AfIA

Association française
pour l'Intelligence Artificielle

JFPDA

Journées Francophones sur
la Planification, la Décision et l'Apprentissage
pour la conduite de systèmes

PFIA 2021

Crédit photo : Flicr/xlibber

Table des matières

Éditorial

Journées Francophones sur la Planification, la Décision et l'Apprentissage pour la conduite de systèmes

Les Journées Francophones Planification, Décision et Apprentissage (JFPDA) ont pour but de rassembler la communauté de chercheurs francophones travaillant sur les problèmes d'apprentissage par renforcement, de la théorie du contrôle, de programmation dynamique et plus généralement dans les domaines liés à la prise de décision séquentielle sous incertitude et à la planification. La conférence JFPDA est soutenue par le Collège Représentation et Raisonnement de l'AFIA.

Nous remercions tous les membres du comité de programme pour leur travail de relecture.

François Schwarzentruber

Comité de programme

Président

— François Schwarzentruber (ENS Rennes, IRISA)

Membres

— Olivier Buffet (LORIA, INRIA)
— Alain Dutech (LORIA, INRIA)
— Humbert Fiorino (LIG, Grenoble)
— Andreas Herzig (CNRS, IRIT, Université de Toulouse)
— Jérôme Lang (CNRS LAMSADE, Université Paris-Dauphine)
— Frédéric Maris (Université Toulouse 3 Paul Sabatier, IRIT)
— Laetitia Matignon (LIRIS CNRS)
— Alexandre Niveau (GREYC, Université de Normandie)
— Damien Pellier (LIG, Grenoble)
— Sophie Pinchinat (IRISA, Rennes)
— Cédric Pralet (ONERA, Toulouse)
— Philippe Preux (Université de Lille)
— Emmanuel Rachelson (ISAE-SUPAERO)
— Régis Sabbadin (INRA)
— Abdallah Saffidine (The University of New South Wales)
— Olivier Sigaud (ISIR, UPMC)
— Florent Teichteil-Königsbuch (Airbus Central Resaerch & Technology)
— Vincent Thomas (LORIA, Nancy)
— Paul Weng (UM-SJTU Joint Institute)
— Bruno Zanuttini (GREYC, Normandie Univ. ; UNICAEN, CNRS, ENSICAEN)

Grounding Language to Autonomously-Acquired Skills via Goal Generation

Ahmed Akakzia*[1] Cédric Colas*[2] Pierre-Yves Oudeyer[2] Mohamed Chetouani[1] Olivier Sigaud[1]

[1] Sorbonne Université
[2] INRIA

ahmed.akakzia@isir.upmc.fr

Résumé

L'objectif principal de cet article est de construire des agents artificiels capables d'apprendre aussi bien en autonomie que sous l'assistance d'un tuteur humain. Pour apprendre en autonomie, ces agents doivent être capables de générer et poursuivre leurs propres buts ainsi que d'apprendre à partir de leurs propres signaux de récompense. Pour apprendre sous assistance externe, ils doivent interagir avec un tuteur et apprendre à suivre des instructions basées sur du langage naturel. Nous proposons une nouvelle architecture d'apprentissage par renforcement conditionné sur du langage : Language-Goal-Behavior *(*LGB*). Contrairement aux approches classiques,* LGB *découple l'apprentissage sensorimoteur et l'ancrage du langage grâce à une couche sémantique intermédiaire. Nous présentons* DECSTR*, une instance particulière de* LGB*.* DECSTR *est intrinsèquement motivé et doté d'une représentation spatiale basée sur des prédicats que les enfants préverbaux maîtrisent. Nous comparons* LGB *à l'approche bout-à-bout où la politique est directement basée sur le langage (*LC-RL*) et à l'approche non sémantique où la politique est conditionnée sur des buts représentant des positions 3D. Nous montrons que* LGB *permet de satisfaire les instructions langagières, d'apprendre des comportements plus diversifiés, de changer de stratégie en cas d'échec et de faciliter l'ancrage du langage.*

Mots Clef

Intelligence Artificielle, Apprentissage par Renforcement, Ancrage du Langage.

Abstract

*We are interested in the autonomous acquisition of repertoires of skills. Language-conditioned reinforcement learning (*LC-RL*) approaches are great tools in this quest, as they allow to express abstract goals as sets of constraints on the states. However, most* LC-RL *agents are not autonomous and cannot learn without external instructions and feedback. Besides, their direct language condition cannot account for the goal-directed behavior of pre-verbal infants and strongly limits the expression of behavioral diversity for a given language input. To resolve these issues, we propose a new conceptual approach to language-conditioned* RL*: the Language-Goal-Behavior architecture (*LGB*).* LGB *decouples skill learning and language grounding via an intermediate semantic representation of the world. To showcase the properties of* LGB*, we present a specific implementation called* DECSTR*.* DECSTR *is an intrinsically motivated learning agent endowed with an innate semantic representation describing spatial relations between physical objects. In a first stage (*G→B*), it freely explores its environment and targets self-generated semantic configurations. In a second stage (*L→G*), it trains a language-conditioned goal generator to generate semantic goals that match the constraints expressed in language-based inputs. We showcase the additional properties of* LGB *w.r.t. both an end-to-end* LC-RL *approach and a similar approach leveraging non-semantic, continuous intermediate representations. Intermediate semantic representations help satisfy language commands in a diversity of ways, enable strategy switching after a failure and facilitate language grounding.*

Keywords

Artificial Intelligence, Deep Reinforcement Learning, Language Grounding

1 Introduction

Developmental psychology investigates the interactions between learning and developmental processes that support the slow but extraordinary transition from the behavior of infants to the sophisticated intelligence of human adults (Piaget, 1977; Smith & Gasser, 2005). Inspired by this line of thought, the central endeavour of developmental robotics consists in shaping a set of machine learning processes able to generate a similar growth of capabilities in robots (Weng et al., 2001; Lungarella et al., 2003). In this broad context, we are more specifically interested in designing learning agents able to: 1) explore open-ended environments and grow repertoires of skills in a self-supervised way and 2) learn from a tutor via language commands.

The design of intrinsically motivated agents marked a ma-

*Equal contribution.

jor step towards these goals. The Intrinsically Motivated Goal Exploration Processes family (IMGEPs), for example, describes embodied agents that interact with their environment at the sensorimotor level and are endowed with the ability to represent and set their own goals, rewarding themselves over completion (Forestier et al., 2017). Recently, goal-conditioned reinforcement learning (GC-RL) appeared like a viable way to implement IMGEPs and target the open-ended and self-supervised acquisition of diverse skills.

Goal-conditioned RL approaches train goal-conditioned policies to target multiple goals (Kaelbling, 1993; Schaul et al., 2015). While most GC-RL approaches express goals as target features (e.g. target block positions (Andrychowicz et al., 2017), agent positions in a maze (Schaul et al., 2015) or target images (Nair et al., 2018)), recent approaches started to use language to express goals, as language can express sets of constraints on the state space (e.g. *open the red door*) in a more abstract and interpretable way (Luketina et al., 2019).

However, most GC-RL approaches – and language-based ones (LC-RL) in particular – are not intrinsically motivated and receive external instructions and rewards. The IMAGINE approach is one of the rare examples of intrinsically motivated LC-RL approaches (Colas et al., 2020). In any case, the language condition suffers from three drawbacks. 1) It couples skill learning and language grounding. Thus, it cannot account for goal-directed behaviors in pre-verbal infants (Mandler, 1999). 2) Direct conditioning limits the behavioral diversity associated to language input: a single instruction leads to a low diversity of behaviors only resulting from the stochasticity of the policy or the environment. 3) This lack of behavioral diversity prevents agents from switching strategy after a failure.

To circumvent these three limitations, one can decouple skill learning and language grounding via an intermediate innate semantic representation. On one hand, agents can learn skills by targeting configurations from the semantic representation space. On the other hand, they can learn to generate valid semantic configurations matching the constraints expressed by language instructions. This generation can be the backbone of behavioral diversity: a given sentence might correspond to a whole set of matching configurations. This is what we propose in this work.

Contributions. We propose a novel conceptual RL architecture, named LGB for Language-Goal-Behavior and pictured in Figure 1 (right). This LGB architecture enables an agent to decouple the intrinsically motivated acquisition of a repertoire of skills (Goals $\rightarrow$ Behavior) from language grounding (Language $\rightarrow$ Goals), via the use of semantic goal representation. To our knowledge, the LGB architecture is the only one to combine the following four features:

- It is intrinsically motivated: it selects its own (semantic) goals and generates its own rewards,
- It decouples skill learning from language grounding, accounting for infants learning,

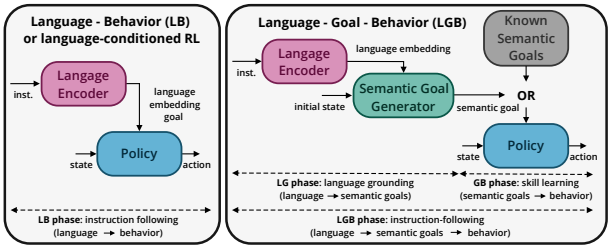


Figure 1: A standard language-conditioned RL architecture (left) and our proposed LGB architecture (right).

- It can exhibit a diversity of behaviors for any given instruction,
- It can switch strategy in case of failures.

Besides, we introduce an instance of LGB, named DECSTR for **DE**ep sets and **C**urriculum with **S**eman**T**ic goal **R**epresentations. Using DECSTR, we showcase the advantages of the conceptual decoupling idea. In the *skill learning* phase, the DECSTR agent evolves in a manipulation environment and leverages semantic representations based on predicates describing spatial relations between physical objects. These predicates are known to be used by infants from a very young age (Mandler, 2012). DECSTR autonomously learns to discover and master all reachable configurations in its semantic representation space. In the *language grounding* phase, we train a Conditional Variational Auto-Encoder (C-VAE) to generate semantic goals from language instructions. Finally, we can evaluate the agent in an *instruction-following* phase by composing the first two phases. The experimental section investigates three questions: how does DECSTR perform in the three phases? How does it compare to end-to-end LC-RL approaches? Do we need intermediate representations to be semantic? Code and videos can be found at https://sites.google.com/view/decstr/.

2 Related Work

Standard language-conditioned RL. Most approaches from the LC-RL literature define *instruction following* agents that receive external instructions and rewards (Hermann et al., 2017; Chan et al., 2019; Bahdanau et al., 2018; Cideron et al., 2019; Jiang et al., 2019; Fu et al., 2019), except the IMAGINE approach which introduced intrinsically motivated agents able to set their own goals and to imagine new ones (Colas et al., 2020). In both cases, the language-condition prevents the decoupling of language acquisition and skill learning, true behavioral diversity and efficient *strategy switching* behaviors. Our approach is different, as we can decouple language acquisition from skill learning. The language-conditioned goal generation allows behavioral diversity and strategy switching behaviors.

Goal-conditioned RL with target coordinates for block manipulation. Our proposed implementation of LGB, called DECSTR, evolves in a block manipulation domain.

Stacking blocks is one of the earliest benchmarks in artificial intelligence (e.g. Sussman (1973); Tate (1975)) and has led to many simulation and robotics studies (Deisenroth et al., 2011; Xu et al., 2018; Colas et al., 2019a). Recently, Lanier et al. (2019) and Li et al. (2019) demonstrated impressive results by stacking up to 4 and 6 blocks respectively. However, these approaches are not intrinsically motivated, involve hand-defined curriculum strategies and express goals as specific target block positions. In contrast, the DECSTR agent is intrinsically motivated, builds its own curriculum and uses semantic goal representations (symbolic or language-based) based on spatial relations between blocks.

Decoupling language acquisition and skill learning. Several works investigate the use of semantic representations to associate meanings and skills (Alomari et al., 2017; Tellex et al., 2011; Kulick et al., 2013). While the two first use semantic representations as an intermediate layer between language and skills, the third one does not use language. While DECSTR acquires skills autonomously, previous approaches all use skills that are either manually generated (Alomari et al., 2017), hand-engineered (Tellex et al., 2011) or obtained via optimal control methods (Kulick et al., 2013). Closer to us, Lynch & Sermanet (2020) also decouple skill learning from language acquisition in a goal-conditioned imitation learning paradigm by mapping both language goals and images goals to a shared representation space. However, this approach is not intrinsically motivated as it relies on a dataset of human tele-operated strategies. The deterministic merging of representations also limits the emergence of behavioral diversity and efficient strategy-switching behaviors.

3 Methods

This section presents our proposed Language-Goal-Behavior architecture (LGB) represented in Figure 1 (Section 3.1) and a particular instance of the LGB architecture called DECSTR. We first present the environment it is set in [3.2], then describe the implementations of the three modules composing any LGB architecture: 1) the semantic representation [3.3]; 2) the intrinsically motivated goal-conditioned algorithm [3.4] and 3) the language-conditioned goal generator [3.5]. We finally present how the three phases described in Figure 1 are evaluated [3.6].

3.1 The Language-Goal-Behavior Architecture

The LGB architecture is composed of three main modules. First, the *semantic representation* defines the behavioral and goal spaces of the agent. Second, the intrinsically motivated GC-RL algorithm is in charge of the skill learning phase. Third, the language-conditioned goal generator is in charge of the language grounding phase. Both phases can be combined in the instruction following phase. The three phases are respectively called G→B for Goal → Behavior, L→G for Language → Goal and L→G→B for Language → Goal → Behavior, see Figure 1 and Appendix 6. Instances of the LGB architecture should demonstrate the four properties listed in the introduction: 1) be intrinsically motivated; 2) decouple skill learning and language grounding (by design); 3) favor behavioral diversity; 4) allow strategy switching. We argue that any LGB algorithm should fulfill the following constraints. For LGB to be intrinsically motivated (1), the algorithm needs to integrate the generation and selection of semantic goals and to generate its own rewards. For LGB to demonstrate behavioral diversity and strategy switching (3, 4), the language-conditioned goal generator must efficiently model the distribution of semantic goals satisfying the constraints expressed by any language input.

