maddpg_noise	
maddpg_epsilon	

Logic-based rules

C

We present below the judgment rules for each of the moral values that were used in the experiments.

Affordability:

```
valueSupport(sell_energy(X, Payoff), "improve_payoff") :-
    X > 100.

valueDefeat(buy_energy(X, Payoff, Budget), "improve_payoff") :-
    X > 100 & Payoff < (-Budget).
```

Environmental sustainability:

```
valueDefeat(buy_energy(X), "promote_grid_autonomy") :-
    X > 100.
valueDefeat(sell_energy(X), "promote_grid_autonomy") :-
    X > 100.
valueDefeat(grid_consumption(X, OverConsumption),
            "balance_supply_demand") :-
    X > 100 & OverConsumption > (3 / 10).
valueDefeat(storage_consumption(X, EnergyWaste),
            "balance_supply_demand") :-
    X > 100 & EnergyWaste > (3 / 10).

valueSupport(buy_energy(X), "promote_grid_autonomy") :-
    X <= 100.
valueSupport(sell_energy(X), "promote_grid_autonomy") :-
    X <= 100.
valueSupport(give_energy(X), "promote_grid_autonomy") :-
    X > 0.
valueSupport(grid_consumption(X), "promote_grid_autonomy") :-
    X > 0.
```

Inclusiveness:

```
valueDefeat(buy_energy(X,Equity,WellBeing,GlobalWellBeing),
            "promote_justice") :-
    Equity < (9 / 10) &
    Threshold = (12 / 10) * GlobalWellBeing &
    WellBeing > Threshold.
valueDefeat(give_energy(X,Exclusion), "promote_justice") :-
    Exclusion > (5 / 10).
valueDefeat(grid_consumption(X,Equity,WellBeing,GlobalWellBeing),
            "promote_justice") :-
    Equity < (9 / 10) &
    Threshold = (12 / 10) * GlobalWellBeing &
    WellBeing > Threshold.
valueDefeat(sell_energy(X,Exclusion), "promote_justice") :-
    Exclusion > (5 / 10).
valueDefeat(sell_energy(X,Equity), "promote_justice") :-
    Equity < (9 / 10).


valueSupport(buy_energy(X,Exclusion), "promote_justice") :-
    Exclusion >= (5 / 10).
valueSupport(buy_energy(X,Equity,WellBeing,GlobalWellBeing),
             "promote_justice") :-
    Equity < (9 / 10) &
    Threshold = (9 / 10) * GlobalWellBeing &
    WellBeing < Threshold.
valueSupport(give_energy(X,Equity,Exclusion), "promote_justice") :-
    Equity < (9 / 10) &
    Exclusion < (5 / 10).
valueSupport(grid_consumption(X,Exclusion), "promote_justice") :-
    Exclusion >= (5 / 10).
valueSupport(grid_consumption(X,Equity,WellBeing,GlobalWellBeing),
             "promote_justice") :-
    Equity < (9 / 10) &
    Threshold = (9 / 10) * GlobalWellBeing &
    WellBeing < Threshold.
```

Security of supply:

```
valueSupport(grid_consumption(X, WellBeing, GlobalWellBeing),
             "promote_comfort") :-
    X > 100 &
    WellBeing >= (8 / 10).

valueSupport(storage_consumption(X, WellBeing, GlobalWellBeing),
             "promote_comfort") :-
     X > 100 &
     WellBeing >= (8 / 10).
```

D Argumentation graphs

As previously, we show the "rules" for judgment, this time in the form of argumentation graphs, represented by arguments as nodes, and attacks as edges. Arguments can be either a pro-argument, in the $\mathcal{F}_p$, a con-argument, in the $\mathcal{F}_c$, or a neutral argument. Let us also recall that arguments can be activated or not, based on the context. We also