3.2 Environment

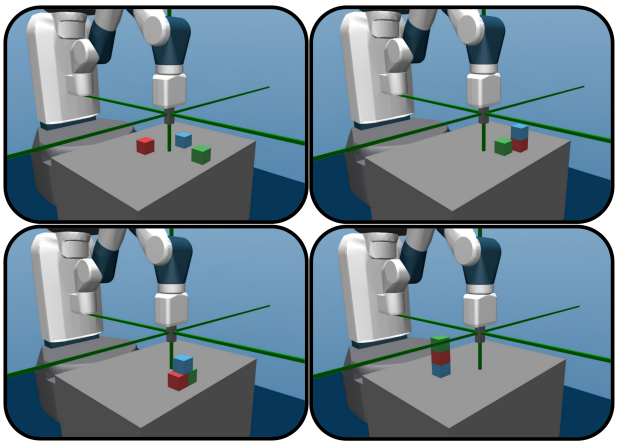


Figure 2: Example configurations. Top-right: (111000100).

The DECSTR agent evolves in the *Fetch Manipulate* environment: a robotic manipulation domain based on MUJOCO (Todorov et al., 2012) and derived from the Fetch tasks (Plappert et al., 2018), see Figure 2. Actions are 4-dimensional: 3D gripper velocities and grasping velocity. Observations include the Cartesian and angular positions and velocities of the gripper and the three blocks. Inspired by the framework of *Zone of Proximal Development* that describes how parents organize the learning environment of their children (Vygotsky, 1978), we let a social partner facilitate DECSTR's exploration by providing non-trivial initial configurations. After a first period of autonomous exploration, the social partner initializes the scene with stacks of 2 blocks 21% of times, stacks of 3 blocks 9% of times, and a block is initially put in the agent's gripper 50% of times. This help is not provided during offline evaluations.

3.3 Semantic Representation

Semantic predicates define the behavioral space. Defining the list of semantic predicates is defining the dimensions of the behavioral space explored by the agent. It replaces the traditional definition of goal spaces and their associated reward functions. We believe it is for the best, as

it does not require the engineer to fully predict all possible behaviors within that space, to know which behaviors can be achieved and which ones cannot, nor to define reward functions for each of them.

Semantic predicates in DECSTR. We assume the DECSTR agent to have access to innate semantic representations based on a list of predicates describing spatial relations between pairs of objects in the scene. We consider two of the spatial predicates infants demonstrate early in their development (Mandler, 2012): the *close* and the *above* binary predicates. These predicates are applied to all permutations of object pairs for the 3 objects we consider: 6 permutations for the *above* predicate and 3 combinations for the *close* predicate due to its order-invariance. A *semantic configuration* is the concatenation of the evaluations of these 9 predicates and represents spatial relations between objects in the scene. In the resulting semantic configuration space $\{0,1\}^9$, the agent can reach 35 physically valid configurations, including stacks of 2 or 3 blocks and pyramids, see examples in Figure 2. The binary reward function directly derives from the semantic mapping: the agent rewards itself when its current configuration c_p matches the goal configuration $c_p = g$. Appendix 7 provides formal definitions and properties of predicates and semantic configurations.

3.4 Intrinsically Motivated Goal-Conditioned Reinforcement Learning

This section describes the implementation of the intrinsically motivated goal-conditioned RL module in DECSTR. It is powered by the Soft-Actor Critic algorithm (SAC) (Haarnoja et al., 2018) that takes as input the current state, the current semantic configuration and the goal configuration, for both the critic and the policy. We use Hindsight Experience Replay (HER) to facilitate transfer between goals (Andrychowicz et al., 2017). DECSTR samples goals via its curriculum strategy, collects experience in the environment, then performs policy updates via SAC. This section describes two particularities of our RL implementation: the self-generated goal selection curriculum and the object-centered network architectures. Implementation details and hyperparameters can be found in Appendix 8.

Goal selection and curriculum learning. The DECSTR agent can only select goals among the set of semantic configurations it already experienced. We use an automatic curriculum strategy (Portelas et al., 2020) inspired from the CURIOUS algorithm (Colas et al., 2019a). The DECSTR agent tracks aggregated estimations of its *competence* (C) and *learning progress* (LP). Its selection of goals to target during data collection and goals to learn about during policy updates (via HER) is biased towards goals associated with high absolute LP and low C.

Automatic bucket generation. To facilitate robust estimation, LP is usually estimated on sets of goals with similar difficulty or similar dynamics (Forestier et al., 2017; Colas et al., 2019a). While previous works leveraged expert-defined *goal buckets*, we cluster goals based on their time of discovery, as the time of discovery is a good proxy for goal difficulty: easier goals are discovered earlier. Buckets are initially empty (no known configurations). When an episode ends in a new configuration, the $N_b = 5$ buckets are updated. Buckets are filled equally and the first buckets contain the configurations discovered earlier. Thus goals change buckets as new goals are discovered.

Tracking competence, learning progress and sampling probabilities. Regularly, the DECSTR agent evaluates itself on goal configurations sampled uniformly from the set of known ones. For each bucket, it tracks the recent history of past successes and failures when targeting the corresponding goals (last $W = 1800$ self-evaluations). C is estimated as the success rate over the most recent half of that history $\mathrm{C} = \mathrm{C}_{\mathrm{recent}}$. LP is estimated as the difference between $\mathrm{C}_{\mathrm{recent}}$ and the one evaluated over the first half of the history ($\mathrm{C}_{\mathrm{earlier}}$). This is a crude estimation of the derivative of the C curve w.r.t. time: $\mathrm{LP} = \mathrm{C}_{\mathrm{recent}} - \mathrm{C}_{\mathrm{earlier}}$. The sampling probability P_i for bucket i is:

$$P_i = \frac{(1 - C_i) * |LP_i|}{\sum_j ((1 - C_j) * |LP_j|)}.$$

In addition to the usual LP bias (Colas et al., 2019a), this formula favors lower C when LP is similar. The absolute value ensures resampling buckets whose performance decreased (e.g. forgetting).

Object-centered architecture. Instead of fully-connected or recurrent networks, DECSTR uses for the policy and critic an *object-centered architecture* similar to the ones used in Colas et al. (2020); Karch et al. (2020), adapted from Deep-Sets (Zaheer et al., 2017). For each pair of objects, a shared network independently encodes the concatenation of body and objects features and current and target semantic configurations, see Appendix Figure 6. This shared network ensures efficient transfer of skills between pairs of objects. A second inductive bias leverages the symmetry of the behavior required to achieve *above*$(o_i,\ o_j)$ and *above*$(o_j,\ o_i)$. To ensure automatic transfer between the two, we present half of the features (e.g. those based on pairs $(o_i,\ o_j)$ where $i < j$) with goals containing one side of the symmetry (all *above*$(o_i,\ o_j)$ for $i < j$) and the other half with the goals containing the other side (all *above*$(o_j,\ o_i)$ for $i < j$). As a result, the *above*$(o_i,\ o_j)$ predicates fall into the same slot of the shared network inputs as their symmetric counterparts *above*$(o_j,\ o_i)$, only with different permutations of object pairs. Goals are now of size 6: 3 *close* and 3 *above* predicates, corresponding to one side of the *above* symmetry. Skill transfer between symmetric predicates are automatically ensured. Appendix 8.1 further describes these inductive biases and our modular architecture.

3.5 Language-Conditioned Goal Generation

The language-conditioned goal generation module (LGG) is a generative model of semantic representations condi-

tioned by language inputs. It is trained to generate semantic configurations matching the agent's initial configuration and the description of a change in one object-pair relation. A training dataset is collected via interactions between a DECSTR agent trained in phase G→B and a social partner. DECSTR generates semantic goals and pursues them. For each trajectory, the social partner provides a description d of one change in objects relations from the initial configuration c_i to the final one c_f. The set of possible descriptions contains 102 sentences, each describing, in a simplified language, a positive or negative shift for one of the 9 predicates (e.g. *get red above green*). This leads to a dataset $\mathcal{D}$ of 5000 triplets: $(c_i,\ d,\ c_f)$. From this dataset, the LGG is learned using a conditional Variational Auto-Encoder (C-VAE) (Sohn et al., 2015). Inspired by the context-conditioned goal generator from Nair et al. (2019), we add an extra condition on language instruction to improve control on goal generation. The conditioning instruction is encoded by a recurrent network that is jointly trained with the VAE via a mixture of Kullback-Leibler and cross-entropy losses. Appendix 8.2 provides the list of sentences and implementation details. By repeatedly sampling the LGG, a set of goals is built for any language input. This enables skill diversity and *strategy switching*: if the agent fails, it can sample another valid goal to fulfill the instruction, effectively switching strategy. This also enables goal combination using logical functions of instructions: *and* is an intersection, *or* is an union and *not* is the complement within the known set of goals.

3.6 Evaluation of the three LGB phases

Skill learning phase G→B**:** DECSTR explores its semantic representation space, discovers achievable configurations and learns to reach them. Goal-specific performance is evaluated offline across learning as the success rate (SR) over 20 repetitions for each goal. The global performance $\overline{\text{SR}}$ is measured across either the set of 35 goals or discovery-organized buckets of goals, see Section 3.4.

Language grounding phase L→G**:** DECSTR trains the LGG to generate goals matching constraints expressed via language inputs. From a given initial configuration and a given instruction, the LGG should generate all compatible final configurations (goals) and just these. This is the source of behavioral diversity and strategy switching behaviors. To evaluate LGG, we construct a synthetic, oracle dataset $\mathcal{O}$ of triplets $(c_i, d, \mathcal{C}_f(c_i,\ d))$, where $\mathcal{C}_f(c_i,\ d)$ is the set of all final configurations compatible with $(c_i,\ d)$. On average, $\mathcal{C}_f$ in $\mathcal{O}$ contains 16.7 configurations, while the training dataset $\mathcal{D}$ only contains 3.4 (20%). We are interested in two metrics: 1) The *Precision* is the probability that a goal sampled from the LGG belongs to $\mathcal{C}_f$ (true positive / all positive); 2) The *Recall* is percentage of elements from $\mathcal{C}_f$ that were found by sampling the LGG 100 times (true positive / all true). These metrics are computed on 5 different subsets of the oracle dataset, each calling for a different type of generalization (see full lists of instructions in Appendix 8.2):

1. Pairs found in $\mathcal{D}$, except pairs removed to form the following test sets. This calls for the extrapolation of known initialization-effect pairs $(c_i,\ d)$ to new final configurations c_f ($\mathcal{D}$ contains only 20% of $\mathcal{C}_f$ on average).
2. Pairs that were removed from $\mathcal{D}$, calling for a recombination of known effects d on known c_i.
3. Pairs for which the c_i was entirely removed from $\mathcal{D}$. This calls for the transfer of known effects d on unknown c_i.
4. Pairs for which the d was entirely removed from $\mathcal{D}$. This calls for generalization in the language space, to generalize unknown effects d from related descriptions and transpose this to known c_i.
5. Pairs for which both the c_i and the d were entirely removed from $\mathcal{D}$. This calls for the generalizations 3 and 4 combined.

Instruction following phase L→G→B**:** DECSTR is instructed to modify an object relation by one of the 102 sentences. Conditioned on its current configuration and instruction, it samples a compatible goal from the LGG, then pursues it with its goal-conditioned policy. We consider three evaluation settings: 1) performing a single instruction; 2) performing a sequence of instructions without failure; 3) performing a logical combination of instructions. The *transition* setup measures the success rate of the agent when asked to perform the 102 instructions 5 times each, resetting the environment each time. In the *expression* setup, the agent is evaluated on 500 randomly generated logical functions of sentences, see the generation mechanism in Appendix 8.2. In both setups, we evaluate the performance in 1-shot (SR_1) and 5-shot (SR_5) settings. In the 5-shot setting, the agent can perform *strategy switching*, to sample new goals when previous attempts failed (without reset). In the *sequence* setup, the agent must execute 20 sequences of random instructions without reset (5-shot). We also test behavioral diversity. We ask DECSTR to follow each of the 102 instructions 50 times each and report the number of different achieved configurations.

4 Experiments

Our experimental section investigates three questions: [4.1]: How does DECSTR perform in the three phases? [4.2]: How does it compare to end-to-end language-conditioned approaches? [4.3]: Do we need intermediate representations to be semantic?

4.1 How does DECSTR perform in the three phases?

This section presents the performance of the DECSTR agent in the skill learning, language grounding, and instruction following phases.

Skill learning phase G→B**:** Figures 3, 4, 5 show that DECSTR successfully masters all reachable configurations

in its semantic representation space. Figure 3 shows the evolution of $\overline{\text{SR}}$ computed per bucket. Buckets are learned in increasing order, which confirms that the time of discovery is a good proxy for difficulty. Figure 4 reports C, LP and sampling probabilities P computed online using self-evaluations for an example agent. The agent leverages these estimations to select its goals: first focusing on the easy goals from bucket 1, it moves on towards harder and harder buckets as easier ones are mastered (low LP, high C). Figure 5 presents the results of ablation studies. Each condition removes one component of DECSTR: 1) *Flat* replaces our object-centered modular architectures by flat ones; 2) *w/o Curr.* replaces our automatic curriculum strategy by a uniform goal selection; 3) *w/o Sym.* does not use the symmetry inductive bias; 4) In *w/o SP*, the social partner does not provide non-trivial initial configurations. In the *Expert buckets* condition, the curriculum strategy is applied on expert-defined buckets, see Appendix 9.1. The full version of LGB performs on par with the *Expert buckets* oracle and outperforms significantly all its ablations. Appendix 10.3 presents more examples of learning trajectories, and dissects the evolution of bucket compositions along training.

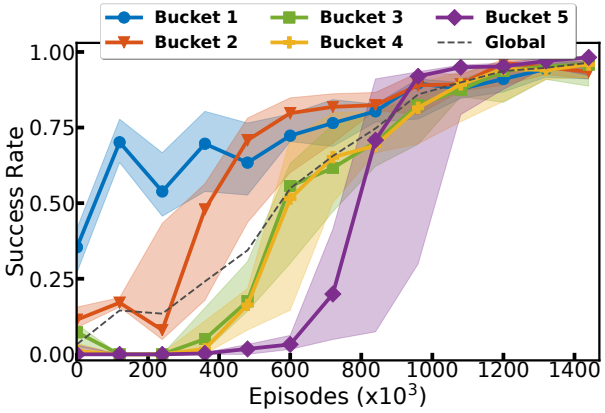


Figure 3: **Skill Learning**: $\overline{\text{SR}}$ per bucket.

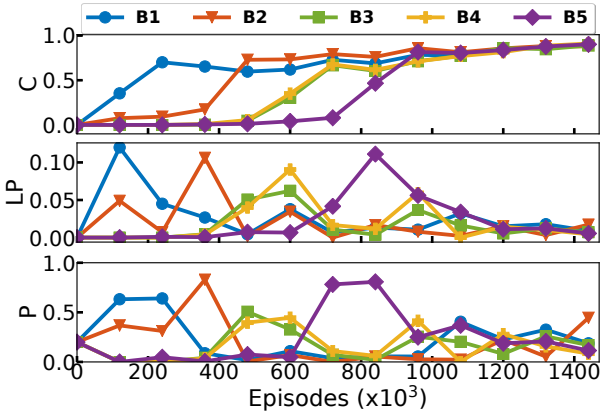


Figure 4: C, LP and P estimated by a DECSTR agent.

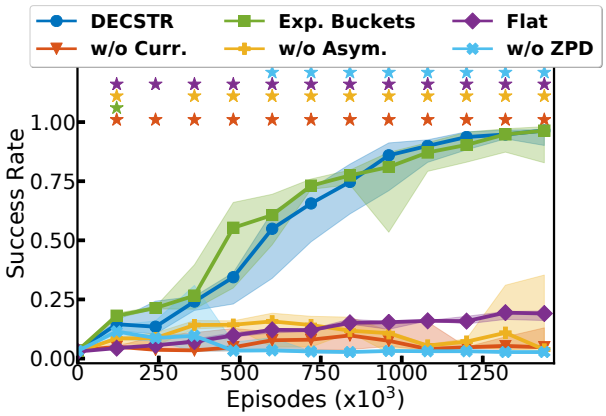


Figure 5: Ablation study. Medians and interquartile ranges over 10 seeds for DECSTR and 5 seeds for others in (a) and (c). Stars indicate significant differences to DECSTR as reported by Welch's t-tests with $\alpha = 0.05$ (Colas et al., 2019b).

Language grounding phase L→G**:** The LGG demonstrates the 5 types of generalization from Table 1. From known configurations, agents can generate more goals than they observed in training data (1, 2). They can do so from new initial configurations (3). They can generalize to new sentences (4) and even to combinations of new sentences and initial configurations (5). These results assert that DECSTR generalizes well in a variety of contexts and shows good behavioral diversity.

Table 1: L→G phase. Metrics are averaged over 10 seeds, stdev < 0.06 and 0.07 respectively.

Metrics	Test 1	Test 2	Test 3	Test 4	Test 5
Precision	0.97	0.93	0.98	0.99	0.98
Recall	0.93	0.94	0.95	0.90	0.92

Instruction following phase L→G→B**:** Table 2 presents the 1-shot and 5-shot results in the *transition* and *expression* setups. In the sequence setups, DECSTR succeeds in $L = 14.9 \pm 5.7$ successive instructions (mean±stdev over 10 seeds). These results confirm efficient language grounding. DECSTR can follow instructions or sequences of instructions and generalize to their logical combinations. Strategy switching improves performance (SR_5 - SR_1). DECSTR also demonstrates strong behavioral diversity: when asked over 10 seeds to repeat 50 times the same instruction, it achieves at least 7.8 different configurations, 15.6 on average and up to 23 depending on the instruction.

Table 2: L→G→B phase. Mean ± stdev over 10 seeds.

Metr.	Transition	Expression
SR_1	0.89 ± 0.05	0.74 ± 0.08
SR_5	0.99 ± 0.01	0.94 ± 0.06

4.2 Do we need an intermediate representation?

This section investigates the need for an intermediate semantic representation. To this end, we introduce an end-to-end LC-RL baseline directly mapping Language to Behavior (L→B) and compare its performance with DECSTR in the instruction following phase (L→G→B).

The LB baseline. To limit the introduction of confounding factors and under-tuning concerns, we base this implementation on the DECSTR code and incorporate defining features of IMAGINE, a state-of-the-art language conditioned RL agent (Colas et al., 2020). We keep the same HER mechanism, object-centered architectures and RL algorithm as DECSTR. We just replace the semantic goal space by the 102 language instructions. This baseline can be seen as an oracle version of the IMAGINE algorithm where the reward function is assumed perfect, but without the imagination mechanism.

Comparison in the instruction following phase L→B vs L→G→B: After training the LB baseline for 14K episodes, we compare its performance to DECSTR's in the instruction-following setup. In the *transition* evaluation setup, LB achieves $SR_1 = 0.76 \pm 0.001$: it always manages to move blocks close to or far from each other, but consistently fails to stack them. Adding more attempts does not help: $SR_5 = 0.76 \pm 0.001$. The LB baseline cannot be evaluated in the *expression* setup because it does not manipulate goal sets. Because it cannot stack blocks, LB only succeeds in 3.01 ± 0.43 random instructions in a row, against 14.9 for DECSTR (*sequence* setup). We then evaluate LB's diversity on the set of instructions it succeeds in. When asked to repeat 50 times the same instruction, it achieves at least 3.0 different configurations, 4.2 on average and up to 5.2 depending on the instruction against 7.8, 17.1, 23 on the same set of instructions for DECSTR. We did not observe *strategy-switching* behaviors in LB, because it either always succeeds (close/far instructions) or fails (stacks).

Conclusion. The introduction of an intermediate semantic representation helps DECSTR decouple skill learning from language grounding which, in turns, facilitates instruction-following when compared to the end-to-end language-conditioned learning of LB. This leads to improved scores in the *transition* and *sequence* setups. The direct language-conditioning of LB prevents the generalization to logical combination and leads to a reduced *diversity* in the set of mastered instructions. Decoupling thus brings significant benefits to LGB architectures.

4.3 Do we need a semantic intermediate representation?

This section investigates the need for the intermediate representation to be semantic. To this end, we introduce the LGB-C baseline that leverages continuous goal representations in place of semantic ones. We compare them on the two first phases.

The LGB-C baseline. The LGB-C baseline uses *continuous* goals expressing target block coordinates in place of semantic goals. The skill learning phase is thus equivalent to traditional goal-conditioned RL setups in block manipulation tasks (Andrychowicz et al., 2017; Colas et al., 2019a; Li et al., 2019; Lanier et al., 2019). Starting from the DECSTR algorithm, LGB-C adds a translation module that samples a set of target block coordinates matching the targeted semantic configuration which is then used as the goal input to the policy. In addition, we integrate defining features of the state-of-the-art approach from Lanier et al. (2019): non-binary rewards (+1 for each well placed block) and multi-criteria HER, see details in Appendix 9.2.

Comparison in skill learning phase G→B: The LGB-C baseline successfully learns to discover and master all 35 semantic configurations by placing the three blocks to randomly-sampled target coordinates corresponding to these configurations. It does so faster than DECSTR: $708 \cdot 10^3$ episodes to reach SR$= 95\%$, against $1238 \cdot 10^3$ for DECSTR, see Appendix Figure 8. This can be explained by the denser learning signals it gets from using HER on continuous targets instead of discrete ones. In this phase, however, the agent only learns one parameterized skill: to place blocks at their target position. It cannot build a repertoire of semantic skills because it cannot discriminate between different block configurations. Looking at the sum of the distances travelled by the blocks or the completion time, we find that DECSTR performs opportunistic goal reaching: it finds simpler configurations of the blocks which satisfy its semantic goals compared to LGB-C. Blocks move less ($\Delta_{\text{dist}} = 26 \pm 5$ cm), and goals are reached faster ($\Delta_{\text{steps}} = 13 \pm 4$, mean±std across goals with p-values $> 1.3 \cdot 10^{-5}$ and $3.2 \cdot 10^{-19}$ respectively).

Table 3: LGB-C performance in the L→G phase. Mean over 10 seeds. Stdev < 0.003 and 0.008 respectively.

Metrics	Test 1	Test 2	Test 3	Test 4	Test 5
Precision	0.66	0.78	0.39	0.0	0.0
Recall	0.05	0.02	0.06	0.0	0.0

Comparison in language grounding phase L→G: We train the LGG to generate continuous target coordinates conditioned on language inputs with a mean-squared loss and evaluate it in the same setup as DECSTR's LGG, see Table 3. Although it maintains reasonable precision in the first two testing sets, the LGG achieves low recall – i.e. diversity – on all sets. The lack of semantic representations of skills might explain the difficulty of training a language-conditioned goal generator.

Conclusion. The skill learning phase of the LGB-C baseline is competitive with the one of DECSTR. However, the poor performance in the language grounding phase prevents this baseline to perform instruction following. For this reason, and because semantic representations enable

agents to perform opportunistic goal reaching and to acquire repertoires for semantic skills, we believe the semantic representation is an essential part of the LGB architecture.

5 Discussion and Conclusion

This paper contributes LGB, a new conceptual RL architecture which introduces an intermediate semantic representation to decouple sensorimotor learning from language grounding. To demonstrate its benefits, we present DECSTR, a learning agent that discovers and masters all reachable configurations in a manipulation domain from a set of relational spatial primitives, before undertaking an efficient language grounding phase. This was made possible by the use of object-centered inductive biases, a new form of automatic curriculum learning and a novel language-conditioned goal generation module. Note that our main contribution is in the conceptual approach, DECSTR being only an instance to showcase its benefits. We believe that this approach could benefit from any improvement in GC-RL (for skill learning) or generative models (for language grounding).

Semantic representations. Results have shown that using predicate-based representations was sufficient for DECSTR to efficiently learn abstract goals in an opportunistic manner. The proposed semantic configurations showcase promising properties: 1) they reduce the complexity of block manipulation where most effective works rely on a heavy hand-crafted curriculum (Li et al., 2019; Lanier et al., 2019) and a specific curiosity mechanism (Li et al., 2019); 2) they facilitate the grounding of language into skills and 3) they enable decoupling skill learning from language grounding, as observed in infants (Piaget, 1977). The set of semantic predicates is, of course, domain-dependent as it characterizes the space of behaviors that the agent can explore. However, we believe it is easier and requires less domain knowledge to define the set of predicates, i.e. the dimensions of the space of potential goals, than it is to craft a list of goals and their associated reward functions.

A new approach to language grounding. The approach proposed here is the first simultaneously enabling to decouple skill learning from language grounding and fostering a diversity of possible behaviors for given instructions. Indeed, while an instruction following agent trained on goals like *put red close_to green* would just push the red block towards the green one, our agent can generate many matching goal configurations. It could build a pyramid, make a blue-green-red pile or target a dozen other compatible configurations. This enables it to *switch strategy*, to find alternative approaches to satisfy a same instruction when first attempts failed. Our goal generation module can also generalize to new sentences or transpose instructed transformations to unknown initial configurations. Finally, with the goal generation module, the agent can deal with any logical expression made of instructions by combining generated goal sets. It would be of interest to simultaneously perform language grounding and skill learning, which would result in “overlapping waves" of sensorimotor and linguistic development (Siegler, 1998).

Semantic configurations of variable size. Considering a constant number of blocks and, thus, fixed-size configuration spaces is a current limit of DECSTR. Future implementations of LGB may handle inputs of variable sizes by leveraging Graph Neural Networks as in Li et al. (2019). Corresponding semantic configurations could be represented as a set of vectors, each encoding information about a predicate and the objects it applies to. These representations could be handled by Deep Sets (Zaheer et al., 2017). This would allow to target partial sets of predicates that would not need to characterize all relations between all objects, facilitating scalability.

Conclusion In this work, we have shown that introducing abstract goals based on relational predicates that are well understood by humans can serve as a pivotal representation between skill learning and interaction with a user through language. Here, the role of the social partner was limited to: 1) helping the agent to experience non-trivial configurations and 2) describing the agent's behavior in a simplified language. In the future, we intend to study more intertwined skill learning and language grounding phases, making it possible to the social partner to teach the agent during skill acquisition.

Acknowledgments

This work was performed using HPC resources from GENCI-IDRIS (Grant 20XX-AP010611667), the MeSU platform at Sorbonne-Université and the PlaFRIM experimental testbed. Cédric Colas is partly funded by the French Ministère des Armées - Direction Générale de l'Armement.

References

Muhannad Alomari, Paul Duckworth, David C Hogg, and Anthony G Cohn. Natural language acquisition and grounding for embodied robotic systems. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. *arXiv preprint arXiv:1707.01495*, 2017.

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Arian Hosseini, Pushmeet Kohli, and Edward Grefenstette. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*, 2018.

Harris Chan, Yuhuai Wu, Jamie Kiros, Sanja Fidler, and Jimmy Ba. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *arXiv preprint arXiv:1902.04546*, 2019.

Geoffrey Cideron, Mathieu Seurin, Florian Strub, and Olivier Pietquin. Self-educated language agent with hindsight experience replay for instruction following. *arXiv preprint arXiv:1910.09451*, 2019.

Cédric Colas, Pierre-Yves Oudeyer, Olivier Sigaud, Pierre Fournier, and Mohamed Chetouani. CURIOUS: Intrinsically motivated multi-task, multi-goal reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 1331–1340, 2019a.

Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms. *arXiv preprint arXiv:1904.06979*, 2019b.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity-driven exploration. *arXiv preprint arXiv:2002.09253*, 2020.

Marc Peter Deisenroth, Carl Edward Rasmussen, and Dieter Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. *Robotics: Science and Systems VII*, pp. 57–64, 2011.

Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3D world. *arXiv preprint arXiv:1706.06551*, 2017.

Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 9414–9426, 2019.

Leslie Pack Kaelbling. Learning to achieve goals. In *International Joint Conference on Artificial Intelligence*, pp. 1094–1099, 1993.