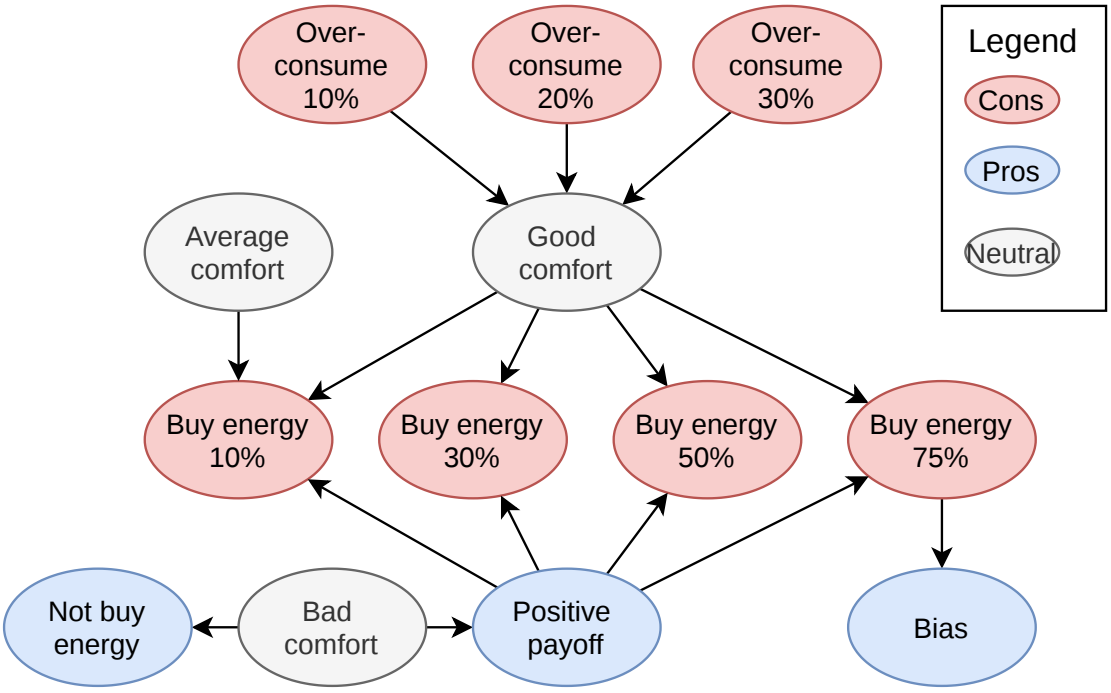


Fig. D.1.: Argumentation graph of the Affordability moral value.

show in Figure D.5 an example of the judgment process using argumentation graphs. The graph is based on the *Affordability* moral value presented above, and updated to judge a specific situation. In this situation, the currently judged agent has achieved a very good comfort, by buying a lot of energy, but still has a positive payoff. Also, the agent did not over-consume from the grid. Thus, some arguments, such as "Over-consume 10%", "Over-consume 20%", and "Over-consume 30%" are disabled: they do not count any more in this situation, and their attacks are removed as well. These disabled arguments are shown as more transparent and drawn as "sketchy", compared to the remaining