Tristan Karch, Cédric Colas, Laetitia Teodorescu, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Deep sets for generalization in RL. *arXiv preprint arXiv:2003.09443*, 2020.

Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. Active learning for teaching a robot grounded relational symbols. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

John B. Lanier, Stephen McAleer, and Pierre Baldi. Curiosity-driven multi-criteria hindsight experience replay. *CoRR*, abs/1906.03710, 2019. URL `http://arxiv.org/abs/1906.03710`.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. *arXiv preprint arXiv:1912.11032*, 2019.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*, 2019.

Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, 2003.

Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

Jean M. Mandler. Preverbal representation and language. *Language and space*, pp. 365, 1999.

Jean M Mandler. On the spatial foundations of the conceptual system and its enrichment. *Cognitive science*, 36 (3):421–451, 2012.

Ashvin Nair, Shikhar Bahl, Alexander Khazatsky, Vitchyr Pong, Glen Berseth, and Sergey Levine. Contextual imagined goals for self-supervised robotic learning. *arXiv preprint arXiv:1910.11670*, 2019.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9191–9200, 2018.

Jean Piaget. *The development of thought: Equilibration of cognitive structures.* Viking, 1977. (Trans A. Rosin).

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

Robert S Siegler. *Emerging minds: The process of change in children's thinking*. Oxford University Press, 1998.

Linda Smith and Michael Gasser. The development of embodied cognition: Six lessons from babies. *Artificial life*, 11(1-2):13–29, 2005.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Gerald J. Sussman. A computational model of skill acquisition. Technical report, HIT Technical Report AI TR-297, 1973.

Austin Tate. Interacting goals and their use. In *International Joint Conference on Artificial Intelligence*, volume 10, pp. 215–218, 1975.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI magazine*, 32(4):64–76, 2011.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

L. S. Vygotsky. Tool and Symbol in Child Development. In *Mind in Society*, chapter Tool and Symbol in Child Development, pp. 19–30. Harvard University Press, 1978. ISBN 0674576292. doi: 10.2307/j.ctvjf9vz4.6.

J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Autonomous mental development by robots and animals. *Science*, 291(5504): 599–600, 2001.

Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

Supplementary Material

6 LGB pseudo-code

Algorithm 1 and 2 present the high-level pseudo-code of any algorithm following the LGB architecture for each of the three phases.

Algorithm 1 LGB architecture
G→B phase

▷ Goal → Behavior phase
Require Env E
Initialize policy Π, goal sampler G_s, buffer B
loop
  $g \leftarrow G_s$.sample()
  $(s, a, s', g, c_p, c'_p)_{\text{traj}} \leftarrow E$.rollout($g$)
  G_s.update(c_p^T)
  B.update($(s, a, s', g, c_p, c'_p)_{\text{traj}}$)
  Π.update(B)
return Π, G_s

Algorithm 2 LGB architecture
L→G and L→G→B phases

▷ Language → Goal phase
Require Π, E, G_s, social partner SP
Initialize language goal generator LGG
dataset $\leftarrow SP$.interact(E, Π, G_s)
LGG.update(dataset)
return LGG
▷ Language → Behavior phase
Require E, Π, LGG, SP
loop
  instr. $\leftarrow SP$.listen()
  loop ▷ Strategy switching loop
    $g \leftarrow LGG$.sample(instr., c^0)
    $c_p^T \leftarrow E$.rollout(g)
    if $g == c_p^T$ **then break**

7 Semantic predicates and application to fetch manipulate

In this paper, we restrict the semantic representations to the use of the *close* and *above* binary predicates applied to $M = 3$ objects. The resulting semantic configurations are formed by:

$$F = [c(o_1, o_2),\ c(o_1, o_3),\ c(o_2, o_3),\ a(o_1, o_2),\ a(o_2, o_1),\ a(o_1, o_3),\ a(o_3, o_1),\ a(o_2, o_3),\ a(o_3, o_2)],$$

where *c()* and *a()* refer to the *close* and *above* predicates respectively and $(o_1,\ o_2,\ o_3)$ are the red, green and blue blocks respectively.

Symmetry and asymmetry of *close* and *above* predicates. We consider objects o_1 and o_2.

- *close* is symmetric: "o_1 is **close** to o_2" $\Leftrightarrow$ "o_2 is **close** to o_1". The corresponding semantic mapping function is based on the Euclidean distance, which is symmetric.
- *above* is asymmetric: "o_1 is **above** o_2" $\Rightarrow$ **not** "o_2 is **above** o_1". The corresponding semantic mapping function evaluates the sign of the difference of the object Z-axis coordinates.

8 The DECSTR algorithm

8.1 Intrinsically Motivated Goal-Conditioned RL

Overview. Algorithm 3 presents the pseudo-code of the sensorimotor learning phase (G→B) of DECSTR. It alternates between two steps:

- **Data acquisition.** A DECSTR agent has no prior on the set of reachable semantic configurations. Its first goal is sampled uniformly from the semantic configuration space. Using this goal, it starts interacting with its environment, generating trajectories of sensory states s, actions a and configurations c_p. The last configuration c_p^T achieved in the episode after T time steps is considered stable and is added to the set of reachable configurations. As it interacts with the environment, the agent explores the configuration space, discovers reachable configurations and selects new targets.
- **Internal models updates.** A DECSTR agent updates two models: its curriculum strategy and its policy. The curriculum strategy can be seen as an active goal sampler. It biases the selection of goals to target and goals to learn about. The policy is the module controlling the agent's behavior and is updated via RL.

Policy updates with a goal-conditioned Soft Actor-Critic. Readers familiar with Markov Decision Process and the use of SAC and HER algorithms can skip this paragraph.

We want the DECSTR agent to explore a semantic configuration space and master reachable configurations in it. We frame this problem as a goal-conditioned MDP (Schaul et al., 2015): $\mathcal{M} = (\mathcal{S}, \mathcal{G}_p, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, where the state space $\mathcal{S}$ is the usual sensory space augmented with the configuration space $\mathcal{C}_p$, the goal space $\mathcal{G}_p$ is equal to the configuration space $\mathcal{G}_p = \mathcal{C}_p$, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the unknown transition probability, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \{0, 1\}$ is a sparse reward function and $\gamma \in [0, 1]$ is the discount factor.

Policy updates are performed with Soft Actor-Critic (SAC) (Haarnoja et al., 2018), a state-of-the-art off-policy actor-critic algorithm. We also use Hindsight Experience Replay (HER) (Andrychowicz et al., 2017). This mechanism enables agents to learn from failures by reinterpreting past trajectories in the light of goals different from the ones originally targeted. HER was designed for continuous goal spaces, but can be directly transposed to discrete goals (Colas et al., 2019a). In our setting, we simply replace the originally targeted goal configuration by the currently achieved configuration in the transitions fed to SAC. We also use our automatic curriculum strategy: the LP-C-based probabilities are used to sample goals to learn about. When a goal g is sampled, we search the experience buffer for the collection of episodes that ended in the configuration $c_p = g$. From these episodes, we sample a transition uniformly. The HER mechanism substitutes the original goal with one of the configurations achieved later in the trajectory. This substitute g has high chances of being the sampled one. At least, it is a configuration on the path towards this goal, as it is sampled from a trajectory leading to it. The HER mechanism is thus biased towards goals sampled by the agent.

Algorithm 3 DECSTR: sensorimotor phase G→B.

Require: env E, # buckets N_b, # episodes before biased init. n_{unb}, self-evaluation probability $p_{\text{self\_eval}}$, noise function $\sigma()$
Initialize: policy Π, buffer B, goal sampler G_s, bucket sampling probabilities p_b, language module LGG.
loop
  self_eval $\leftarrow$ random() $< p_{\text{self\_eval}}$ ▷ If $True$ then evaluate competence
  $g \leftarrow G_s$.sample(self_eval, p_b)
  biased_init $\leftarrow$ $epoch < n_{\text{unb}}$ ▷ Bias initialization only after n_{unb} epochs
  $s^0, c_p^0 \leftarrow E$.reset($biased\_init$) ▷ c_0: Initial semantic configuration
  for $t = 1 : T$ **do**
    $a^t \leftarrow policy(s^t, c^t, g)$
    if not self_eval **then**
      $a^t \leftarrow a^t + \sigma()$
    $s^{t+1}, c_p^{t+1} \leftarrow E$.step($a^t$)
  episode $\leftarrow (s, c, a, s', c')$
  G_s.update(c^T)
  B.update(episode)
  $g \leftarrow G_s$.sample(p_b)
  batch $\leftarrow B$.sample(g)
  Π.update(batch)
  if self_eval **then**
    $p_b \leftarrow G_s$.update_LP()

Object-Centered Inductive Biases. In the proposed *Fetch Manipulate* environment, the three blocks share the same set of attributes (position, velocity, color identifier). Thus, it is natural to encode a *relational inductive bias* in our architecture. The behavior with respect to a pair of objects should be independent from the position of the objects in the inputs. The architecture used for the policy is depicted in Figure 6.

A shared network (NN_{shared}) encodes the concatenation of:

1) agent's body features; 2) object pair features; 3) current configuration (c_p) and 4) current goal g. This is done independently for all object pairs. No matter the location of the features of the object pair in the initial observations, this shared network ensures that the same behavior will be performed, thus skills are transferred between object pairs. A sum is then used to aggregate these outputs, before a final network (NN_{policy}) maps the aggregation to actions a. The critic follows the same architecture, where a final network NN_{critic} maps the aggregation to an action-value Q. Parallel encoding of each pair-specific inputs can be seen as different modules trying to reach the goal by only seeing these pair-specific inputs. The intuition is that modules dealing with the pair that should be acted upon to reach the goal will supersede others in the sum aggregation.

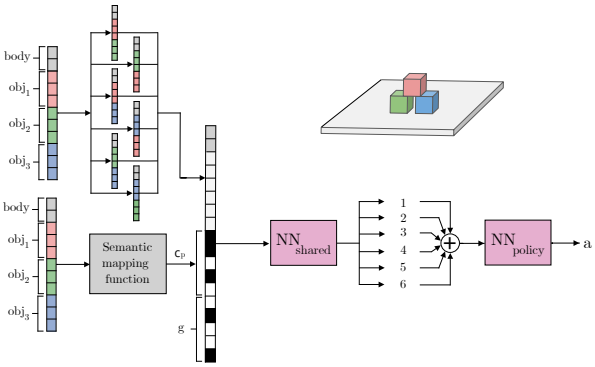


Figure 6: Object-centered modular architecture for the policy.

Although in principle our architecture could work with combinations of objects (3 modules), we found permutations to work better in practice (6 modules). With combinations, the shared network would need to learn to put block A on block B to achieve a predicate *above*$(o_i,\ o_j)$, and would need to learn the reverse behavior (put B on A) to achieve the symmetric predicate *above*$(o_j,\ o_i)$. With permutations, the shared network can simply learn one of these behaviors (e.g. A on B). Considering the predicate *above*$(o_A,\ o_B)$, at least one of the modules has objects organized so that this behavior is the good one: if the permutation $(o_B,\ o_A)$ is not the right one, permutation $(o_A,\ o_B)$ is. The symmetry bias is explained in Section 3.4. It leverages the symmetry of the behaviors required to achieve the predicates *above*$(o_i,\ o_j)$ and *above*$(o_j,\ o_i)$. As a result, the two goal configurations are:

$$g_1 = [c(o_1, o_2),\ c(o_1, o_3),\ c(o_2, o_3),\ a(o_1, o_2),\ a(o_1, o_3),\ a(o_2, o_3)],$$
$$g_2 = [c(o_1, o_2),\ c(o_1, o_3),\ c(o_2, o_3),\ a(o_2, o_1),\ a(o_3, o_1),\ a(o_3, o_2)],$$

where g_1 is used in association with object permutations $(o_i,\ o_j)$ with $i\ <\ j$ and g_2 is used in association with object permutations $(o_j,\ o_i)$ with $i\ <\ j$. As a result, the shared network automatically ensures transfer between predicates based on symmetric behaviors.

Implementation details. This part includes details necessary to reproduce our results. The code is available at https://sites.google.com/view/decstr/.
Parallel implementation of SAC-HER*:* We use a parallel implementation of SAC (Haarnoja et al., 2018). Each of the 24 parallel worker maintains its own replay buffer of size 10^6 and performs its own updates. Updates are summed over the 24 actors and the updated network are broadcast to all workers. Each worker alternates between 2 episodes of data collection and 30 updates with batch size 256. To form an epoch, this cycle is repeated 50 times and followed by the offline evaluation of the agent on each reachable goal. An epoch is thus made of $50 \times 2 \times 24 = 2400$ episodes.
Goal sampler updates: The agent performs self-evaluations with probability $self\_eval = 0.1$. During these runs, the agent targets uniformly sampled discovered configurations without exploration noise. This enables the agent to self-evaluate on each goal. Goals are organized into buckets. Main Section 3.4 presents our automatic bucket generation mechanism. Once buckets are formed, we compute C, LP and P, based on windows of the past $W = 1800$ self-evaluation interactions for each bucket.
Modular architecture: The shared network of our modular architecture NN_{shared} is a 1-hidden layer network of hidden size 256. After all pair-specific inputs have been encoded through this module, their output (of size 84) are summed. The sum is then passed through a final network with a hidden layer of size 256 to compute the final actions (policy) or action-values (critic). All networks use $ReLU$ activations and the Xavier initialization. We use Adam optimizers, with learning rates 10^{-3}. The list of hyperparameters is provided in https://github.com/akakzia/decstr.

Computing resources. The sensorimotor learning experiments contain 8 conditions: 2 of 10 seeds and 6 of 5 seeds. Each run leverages 24 cpus (24 actors) for about 72h for a total of 9.8 cpu years. Experiments presented in this paper requires machines with at least 24 cpu cores. The language grounding phase runs on a single cpu and trains in a few minutes.