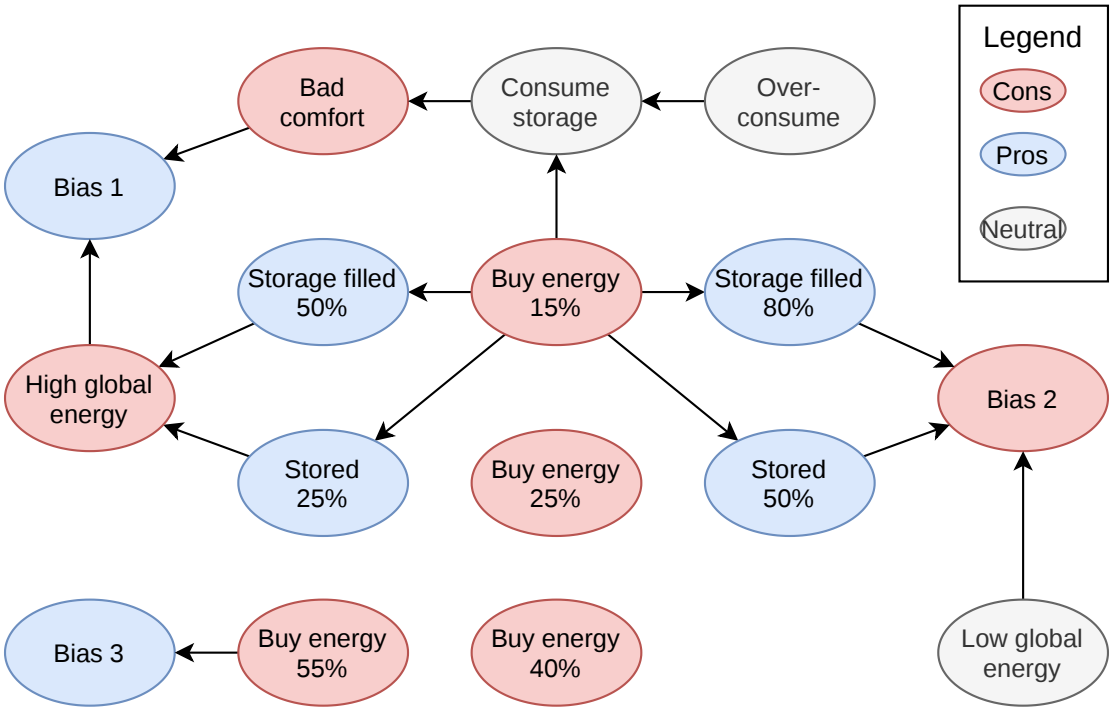


Fig. D.2.: Argumentation graph of the Environmental sustainability moral value.

activated arguments. Then, we want to compute the *grounded extension*, based on only the activated arguments. The grounded extension allows to remove the arguments "Buy energy": although they were activated, they are also attacked by at least one argument, in this case "Good comfort" and "Positive payoff". Intuitively, this means that it is acceptable to buy a lot of energy, according to the *affordability* moral value, if this means we can obtain a good comfort, or if we still have a positive payoff. Note that, for example, "Over-consume" arguments themselves attack the "Good comfort", meaning that we cannot justify buying a lot of energy to get a good comfort, if we also consume too much from the grid. However, in the current situation, the "Over-consume" arguments were disabled, and thus ignored to compute the grounded; a similar reasoning applies to "Bad comfort", with respect to "Positive payoff". As the "Buy energy 4" argument is killed, the "Bias" argument is kept in the grounded: it is somehow defended by "Good comfort" and "Positive payoff". The arguments in the grounded extension are represented by a bold border, and are in this situation: "Average comfort", "Good comfort", "Positive payoff", and "Bias". Finally, the judgment function takes these arguments, identifies those in the $\mathcal{F}_p$ and in the $\mathcal{F}_c$, and potentially compares them to the total number of pros and cons arguments in the original graph. For example, we can see that 2 out of the 3 possible pros arguments are in the grounded extension: "Not buy energy" was disabled because of the situation. None of the cons arguments are in the grounded extension. Thus, the

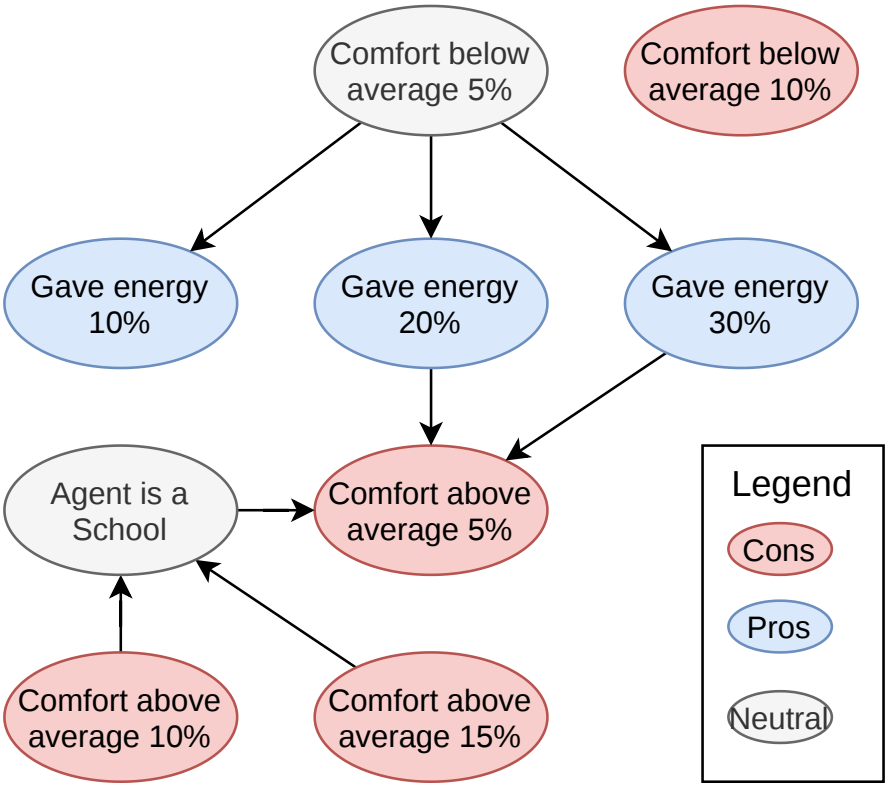


Fig. D.3.: Argumentation graph of the Inclusiveness moral value.