8.2 Language-conditioned goal generator

Language-Conditioned Goal Generator Training. We use a conditional Variational Auto-Encoder (C-VAE) (Sohn et al., 2015). Conditioned on the initial configuration and a sentence describing the expected transformation of one object relation, it generates compatible goal configurations. After the first phase of goal-directed sensorimotor training, the agent interacts with a hard-coded social partner as described in Main Section 3. From these interactions, we obtain a dataset of 5000 triplets: initial configuration, final configuration and sentence describing one change of predicate from the initial to the final configuration. The list of sentences used by the synthetic social partner is provided in Table 4. Note that *red*, *green* and *blue* refer to objects $o_1,\ o_2,\ o_3$ respectively.

Content of test sets. We describe the 5 test sets:

1. Test set 1 is made of input pairs $(c_i,\ s)$ from the training set, but tests the coverage of all compatible final configurations $\mathcal{C}_f$, 80% of which are not found in the training set. In that sense, it is partly a test set.
2. Test set 2 contains two input pairs: {[0 1 0 0 0 0 0 0 0 0], *put blue close_to green*} and {[0 0 1 0 0 0 0 0 0 0], *put green below red*} corresponding to 7 and 24 compatible final configurations respectively.
3. Test set 3 corresponds to all pairs including the initial configuration $c_i = [1\,1\,0\,0\,0\,0\,0\,0\,0\,0]$ (29 pairs), with an average of 13 compatible final configurations.
4. Test set 4 corresponds to all pairs including one of the sentences *put green on_top_of red* and *put blue far_from red*, i.e. 20 pairs with an average of 9.5 compatible final configurations.
5. Test set 5 is all pairs that include both the initial configuration of test set 3 and one of the sentences of test set 4, i.e. 2 pairs with 6 and 13 compatible goals respectively. Note that pairs of set 5 are removed from sets 3 and 4.

Table 4: List of instructions. Each of them specifies a shift of one predicate, either from false to true ($0 \rightarrow 1$) or true to false ($1 \rightarrow 0$). **block A** and **block B** represent two different blocks from {red, blue, green}.

Transition type	Sentences
Close $0 \rightarrow 1$	*Put **A** close_to **B***,
	*Bring **A** and **B** together*,
($\times 3$)	*Get **A** and **B** close_from_each_other*,
	*Get **A** close_to **B***.
Close $1 \rightarrow 0$	*Put **A** far_from **B***,
	*Get **A** far_from **B***,
($\times 3$)	*Get **A** and **B** far_from_each_other*,
	*Bring **A** and **B** apart*,
Above $0 \rightarrow 1$	*Put **A** above **B***,
	*Put **A** on_top_of **B***,
($\times 6$)	*Put **B** under **A***,
	*Put **B** below **A***.
Above $1 \rightarrow 0$	*Remove **A** from_above **B***,
	*Remove **A** from **B***,
($\times 6$)	*Remove **B** from_below **A***,
	*Put **B** and **A** on_the_same_plane*,
	*Put **A** and **B** on_the_same_plane*.

Testing on logical expressions of instructions. To evaluate DECSTR on logical functions of instructions, we generate three types of expressions:

1. 100 instructions of the form "A and B" where A and B are basic instructions corresponding to shifts of the form *above* $0 \rightarrow 1$ (see Table 4). These intersections correspond to stacks of 3 or pyramids.
2. 200 instructions of the form "A and B" where A and B are *above* and *close* instructions respectively. B can be replaced by "not B" with probability 0.5.
3. 200 instructions of the form "(A and B) or (C and D))", where A, B, C, D are basic instructions: A and C are *above* instructions while B and D are *close* instructions. Here also, any instruction can be replaced by its negation with probability 0.5.

Implementation details. The encoder is a fully-connected neural network with two layers of size 128 and *ReLU* activations. It takes as input the concatenation of the final binary configuration and its two conditions: the initial binary configuration and an embedding of the NL sentence. The NL sentence is embedded with an recurrent network with embedding size 100, *tanh* non-linearities and biases. The encoder outputs the mean and log-variance of the latent distribution of size 27. The decoder is also a fully-connected network with two hidden layers of size 128 and *ReLU* activations. It takes as input the latent code z and the same conditions as the encoder. As it generates binary vectors, the last layer uses *sigmoid* activations. We train the architecture with a mixture of Kullback-Leibler divergence loss (KD_{loss}) w.r.t a standard Gaussian prior and a binary Cross-Entropy loss (BCE_{loss}). The combined loss is $BCE_{\text{loss}} + \beta \times KD_{\text{loss}}$ with $\beta = 0.6$. We use an Adam optimizer, a learning rate of 5×10^{-4}, a batch size of 128 and optimize for 150 epochs. As training is fast (≈ 2 min on a single cpu), we conducted a quick hyperparameter search over β, layer sizes, learning rates and latent sizes (see Table 5). We found robust results for various layer sizes, various β below 1. and latent sizes above 9.

Table 5: LGG hyperparameter search. In bold are the selected hyperparameters.

Hyperparam.	Values.
β	[0.5, **0.6**, 0.7, 0.8, 0.9, 1.]
layers size	[**128**, 256]
learning rate	[0.01, **0.005**, 0.001]
latent sizes	[9, 18, **27**]

9 Baselines and oracle

The language-conditioned LB baseline is fully described in the main document.

9.1 Expert buckets oracle

In the EXPERT BUCKETS oracle, the automatic bucket generation of DECSTR is replaced with an expert-predefined set of buckets using *a priori* measures of similarity and difficulty. To define these buckets, one needs prior knowledge of the set of unreachable configurations, which are ruled out. The 5 predefined buckets contain all configurations characterized by:

- Bucket 1: a single *close* relation between a pair of objects and no *above* relations (4 configurations).
- Bucket 2: 2 or 3 *close* relations and no *above* relations (4 configurations).

- Bucket 3: 1 stack of 2 blocks and a third block that is either away or close to the base, but is not close to the top of the stack (12 configurations).
- Bucket 4: 1 stack of 2 blocks and the third block close to the stack, as well as pyramid configurations (9 configurations).
- Bucket 5: stacks of 3 blocks (6 configurations).

These buckets are the only difference between the EXPERT BUCKETS baseline and DECSTR.

9.2 LGB-C baseline

The LGB-C baseline represent goals not as semantic configurations but as particular 3D targets positions for each block, as defined for example in Lanier et al. (2019) and Li et al. (2019). The goal vector size is also 9 and contains the 3D target coordinates of the three blocks. This baselines also implements decoupling and, thus, can be compared to DECSTR in the three phases. We keep as many modules as possible common with DECSTR to minimize the amount of confounding factors and reduce the *under-fitting* bias. The goal selection is taken from DECSTR, but converts semantic configuration into specific randomly-sampled target coordinates for the blocks, see Figure 7. The agent is not conditioned on its current semantic configuration nor its semantic goal configuration. For this reason, we do not apply the symmetry bias. The binary reward is positive when the maximal distance between a block and its target position is below 5 cm, i.e. the size of a block (similar to (Andrychowicz et al., 2017)). To make this baseline competitive, we integrate methods from a state of the art block manipulation algorithm (Lanier et al., 2019). The agent receives positive rewards of 1, 2, 3 when the corresponding number of blocks are well placed. We also introduce the multi-criteria HER from Lanier et al. (2019). Finally, we add an additional object-centered inductive bias by only considering, for each Deep Sets module, the 3D target positions of the corresponding pair.That is, for each object pair, we ignore the 3D positions of the remaining object, yielding to a vector of size 6. Language grounding is based on a C-VAE similar to the one used by DECSTR. We only replace the cross-entropy loss by a mean-squared loss due to the continuous nature of the target goal coordinates. We use the exact same training and testing sets as with semantic goals.

10 Additional results

10.1 Comparison DECSTR - LGB-C in skill learning phase

Figure 8 presents the average success rate over the 35 valid configurations during the skill learning phase for DECSTR and the LGB-C baseline. Because LGB-C cannot pursue semantic goals as such, we randomly sample a specific instance of this semantic goal: target block coordinates that satisfy the constraints expressed by it. Because LGB-C is not aware of the original semantic goal, we cannot measure success as the ability to achieve it. Instead, *success*

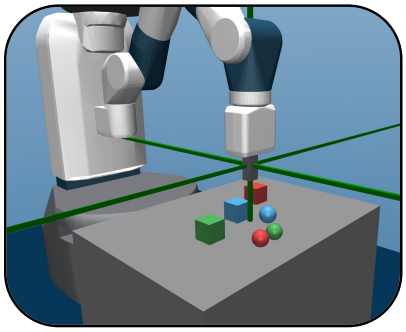


Figure 7: The LGB-C baseline samples target positions for each block (example for a pyramid here).

is defined as the achievement of the corresponding specific goal: bringing blocks to their respective targets within an error margin of 5 cm each. In short, DECSTR targets semantic goals and is evaluated on its ability to reach them. LGB-C targets specific goals and is evaluated on its ability to reach them. These two measures do not match exactly. Indeed, LGB-C sometimes achieves its specific goal but, because of the error margins, does not achieve the original semantic goal.

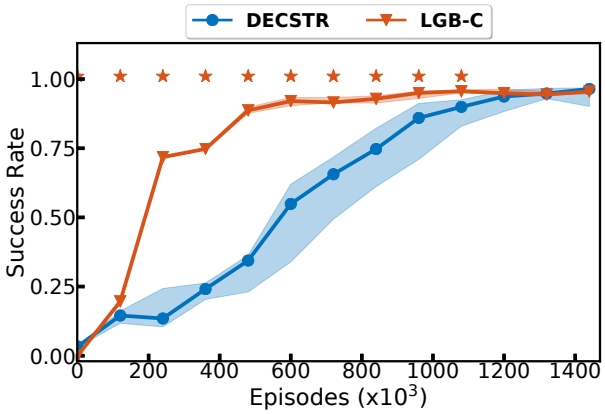


Figure 8: Comparison DECSTR and LGB-C in the skill learning phase.

10.2 Automatic bucket generation.

Figure 9 depicts the evolution of the content of buckets along training (epochs 1, 50 and 100). Each pie chart corresponds to a reachable configuration and represents the distribution of configurations into buckets across 10 different seeds. Blue, orange, green, yellow, purple represent buckets 1 to 5 respectively and grey are undiscovered configurations. At each moment, the discovered configurations are equally spread over the 5 buckets. A given configuration may thus change bucket as new configurations are discovered, so that the ones discovered earlier are assigned buckets with lower indexes. Goals are organized by their bucket assignments in the *Expert Buckets* condition (from

top to bottom).
After the first epoch (left), DECSTR has discovered all configurations from the expert buckets 1 and 2, and some runs have discovered a few other configurations. After 50 epochs, more configurations have been discovered but they are not always the same across runs. Finally, after 100 epochs, all configurations are found. Buckets are then steady and can be compared to expert-defined buckets. It seems that easier goals (top-most group) are discovered first and assigned in the first-easy buckets (blue and orange). Hardest configurations (stacks of 3, bottom-most group) seem to be discovered last and assigned the last-hardest bucket (purple). In between, different runs show different compositions, which are not always aligned with expert-defined buckets. Goals from expert-defined buckets 3 and 4 (third and fourth group from the top) seem to be attributed different automatic buckets in different runs. This means that they are discovered in different orders depending on the runs. In summary, easier and harder goals from expert buckets 1 - 2 and 5 respectively seem to be well detected by our automatic bucket generations. Goals in medium-level expected difficulty as defined by expert buckets seem not to show any significant difference in difficulty for our agents.

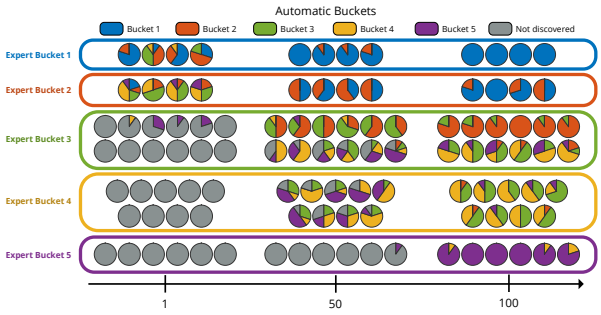


Figure 9: Evolution of the content of buckets from automatic bucket generation: epoch 1 (2400 episodes, left), 50 (middle) and 100 (right). Each pie chart corresponds to one of the 35 valid configurations. It represents the distribution of the bucket attributions of that configuration across 10 runs. Blue, orange, green, yellow, purple represent automatically generated buckets 1 to 5 respectively (increasing order of difficulty) and grey represents undiscovered configurations. Goals are organized according to their expert bucket attributions in the *Expert Buckets* condition (top-bottom organization).

10.3 DECSTR learning trajectories

Figure 10 shows the evolution of internal estimations of the competence C, the learning progress LP and the associated sampling probabilities P. Note that these metrics are computed online by DECSTR, as it self-evaluates on random discovered configurations. Learning trajectories seem to be uniform across different runs, and buckets are learned in increasing order. This confirms that the time of discovery is a good proxy for goal difficulty. In that case, configurations discovered first end up in the lower index buckets and are indeed learned first. Note that a failing automatic bucket generation would assign goals to random buckets. This would result in uniform measures of learning progress across different buckets, which would be equivalent to uniform goal sampling. As Main Figure 5 shows, DECSTR performs much better than the *random goals* conditions. This proves that our automatic bucket algorithm generates useful goal clustering.

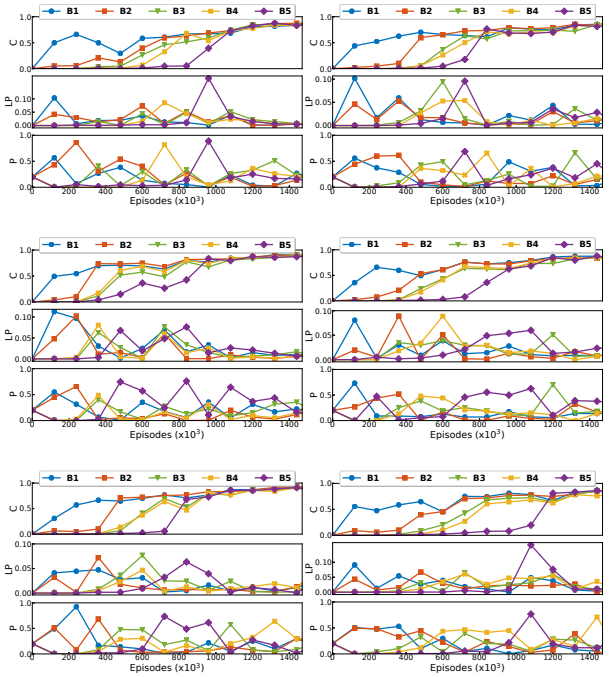


Figure 10: Learning trajectories of 6 DECSTR agents.