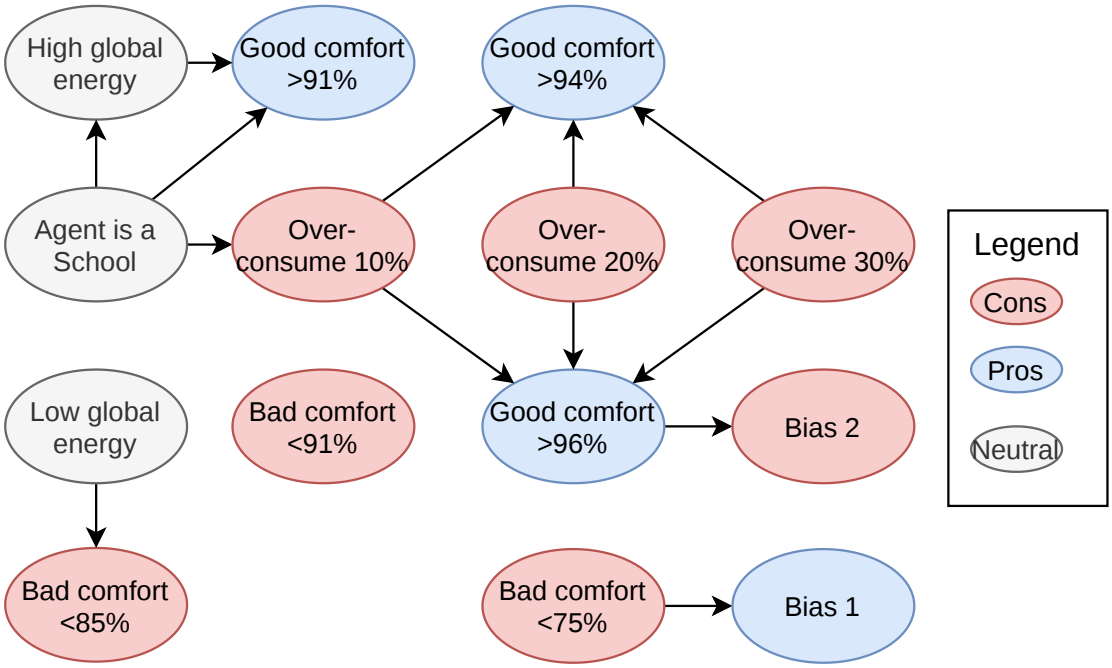


Fig. D.4.: Argumentation graph of the Security of supply moral value.

resulting reward may be high, but not the maximum, depending on the actual judgment function.