HSVI pour zs-POSG usant de propriétés de convexité, concavité, et Lipschitz-continuité

Aurélien Delage[1] Olivier Buffet[2] Jilles Dibangoye[1]

[1] Univ Lyon, INSA Lyon, INRIA, CITI, F-69621 Villeurbanne, France
[2] Université de Lorraine, INRIA, CNRS, LORIA, F-54000 Nancy, France

prénom.nom@inria.fr

Résumé

La résolution d'un jeu stochastique partiellement observable à 2 joueurs et somme nulle (zs-POSG) repose typiquement sur sa transformation en un jeu en forme extensive, perdant ainsi de l'information structurelle contenue dans la représentation originale. Nous évitons une telle perte en transformant le problème plutôt en un jeu de Markov sur les états d'occupation, ce qui permet de tirer parti d'approches de l'état de l'art pour les processus de décision markoviens partiellement observables, les POMDP décentralisés, et un certain nombre de sous-classes de zs-POSG. Pour appliquer le principe d'optimalité de Bellman dans ce cadre, nous (i) exhibons des propriétés de continuité originales de la fonction de valeur optimale ; (ii) dérivons des approximateurs encadrants à base de points ; et (iii) proposons des opérateurs de mise à jour efficaces reposant sur la programmation linéaire. Une variante de l'algorithm HSVI de SMITH *et* SIMMONS *[23] est construite sur la base de ces idées et nous prouvons sa convergence vers une solution ϵ-optimale avant de présenter des résultats expérimentaux.*

Mots Clef

POSG, jeu stochastique partiellement observable, principe d'optimalité de Bellman, Heuristic Search Value Iteration, jeu à information imparfaite.

Abstract

Solving a 2-player zero-sum partially observable stochastic game (zs-POSG) typically relies on turning it into an extensive-form game, thus losing structural information contained in the original representation. We prevent such a loss by turning the problem instead into an occupancy Markov game, which allows building on state-of-the-art approaches for partially observable Markov decision processes, decentralized POMDPs, and a number of subclasses of zs-POSGs. To apply Bellman's optimality principle in this setting, we (i) exhibit novel continuity properties of the optimal value function ; (ii) derive point-based bounding approximators ; and (iii) propose efficient backup operators based on linear programming. A variant of SMITH *et* SIMMONS*'s (2005) HSVI algorithm is built on these ideas and we prove its finite-time convergence to an ϵ-optimal solution before presenting experimental results.*

Keywords

POSG, partially observable stochastic game, Bellman's optimality principle, Heuristic Search Value Iteration, imperfect information game.

1 Introduction

Résoudre des jeux séquentiels à information imparfaite est un sujet difficile avec de nombreuses applications, du jeu du Poker [17] aux jeux de sécurité [2]. Nous nous concentrons sur les jeux stochastiques partiellement observables à 2 joueurs et somme nulle ((2p) zs-POSG), une sous-classe importante venant avec des représentations de problèmes qui sont compactes et permettent d'exploiter leur structure (par exemple, pour dériver des relaxations). Les principales techniques de résolution des zs-POSG requièrent de les transformer en jeux à forme extensive et à somme nulle (zs-EFG) [20]. Ensuite, une première approche repose sur la résolution d'un programme linéaire en *forme séquentielle* [15, 24, 4], ce qui engendre un algorithme exact. Une seconde approche est d'employer un solveur de jeu itératif, c'est-à-dire soit une méthode basée sur le regret contrefactuel [29, 5], soit une méthode du premier ordre [16], les deux venant avec des propriétés de convergence asymptotiques. En revanche, en évitant la transformation POSG→EFG, notre approche préserve et exploite l'information structurelle contenue dans la représentation POSG. Dans cet article, nous exploitons aussi les propriétés de concavité et de convexité de V^* (la fonction de valeur optimale des zs-POSG) introduites par WIGGERS, OLIEHOEK et ROIJERS [28] et dérivons pour la première fois une approche reposant sur le principe d'optimalité de Bellman. Elle est prometteuse, étant similaire à des approches de l'état de l'art dans des classes de problèmes connexes (POMDP [1, 22], Dec-POMDP [25, 9]) ou des sous-classes

des zs-POSG [10, 7, 3, 13, 8, 12]. Comme elles le font, nous : (i) transformons le problème initial en un problème complètement observable pour lequel le principe d'optimalité de Bellman s'applique directement (section 3) ; puis (ii) exhibons de nouvelles propriétés de continuité de V^*, lesquelles permettent de proposer des approximateurs encadrant à base de points et des opérateurs de mise-à-jour et de sélection efficaces reposant sur la programmation linéaire (sect. 4) ; et (iii) montrons comment adapter le schéma algorithmique HSVI de SMITH et SIMMONS [23] pour résoudre le problème ϵ-optimalement en un nombre fini d'itérations (sect. 5). Enfin, la sect. 6 présente des résultats expérimentaux préliminaires pour valider l'algorithme proposé.

2 Travaux connexes

Les POSG à horizon infini sont indécidables [18], ce qui justifie de chercher des solutions quasi-optimales, par exemple à travers des horizons finis (lesquels garantissent d'être à ϵ de l'optimum sous horizon infini), comme nous allons le faire. Peu de travaux se sont intéressés à la résolution de POSG. Une exception est le travail de HANSEN, BERNSTEIN et ZILBERSTEIN [11] sur les POSG à horizon fini, dans lequel des arbres-politiques non-dominés pour chaque joueur sont construits incrémentalement par programmation dynamique (DP), ce qui permet ensuite de dériver un solveur pour POSG à gain commun, c'est-à-dire des processus de décision markoviens partiellement observables (Dec-POMDP). À l'inverse, nous employons le principe d'optimalité de Bellman pour estimer la fonction de valeur optimale.

Dec-POMDP Le principe d'optimalité de Bellman apparaît comme le composant principal d'un solveur de Dec-POMDP quand SZER, CHARPILLET et ZILBERSTEIN [25] adoptent un point de vue centré-planificateur dans lequel le planificateur fournit aux joueurs leurs politiques privées couvrant tous les historiques d'action-observation qu'ils pourraient rencontrer. L'état d'information du planificateur à t contient ainsi l'état de croyance initial et la politique jointe jusqu'à t. Cela conduit à un MDP sur les états d'information avec une dynamique déterministe résolu avec une recherche A* appelée MAA* (*multi-agent A**).
Ensuite, une autre étape importante est quand DIBANGOYE et al. [9] montrent que (i) l'*état d'occupation*, une statistique employée pour calculer les récompenses espérées dans MAA*, est en fait une statistique suffisant pour la planification, et (ii) la fonction de valeur optimale est linéaire par morceaux et convexe (PWLC) dans l'espace des états d'occupation, ce qui permet d'adapter des solveurs POMDP à base de points en utilisant des approximateurs de V^*.

Sous-classes de POSG Des travaux récents ont abordé des cas particuliers des jeux stochastiques partiellement observables atténués (POSG), à 2 joueurs et somme-0 sauf indication contraire, en exploitant la structure du problème pour le transformer en un problème équivalent dans lequel le principe d'optimalité de Bellman s'applique. GHOSH, MCDONALD et SINHA [10] ont considéré des POSG avec actions publiques et observations partagées, lesquels peuvent être convertis en des jeux stochastiques définis sur l'espace commun des états de croyance, de même que pour les POMDP convertis en MDP sur les états de croyance. CHATTERJEE et DOYEN, BASU et STETTNER, HORÁK, BOŠANSKÝ et PĚCHOUČEK [7, 3, 13] ont considéré des One-Sided POSG, c'est-à-dire des scénarios dans lesquels (le joueur) 2 (sans perte de généralité) n'observe que partiellement l'état du système, et 1 a accès à l'état du système ainsi qu'à l'action et à l'observation de 2. COLE et KOCHERLAKOTA [8] ont considéré des POSG (à n joueurs) avec des états privés indépendants, une observabilité partiellement partagée, et la fonction d'utilité du joueur i dépendant de son état privé et de son observation partagée. HORÁK et BOŠANSKÝ [12] ont considéré des zs-POSG avec des états privés indépendants et des observations publiques, c'est-à-dire des scénarios dans lesquels (i) chaque joueur i a un état privé s_i qu'il observe complètement, et (ii) les deux joueurs reçoivent la même observation publique de l'état privé de chaque joueur. Toute croyance d'un joueur sur l'état privé de l'autre joueur est ainsi une connaissance commune.
Concernant les travaux de HORÁK, BOŠANSKÝ et PĚCHOUČEK, HORÁK et BOŠANSKÝ [13, 12], dans les deux cas des propriété de convexité et de concacivté de la fonction de valeur optimale sont obtenus, lesquels permettent de dériver des approximateurs majorant et minorant. Ces approximateurs sont ensuite employés dans des algorithmes reposant sur HSVI. Toutefois, passer des MDP et POMDP (comme dans le travail de SMITH [22]) à ces cadres induit un arbre des futurs possibles dont le facteur de branchement est infini, ce qui requiert des modifications au schéma algorithmique, et donc à l'analyse théorique de la convergence en temps fini. Comme on le verra, le présent travail adopte des modifications similaires.
WIGGERS, OLIEHOEK et ROIJERS [27] [28] démontrent que, en utilisant des représentations appropriées, la fonction de valeur associée à un zs-POSG est convexe pour le joueur (maximisant) 1 et concave pour le joueur (minimisant) 2. Sur cette base, WIGGERS [26] propose des solveurs heuristiques dépourvus de garanties de convergence, une limitation que nous surmontons dans le présent travail.

Jeux à information imparfaite Comme déjà évoqué en introduction, les POSG à horizon fini (et somme générale) peuvent être écrits comme des jeux en forme extensive avec information imparfaite et mémoire parfaite (EFG, souvent désignés comme des *jeux à information imparfaite*) [20], ce qui rend les techniques de résolution pour EFG pertinentes même pour les POSG à horizon infini. Une première approche pour résoudre les EFG est de les convertir en un jeu en forme normale avant de chercher un équilibre de Nash, en ignorant donc l'aspect temporel du problème [21] et en induisant une explosion combinatoire. KOLLER et MEGIDDO [14] proposent une approche par programma-

tion linéaire améliorée pour zs-EFG qui exploite l'aspect temporel à travers le choix de variables de décision (voir aussi [24, 15]). En utilisant une méthode de double-oracle, BOŠANSKÝ et al. [4] accélèrent ce processus de résolution d'autant plus que les stratégies optimales ont un petit support.

Plus récemment, des méthodes itératives ont été proposées qui fournissent des garanties de convergence asymptotiques et permettent d'aborder des jeux à information imparfaite de grande taille. (i) La minimisation du regret contrefactuel (*Counterfactual Regret minimization* (CFR)) [29] a une vitesse de convergence en $O(\frac{1}{\epsilon^2})$ (avec des versions modernes gagnant désormais contre les meilleurs joueurs humains au poker *heads-up no limit hold'em* [5]), alors que (ii) les méthodes du premier ordre (*first-order methods* (FOMs)) [16] ont une vitesse de convergence en $O(\frac{1}{\epsilon})$. Pour notre part, nous visons un algorithme à erreur bornée ayant des garanties de convergence en temps fini.

3 État de l'art

Par souci de clarté, les concepts et résultats de la littérature EFG employés dans ce travail sont retranscrits dans le cadre POSG.

Un jeu stochastique partiellement observable (à 2 joueurs et) à somme nulle (zs-POSG) est défini par un tuple $\langle \mathcal{S}, \mathcal{A}^1, \mathcal{A}^2, \mathcal{Z}^1, \mathcal{Z}^2, P, r, H, \gamma, b_0 \rangle$, où

- $\mathcal{S}$ est un ensemble fini d'états ;
- $\mathcal{A}^i$ est l'ensemble fini des actions de i ;
- $\mathcal{Z}^i$ est l'ensemble fini des observations de i ;
- $P^{z^1,z^2}_{a^1,a^2}(s'|s)$ est la probabilité de transiter vers l'état s' et de recevoir les observations z^1 et z^2 quand les actions a^1 et a^2 sont effectuées dans l'état s ;
- $r(s, a^1, a^2)$ est une fonction de récompense (scalaire) ;
- $H \in \mathbb{N}$ est un horizon temporel (fini) ;
- $\gamma \in [0, 1]$ est un facteur d'atténuation ; et
- b_0 est l'état de croyance initial.

Le joueur 1 voudrait maximiser l'espérance du retour, définie comme la somme des récompenses futures atténuées, alors que 2 voudrait la minimiser, ce que nous formalisons plus loin. Du fait du cadre symétrique, certaines définitions ou résultats seront donnés du point de vue d'un joueur lorsque seuls des changements évidents sont requis pour l'autre.

Des littératures Dec-POMDP, POSG et EFG, nous employons les concepts et définitions suivants, où $i \in \{1, 2\}$:

$-i$ est l'adversaire de i ($-1 = 2$ & $-2 = 1$).

$\theta^i_\tau = (a^i_1, z^i_1, \ldots, a^i_\tau, z^i_\tau)$ $(\in \Theta^i = \cup_{t=0}^{H-1} \Theta^i_t)$ est un *historique d'action-observation* (AOH) de longueur τ pour i.

$\boldsymbol{\theta}_\tau = (\theta^1_\tau, \theta^2_\tau)$ $(\in \boldsymbol{\Theta} = \cup_{t=0}^{H-1} \boldsymbol{\Theta}_t)$ est une AOH *jointe* à τ.

[σ_τ] Un *état d'occupation* (OS) σ_τ $(\in \mathcal{O}^\sigma = \cup_{t=0}^{H-1} \mathcal{O}^\sigma_t)$ à τ est une distribution de probabilité sur les AOH joints $\boldsymbol{\theta}_\tau$.

[β^i_τ] Une *règle de décision (comportementale)* (DR) au temps τ pour i est une application β^i_τ des AOH privés dans Θ^i_τ vers les *distributions* sur les actions privées. Nous notons $\beta^i_\tau(\theta^i_\tau, a^i)$ la probabilité de choisir a^i en présence de θ^i_τ.

$\boldsymbol{\beta}_\tau = \langle \beta^1_\tau, \beta^2_\tau \rangle$ $(\in \mathcal{B} = \cup_{t=0}^{H-1} \mathcal{B}_t)$ est un *profil de règles de décision*.

$\beta^i_{\tau:\tau'} = (\beta^i_\tau, \ldots, \beta^i_{\tau'})$ est une *stratégie comportementale* pour i du pas de temps τ à τ' (inclus).

$\boldsymbol{\beta}_{\tau:\tau'} = \langle \beta^1_{\tau:\tau'}, \beta^2_{\tau:\tau'} \rangle$ est un *profil de stratégies comportementales*.

[$V_0(\sigma_0, \boldsymbol{\beta}_{0:H-1})$] La *valeur* du profile de stratégies comportementales $\boldsymbol{\beta}_{0:H-1}$ dans σ_0 (à partir du pas de temps 0) est

$$V_0(\sigma_0, \boldsymbol{\beta}_{0:H-1}) = E[\sum_{t=0}^{H-1} \gamma^t R_t \mid \sigma_0, \boldsymbol{\beta}_{0:H-1}], \quad (1)$$

où R_t est la variable aléatoire associée à la récompense instantanée à t.

Notes : (i) Par souci de concision, l'indice «$\tau : H-1$» est souvent écrit «$\tau :$». (ii) On pourra écrire $\beta^{1\intercal}_\tau \cdot f(\cdot)$ toute fonction $f(\beta^1_\tau)$ linéaire en β^1_τ (vu comme un vecteur de $\mathbb{R}^{\Theta^1_\tau \times \mathcal{A}^1}$).

L'objectif premier est ici de trouver une stratégie à l'équilibre de Nash (NES), c'est-à-dire un profil de stratégies comportementales $\boldsymbol{\beta}^*_{0:} = \langle \beta^{1*}_{0:}, \beta^{2*}_{0:} \rangle$ tel qu'aucun joueur n'a intérêt à en dévier, ce qui peut s'écrire :

$$\forall \beta^1, V_0(\sigma_0, \beta^{1*}_{0:}, \beta^{2*}_{0:}) \geq V_0(\sigma_0, \beta^1_{0:}, \beta^{2*}_{0:}), \quad (2)$$
$$\forall \beta^2, V_0(\sigma_0, \beta^{1*}_{0:}, \beta^{2*}_{0:}) \leq V_0(\sigma_0, \beta^{1*}_{0:}, \beta^2_{0:}). \quad (3)$$

Dans un tel jeu à somme-0 et 2 joueurs, tous les NES ont la même valeur (NEV) $V^*_0(\sigma_0) \stackrel{\text{def}}{=} V_0(\sigma_0, \beta^{1*}_{0:}, \beta^{2*}_{0:})$.

Le principe d'optimalité de Bellman ne peut pas s'appliquer directement dans un jeu où les joueurs ne partagent pas leurs historiques individuels, et n'ont ainsi pas la même information sur la situation courante (sauf à $\tau = 0$). Pour remédier à ce problème, nous suivons la même idée que pour les Dec-POMDP ou certaines sous-classes de zs-POSG telles que les One-Sided POSG [13] en considérant un jeu différent dont les nouveaux joueurs (disons $\tilde{1}$ et $\tilde{2}$) n'accèdent pas aux historiques individuels de 1 et 2 en cours de partie, mais doivent (publiquement) fournir à 1 et 2 les règles de décision qu'ils devront exécuter. C'est ce que nous faisons dans la section suivante, en démontrant ultérieurement que l'on peut au final dériver des stratégies solution pour le problème original qui sont robustes aux déviations.

3.1 Résoudre des POSG comme des Occupancy MG

Nous décrivons ici la reformulation d'un zs-POSG en un jeu de Markov à somme nulle différent, complètement observable.

Notons d'abord qu'un profil de stratégies partiel $\boldsymbol{\beta}_{0:\tau-1}$ est associé de manière déterministe à un état d'occupation $\sigma_\tau = \sigma_{\boldsymbol{\beta}_{0:\tau-1}}$, lequel exhibe les propriétés suivantes.

Lemma 1. *$\sigma_{\boldsymbol{\beta}_{0:\tau-1}}$, pris avec $\boldsymbol{\beta}_\tau$, est une statistique suffisante pour calculer (i) le prochain état d'occupation, $\sigma_{\boldsymbol{\beta}_{0:\tau}}$, et (ii) l'espérance de récompense à τ : $\mathbb{E}\left[R_\tau \mid \boldsymbol{\beta}_{0:\tau-1} \oplus \boldsymbol{\beta}_\tau\right]$.*

Cette propriété de Markov sur les états d'occupation et la capacité à estimer la récompense espérée permettent d'introduire un jeu équivalent (employé implicitement par WIGGERS, OLIEHOEK et ROIJERS [27]), appelé *jeu de Markov à somme nulle sur les états d'occupation* (*zs Occupancy Markov Game* (zs-OMG)),[1] défini formellement par le tuple $\langle \mathcal{O}^\sigma, \mathcal{B}, T, r, H, \gamma, b_0 \rangle$, où :

— $\mathcal{O}^\sigma$ est l'ensemble des états d'occupation induits par le zs-POSG ;

— $\mathcal{B}$ est l'ensemble des profils de règles de décision du zs-POSG ;

— T est une fonction de transition déterministe qui associe à chaque couple $(\sigma_\tau, \boldsymbol{\beta}_\tau)$ le (seul) état d'occupation suivant possible $\sigma_{\tau+1}$; formellement, $\forall \theta^1_\tau, a^1, z^1, \theta^2_\tau, a^2, z^2$,

$$T(\sigma_\tau, \boldsymbol{\beta}_\tau)((\theta^1_\tau, a^1, z^1), (\theta^2_\tau, a^2, z^2)) \tag{4}$$

$$\stackrel{\text{def}}{=} Pr((\theta^1_\tau, a^1, z^1), (\theta^2_\tau, a^2, z^2) | \sigma_\tau, \boldsymbol{\beta}_\tau) \tag{5}$$

$$= \beta^1_\tau(\theta^1_\tau, a^1)\beta^2_\tau(\theta^2_\tau, a^2)\sigma_\tau(\boldsymbol{\theta}_\tau) \sum_{s,s'} P^{\boldsymbol{z}}_{\boldsymbol{a}}(s'|s) b(s|\boldsymbol{\theta}_\tau), \tag{6}$$

où $b(s|\boldsymbol{\theta}_\tau)$ est un état de croyance obtenu par filtrage HMM ;

— r est une fonction de récompense induite par le zs-POSG comme récompense espérée pour l'état d'occupation courant et le profil de règle de décision courant :

$$r(\sigma_\tau, \boldsymbol{\beta}_\tau) \stackrel{\text{def}}{=} E[r(S, A^1, A^2) | \sigma_\tau, \boldsymbol{\beta}_\tau] \tag{7}$$

$$= \sum_{s, \boldsymbol{\theta}_\tau, \boldsymbol{a}} \sigma_\tau(\boldsymbol{\theta}_\tau) b(s|\boldsymbol{\theta}_\tau) \beta^1_\tau(a^1|\theta^1)\beta^2_\tau(a^2|\theta^2) r(s, \boldsymbol{a}); \tag{8}$$

nous employons la même notation r pour les zs-POSG dans la mesure où le contexte indiquera duquel il est question ;

— H, γ, et b_0 sont les mêmes que dans le zs-POSG.

1. Nous utilisons (i) «jeu de Markov» au lieu de «jeu stochastique» parce que la dynamique n'est pas stochastique, et (ii) «partially observable stochastic game» pour correspondre à la littérature.

Ce n'est pas un jeu de Markov à somme nulle fini standard non plus puisque (i) il est non stationnaire, avec des espaces d'états et d'actions différents (et continus) à chaque pas de temps ; (ii) à chaque pas de temps, il y a un nombre infini d'actions, et un mélange de telles actions pures est équivalent à une action pure pré-existante ; et (iii) la dynamique est déterministe (même pour des actions «mixtes»). Mais un bénéfice important de travailler avec un zs-OMG est que $\tilde{1}$ et $\tilde{2}$ connaissent tous deux l'état d'occupation courant, σ_τ, et partagent ainsi la même information sur le jeu. Cela va permettre de trouver un ϵ-équilibre de Nash solution de ce jeu en explorant l'arbre des profils de stratégies comportementales partielles. Cet arbre a un facteur de branchement infini du fait des espaces d'action (les règles de décision) et d'état (les états d'occupation) continus (contrairement à l'arbre exploré lors de la résolution d'un Dec-POMDP à l'aide d'un occupancy MDP).

Dans un zs-OMG, à cause de l'observabilité partielle, si $\tilde{2}$ dévie de sa stratégie solution obtenue, alors $\tilde{1}$ pourrait s'adapter en-ligne en replanifiant depuis l'état d'occupation (observé) résultant (à supposer qu'il dispose d'un temps de calcul suffisant). Dans le zs-POSG correspondant, l'état d'occupation n'est pas observé, ce qui interdit une telle replanification. Toutefois, comme nous le verrons, on peut quand même dériver une stratégie solution ϵ-optimale pour 1 comme pour 2 (lesquels nous ne distinguerons plus de $\tilde{1}$ et $\tilde{2}$ désormais).

Nous allons étudier les sous-jeux d'un zs-OMG, c'est-à-dire les situations où un état d'occupation σ_τ a été atteint au temps τ, et le solveur cherche des stratégies optimales ($\beta^1_{\tau:}$ et $\beta^2_{\tau:}$) à fournir aux joueurs. σ_τ indique à quels historiques chaque joueur pourrait faire face avec probabilité non-nulle, et sont donc pertinents pour la planification. Nous pouvons alors étendre la définition de la fonction de valeur (optimale) à tout pas de temps τ comme suit :

$$V_\tau(\sigma_\tau, \boldsymbol{\beta}_{\tau:}) \stackrel{\text{def}}{=} E[\sum_{t=\tau}^{\infty} \gamma^{t-\tau} R(S_t, A_t) | \sigma_\tau, \boldsymbol{\beta}_{\tau:}], \tag{9}$$

$$V^*_\tau(\sigma_\tau) \stackrel{\text{def}}{=} \max_{\beta^1_{\tau:}} \min_{\beta^2_{\tau:}} V_\tau(\sigma_\tau, \beta^1_{\tau:}, \beta^2_{\tau:}). \tag{10}$$

Notons qu'une telle extension est non triviale quand on utilise des stratégies mixtes puisque celles-ci sont usuellement définies à partir du pas de temps 0.

Nous regardons maintenant des propriétés structurelles de V^* qui seront utilisées plus tard pour définir des approximateurs de fonction.

3.2 Propriétés de concavité et de convexité de V^*

WIGGERS, OLIEHOEK et ROIJERS [27] (dont nous étendons ici les résultats de $\gamma = 1$ à $\gamma \leq 1$) définissent les distributions marginale et conditionnelle du point de vue de 1 comme $\sigma^{m,1}_\tau(\theta^1_\tau) \stackrel{\text{def}}{=} \sum_{\theta^2_\tau} \sigma(\theta^1_\tau, \theta^2_\tau)$ et $\sigma^{c,1}_\tau(\theta^2_\tau | \theta^1_\tau) \stackrel{\text{def}}{=} \frac{\sigma(\theta^1, \theta^2)}{\sigma^1_{\tau,m}(\theta^1)}$, de sorte que $\sigma_\tau = \sigma^{m,1}_\tau \sigma^{c,1}_\tau$. De plus, $T^1_m(\cdot)$ et

$T_c^1(\cdot)$ dénoteront les états d'occupation marginal et conditionnel de 1 induits par $T(\cdot)$.
Pour un $\beta^2_{\tau:}$ fixé, 1 est confronté à un POMDP, et la valeur (POMDP) optimale dans chaque historique d'action-observation θ^1_τ est donnée par $\nu_{[\sigma^{c,1}_\tau, \beta^2_{\tau:}]}(\theta^1_\tau) \stackrel{\text{def}}{=} \max_{\beta^1_{\tau:}} \mathbb{E}\left\{\sum_{t=\tau}^{H-1} \gamma^{t-\tau} R(S_t, A^1_t, A^2_t) \mid \theta^1_\tau, \beta^1_{\tau:}, \beta^2_{\tau:}, \sigma^{c,1}_\tau\right\}$.
WIGGERS, OLIEHOEK et ROIJERS démontrent ensuite les propriétés de concavité et convexité suivantes de V^*.

Theorem 1 (Concavité et convexité de V^*_τ)**.** *Pour tout $\tau \in \{0\,..\,H-1\}$, V^*_τ est (i) concave en $\sigma^{m,1}_\tau$ pour $\sigma^{c,1}_\tau$ fixé, et (ii) convexe en $\sigma^{m,2}_\tau$ pour $\sigma^{c,2}_\tau$ fixé. Plus précisément,*

$$V^*_\tau(\sigma_\tau) = \min_{\beta^2_{\tau:}} \left[\sigma^{m,1}_\tau \cdot \nu_{[\sigma^{c,1}_\tau, \beta^2_{\tau:}]}\right], \tag{11}$$

$$= \max_{\beta^1_{\tau:}} \left[\sigma^{m,2}_\tau \cdot \nu_{[\sigma^{c,2}_\tau, \beta^1_{\tau:}]}\right]. \tag{12}$$

Toutefois, cette propriété seule ne permet d'approximer la valeur optimale que pour des états d'occupation conditionnels $\sigma^{c,i}_\tau$ fixes (donc en nombre fini), en utilisant pour chacun un ensemble fini de vecteurs $\nu_{[\sigma^{c,i}_\tau, \beta^{-i}_{\tau:}]}$, mais pas pour l'espace d'occupation complet.