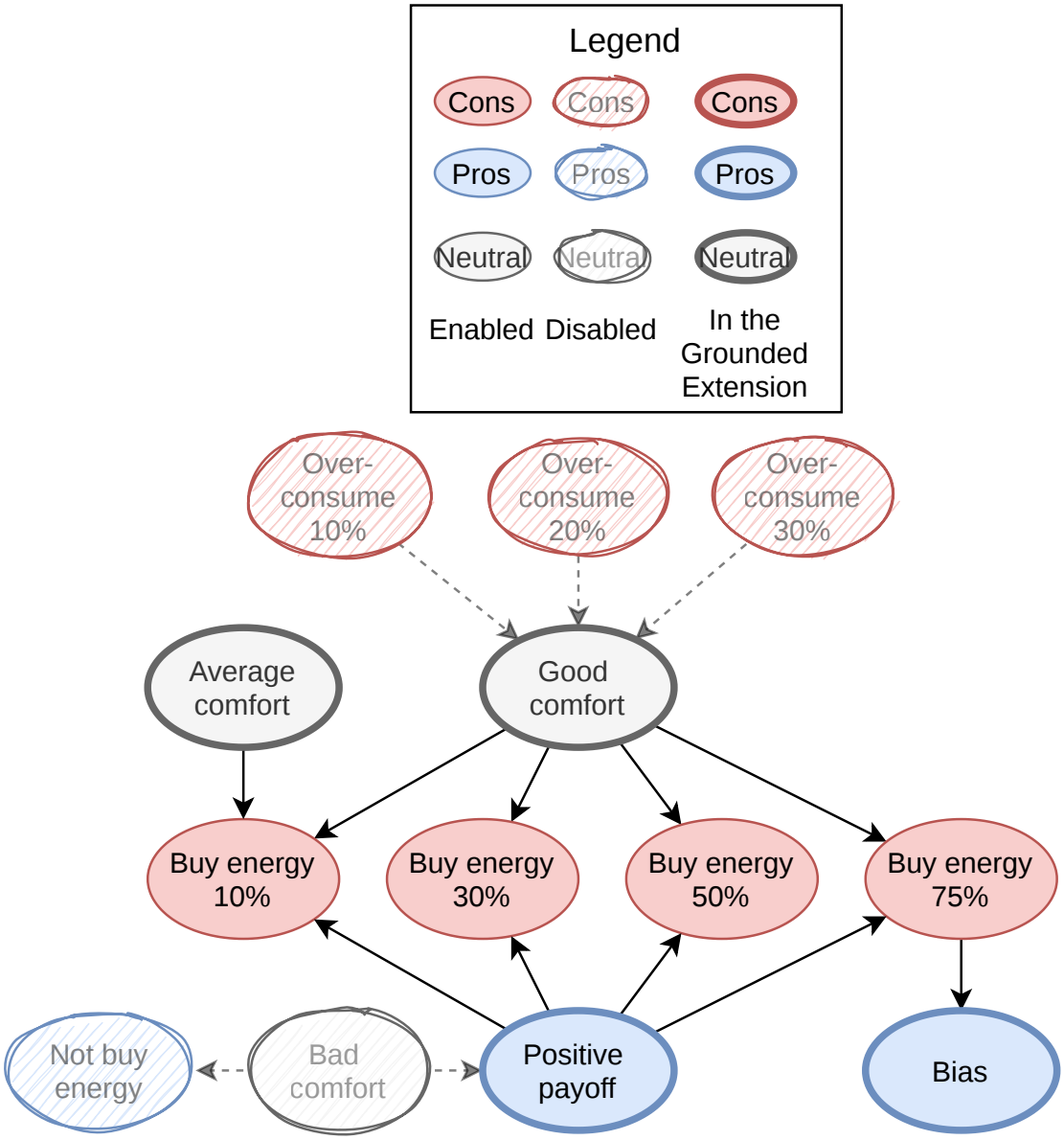


Fig. D.5.: Example of judgment process on the Affordability graph. The judgment relies on the grounded extension, computed from the activated arguments, ignoring the disabled ones.

E

Unique contexts and occurrences

We have described in Section 6.4 a “naive” definition for *context*, which is based on the exploration profiles and their state discretization. Let us recall that each profile has its own State-(D)SOM that is used to discretize a continuous observation vector into a discrete state identifier. The “naive” definition simply takes the list of state identifiers, i.e., the state identifier determined by each exploration profile, as the context. For example, such a context could be $[35, 23, 43, 132, 12]$, meaning that the first profile has discretized the observations as state 35, the second profile as 23, etc. However, we also mentioned that this naive method yielded too many contexts: we expand this here and detail the number of times each unique context appeared. Figure E.1 shows the proportion and

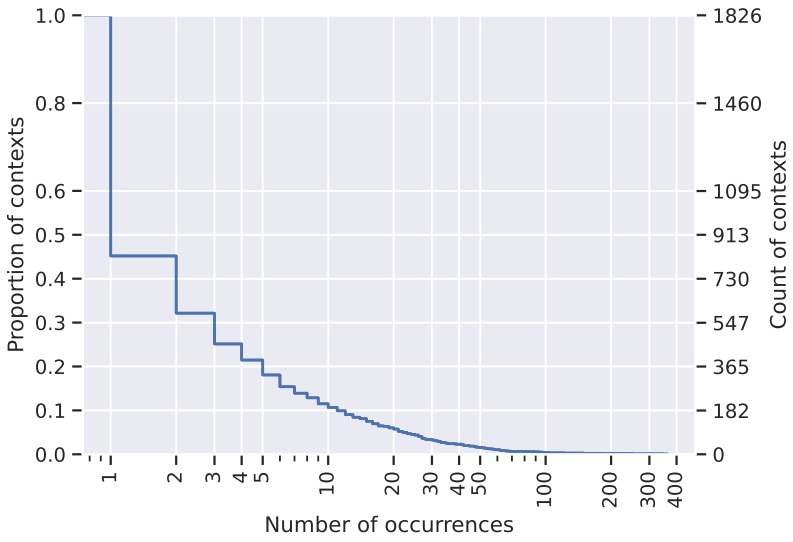


Fig. E.1.: Count and proportion (Y axes) of contexts appearing at least X times, using the "naive" definition, i.e., taking the list of discretized states by exploration profiles as the context.

count (respectively on the left-side Y axis and right-side Y axis) of contexts that appeared at last X times in the simulation. The first information we can grasp for this figure is the total number of unique contexts, which is $1,826$. This is an improvement, when

compared to the $10,000$ time steps of the simulation, yet it is too much for a human user: it means we have to ask $1,826$ times during the simulation. The second information is that most of the contexts appear very rarely: 55% of contexts have a single occurrence. About 30% of contexts appear more than 3 times, 10% appear more than 10 times, and so on. Although one context managed to occur about 400 times, the vast majority of contexts are rare. This means that the human user would spend time setting preferences, only for a small effect. The mean number of occurrences was around 5.48. On contrary,

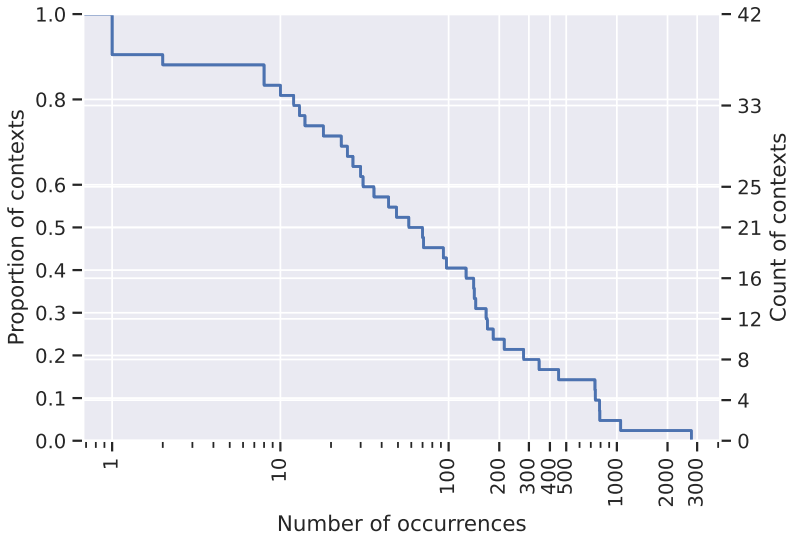


Fig. E.2.: Count and proportion (Y axes) of contexts appearing at least X times, using the "human" definition, i.e., asking the user to set context bounds.

the human-based definition, which relies on the human user to define the bounds of a context when a new context is necessary, has managed to create less unique contexts. This is highlighted by Figure E.2, which shows a plot with a similar structure as the previous one. However, in this case, we have few contexts: only 42 unique contexts, which is a huge improvement over the $1,826$ naive ones. Also, many contexts appeared several times: more than 80% of them appeared at least 10 times, about 20% appeared more than 200 times, and approximately 10% appeared more than 700 times. The mean number of occurrences, with this definition, was 238.1. This very high mean gives us another information: contrary to the previous case, where the majority of time steps were associated to various contexts that only appeared once or twice, with this new definition, the majority of steps are captured by only few contexts (less than 4) that appear thousands of times.

Publications

F

A Multi-Agent Approach to Combine Reasoning and Learning for an Ethical Behavior

International Conference

Rémy Chaput, Jérémy Duval, Olivier Boissier, Mathieu Guillermin, Salima Hassas. A Multi-Agent Approach to Combine Reasoning and Learning for an Ethical Behavior. AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, May 2021, Virtual Event USA, United States. pp.13-23, 10.1145/3461702.3462515. emse-03318195

Explanation for Humans, for Machines, for Human-Machine Interactions?

International Workshop

Rémy Chaput, Amélie Cordier, Alain Mille. Explanation for Humans, for Machines, for Human-Machine Interactions?. AAAI-2021, Explainable Agency in Artificial Intelligence WS, AAAI, Feb 2021, Virtual Conference, United States. hal-03106286

Approche multi-agent combinant raisonnement et apprentissage pour un comportement éthique

National Conference

Rémy Chaput, Jérémy Duval, Olivier Boissier, Mathieu Guillermin, Salima Hassas. Approche multi-agent combinant raisonnement et apprentissage pour un comportement éthique. Journées Francophones sur les Systèmes Multi-Agents, Jun 2021, Bordeaux, France. emse-03278353

Apprentissage adaptatif de comportements éthiques

National Conference

Rémy Chaput, Olivier Boissier, Mathieu Guillermin, Salima Hassas. Apprentissage adaptatif de comportements éthiques. 28e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'2020), Jun 2020, Angers, France. hal-03012127