3.3 Introduction des jeux locaux

Pour tous τ et σ_τ, définissons (i) $\boldsymbol{\beta}^*_{\tau:}(\sigma_\tau)$ un profil à l'équilibre de Nash pour le *sous*-jeu à σ_τ et (ii) le jeu *local* à σ_τ

$$Q^*_\tau(\sigma_\tau, \boldsymbol{\beta}_\tau) = r(\sigma_\tau, \boldsymbol{\beta}_\tau) + \gamma V^*_{\tau+1}(T(\sigma_\tau, \boldsymbol{\beta}_\tau)), \tag{13}$$

(qui suppose une fonction $V^*_{\tau+1}(\cdot)$ connue). T étant linéaire en β^1_τ et β^2_τ, le théorème 1 conduit au résultat suivant.

Lemma 2. *$Q^*_\tau(\sigma_\tau, \boldsymbol{\beta}_\tau)$ est concave en β^1_τ et convexe en β^2_τ.*

Ce lemme 2 permet d'appliquer le théorème du minimax de VON NEUMANN [19] dans ces jeux locaux. En conséquence, et alors que $Q^*_\tau(\sigma_\tau, \cdot, \cdot)$ peut ne pas être bilinéaire, (i) tous les profils de stratégies à l'équilibre de Nash ont même valeur et sont interchangeables, et (ii) une stratégie à l'équilibre de Nash peut être trouvée par optimisation biniveau. On peut donc travailler avec des jeux locaux au lieu des sous-jeux, construisant une solution du zs-OMG par concaténation de solutions de jeux locaux successifs.

4 Propriétés et approximation de V^*

Cette section introduit des approximateurs majorant et minorant de la fonction de valeur exploitant des propriétés de concavité et convexité (CC) et de Lipschitz-continuité (LC).

4.1 Lipschitz-continuité de V^*

L'établissement de la Lipschitz-continuité de V^* part de propriétés de T.

Lemma 3. *À profondeur τ, $T(\sigma_\tau, \boldsymbol{\beta}_\tau)$ est linéaire en β^1_τ, β^2_τ, et σ_τ, où $\boldsymbol{\beta}_\tau = \langle \beta^1_\tau, \beta^2_\tau \rangle$. Elle est plus précisément 1-Lipschitz-continue (1-LC) en σ_τ (en norme 1), c'est-à-dire que, pour tous σ_τ, σ'_τ :*

$$\|T(\sigma'_\tau, \boldsymbol{\beta}_\tau) - T(\sigma_\tau, \boldsymbol{\beta}_\tau)\|_1 \leq 1 \cdot \|\sigma'_\tau - \sigma_\tau\|_1. \tag{14}$$

Aussi, la récompense espérée en tout τ est linéaire en σ_τ (voir anx. A.1), tout comme l'est aussi la valeur espérée d'un profil de stratégies partielles partant de τ.

Lemma 4. *À profondeur τ, $V_\tau(\sigma_\tau, \boldsymbol{\beta}_{\tau:})$ est linéaire en σ_τ.*

Cela conduit à la Lipschitz-continuité de V^*_τ en σ_τ.

Theorem 2. *Soit $h_\tau \stackrel{\text{def}}{=} \frac{1-\gamma^{H-\tau}}{1-\gamma}$ si $\gamma < 1$, sinon $h_\tau \stackrel{\text{def}}{=} H - \tau$ (si $\gamma = 1$). Alors $V^*_\tau(\sigma_\tau)$ est λ_τ-Lipschitz-continue en σ_τ à toute profondeur $\tau \in \{0\,..\,H-1\}$, où $\lambda_\tau = \frac{1}{2} h_\tau (r_{\max} - r_{\min})$.*

4.2 Approximateurs

Un algorithme de résolution peut reposer sur la seule Lipschitz-continuité, comme présenté par BUFFET et al. [6]. Celui-ci souffre toutefois non seulement d'approximateurs faibles, mais aussi d'employer une optimisation bi-niveau globale dans l'espace (à grande dimension) des règles de décision.
L'exploitation des propriétés de concavité et de convexité de V^*_τ permet de dériver les approximateurs majorant et minorant suivants :

$$\overline{V}_\tau(\sigma_\tau) = \min_{\substack{\langle \tilde{\sigma}^{c,1}_\tau, \nu^2_\tau \rangle \\ \in \overline{bag}V}} \left[\sigma^{m,1}_\tau \cdot \nu^2_\tau + \lambda \|\sigma_\tau - \sigma^{m,1}_\tau \tilde{\sigma}^{c,1}_\tau\|_1\right], \tag{15}$$

$$\underline{V}_\tau(\sigma_\tau) = \max_{\substack{\langle \tilde{\sigma}^{c,2}_\tau, \nu^1_\tau \rangle \\ \in \underline{bag}V}} \left[\sigma^{m,2}_\tau \cdot \nu^1_\tau - \lambda \|\sigma_\tau - \sigma^{m,2}_\tau \tilde{\sigma}^{c,2}_\tau\|_1\right]. \tag{16}$$

Ne disposant pas d'une façon efficace de résoudre, à l'aide de ces approximateurs, les jeux locaux majorant et minorant à σ_τ induits par l'éq. (13), nous introduisons les deux fonctions de valeurs intermédiaires suivantes (entre Q^* et V^*) dont les approximations vont faciliter la résolution des jeux locaux :

$$W^{1,*}_\tau(\sigma_\tau, \beta^1_\tau) \stackrel{\text{def}}{=} \min_{\beta^2_\tau} Q^*_\tau(\sigma_\tau, \beta^1_\tau, \beta^2_\tau), \tag{17}$$

$$W^{2,*}_\tau(\sigma_\tau, \beta^2_\tau) \stackrel{\text{def}}{=} \max_{\beta^1_\tau} Q^*_\tau(\sigma_\tau, \beta^1_\tau, \beta^2_\tau). \tag{18}$$

Ensuite, $V^*_{\tau+1}$ étant Lipschitz-continue en $\sigma_{\tau+1}$ (théorème 2), et en exploitant les propriétés de linéarité et d'indépendance de $T^1_m(\sigma_\tau, \boldsymbol{\beta}_\tau)$ et $T^1_c(\sigma_\tau, \boldsymbol{\beta}_\tau)$ (lemmes 7+8, anx. B.2), nous pouvons dériver un approximateur *majorant* $\overline{W}^1_\tau$ de $W^{1,*}_\tau$ (et réciproquement un approximateur *minorant* $\underline{W}^2_\tau$ de $W^{2,*}_\tau$) en utilisant un nombre fini de

tuples $\langle \tilde{\sigma}_\tau, \beta^2_\tau, \nu^2_{\tau+1} \rangle$ stockés dans un ensemble $\overline{bagW}_\tau$ (*cf.* anx. B.2) :

$$\overline{W}^1_\tau(\sigma_\tau, \beta^1_\tau) = \min_{\substack{\langle \tilde{\sigma}_\tau, \beta^2_\tau, \nu^2_{\tau+1} \rangle \\ \in \overline{bagW}_\tau}} \Big(\beta^{1\intercal}_\tau \cdot \big[r(\sigma_\tau, \cdot, \beta^2_\tau) \tag{19}$$

$$+ \gamma T^1_m(\tilde{\sigma}_\tau, \cdot, \beta^2_\tau) \cdot \nu^2_{\tau+1} \big] + \gamma \lambda_\tau \cdot \|\sigma_\tau - \tilde{\sigma}_\tau\|_1 \Big). \tag{20}$$

Comme détaillé dans **??**, étant donnée une distribution δ^2_τ sur des tuples $w = \langle \tilde{\sigma}_\tau, \beta^2_\tau, \nu^2_{\tau+1} \rangle$ de $\overline{W}_\tau$, nous pouvons maintenant majorer la valeur du «profil» $\langle \beta^1_\tau, \delta^2_\tau \rangle$ dans σ_τ comme

$$\sum_{w \in \overline{W}_\tau} \beta^{1\intercal}_\tau \cdot \Big[r(\sigma_\tau, \cdot, \beta^2_\tau[w]) \tag{21}$$

$$+ \gamma T^1_m(\tilde{\sigma}_\tau[w], \cdot, \beta^2_\tau[w]) \cdot \nu^2_{\tau+1}[w] \tag{22}$$

$$+ \gamma \lambda_\tau \cdot \sigma^{m,1}_\tau(\Theta^1_\tau \times \mathcal{A}^1) \cdot \|\sigma_\tau - \tilde{\sigma}_\tau[w]\|_1 \Big] \delta^2_\tau(w) \tag{23}$$

(où $x[w]$ dénote le champ x du tuple w, et $\sigma^{m,1}_\tau(\Theta^1_\tau \times \mathcal{A}^1)$ est un vecteur indexé par des couples (θ^1_τ, a^1), et dont le composant $\sigma^{m,1}_\tau(\theta^1_\tau, a^1)$ a comme valeur $\sigma^{m,1}_\tau(\theta^1_\tau)$)

$$= \beta^{1\intercal}_\tau \cdot M^{\sigma_\tau} \cdot \delta^2_\tau, \tag{24}$$

avec M^{σ_τ} une matrice de taille $|\Theta^1_\tau \times \mathcal{A}^1| \times |\overline{bagW}^1_\tau|$, où

$$M^{\sigma_\tau}_{(\langle \theta^1_\tau, a^1 \rangle, w)} = \sum_{s, \theta^2_\tau, a^2} \sigma_\tau(\boldsymbol{\theta}_\tau) b(s|\boldsymbol{\theta}_\tau) \beta^2_\tau[w](a^2|\theta^2) r(s, \boldsymbol{a}) \tag{25}$$

$$+ \sum_{z^1} \Big[\sum_{\theta^2_\tau, a^2} \beta^2_\tau[w](a^2|\theta^2_\tau) \sum_{s, s', z^2} P^{\boldsymbol{z}}_{\boldsymbol{a}}(s'|s) b(s|\boldsymbol{\theta}_\tau) \tag{26}$$

$$\cdot \tilde{\sigma}_\tau[w](\boldsymbol{\theta}_\tau) \Big] \cdot \nu^2_{\tau+1}[w](\theta^1_\tau, a^1, z^1) \tag{27}$$

$$+ \gamma \lambda_{(\tau+1)} \cdot \sigma^{m,1}_\tau(\theta^1_\tau) \cdot \|\sigma_\tau - \tilde{\sigma}_\tau[w]\|_1. \tag{28}$$

Ainsi, la résolution de $\max_{\beta^1_\tau} \overline{W}^1_\tau(\sigma_\tau, \beta^1_\tau)$ peut être réécrite comme la résolution d'un jeu de matrice à somme nulle dont les stratégies pures sont : pour 1, le choix de $|\Theta^1_\tau|$ actions et, pour 2, le choix d'1 élément de $\overline{bagW}^1_\tau$. Le LP correspondant est :

$$\max_{\beta^1_\tau, v} v \text{ s.t. } \forall w \in \overline{bagW}_\tau, \quad v \leq \beta^{1\intercal}_\tau \cdot M^{\sigma_\tau}_{(\cdot, w)} \tag{29}$$

$$\forall \theta^1_\tau \in \Theta^1_\tau, \ \textstyle\sum_{a^1} \beta^1_\tau(a^1|\theta^1_\tau) = 1, \tag{30}$$

et le LP dual :

$$\min_{\delta^2_\tau, v} v \text{ s.t. } \forall (\theta^1_\tau, a^1), \quad v \leq M^{\sigma_\tau}_{((\theta^1_\tau, a^1), \cdot)} \cdot \delta^2_\tau \tag{31}$$

$$\textstyle\sum_{w \in \overline{bagW}_\tau} \delta^2_\tau(w) = 1. \tag{32}$$

Nous regardons maintenant les opérateurs employés pour manipuler ces approximateurs.

4.3 Opérateurs associés

Pour $\tau \geq 1$, $\overline{V}_\tau$ et $\overline{W}^1_{\tau-1}$ reposent tous deux essentiellement sur la même information et sont fortement reliés, de sorte que nous les discuterons conjointement. $\overline{bagV}_\tau$ contient des tuples $\langle \sigma^{c,1}_\tau, \langle \nu^2_\tau, \delta^2_\tau \rangle \rangle$, et $\overline{bagW}^1_{\tau-1}$ (pour $\tau \geq 1$) les tuples associés $\langle \sigma_{\tau-1}, \beta^2_{\tau-1}, \langle \nu^2_\tau, \delta^2_\tau \rangle \rangle$. Ici, δ^2_τ, en tant que distribution sur les tuples de $\overline{bagW}^1_\tau$, induit une stratégie définie récursivement pour 2 comme (i) une mixture de règles de décisions comportementales à τ :

$$\beta^2_\tau[\delta^2_\tau] \stackrel{\text{def}}{=} \sum_{\tilde{\beta}^2_\tau \in \overline{bagW}^1_\tau} \delta^2_\tau(\tilde{\beta}^2_\tau) \cdot \tilde{\beta}^2_\tau, \tag{33}$$

et (ii) une mixture d'autres stratégies mélangées pour les pas de temps $\tau + 1$ et suivants :

$$\delta^2_{\tau+1}[\delta^2_\tau] \stackrel{\text{def}}{=} \sum_{\tilde{\delta}^2_{\tau+1} \in \overline{bagW}^1_\tau} \delta^2_\tau(\tilde{\delta}^2_{\tau+1}) \cdot \tilde{\delta}^2_{\tau+1}, \tag{34}$$

jusqu'à atteindre l'horizon. Note : Par commodité, nous pourrons remplacer un tuple complet par les quelques éléments d'intérêt.

Initialisations On peut chercher un majorant de V^*, c'est-à-dire une borne optimiste (une heuristique admissible) pour le joueur (maximisant) 1, en relâchant le problème auquel 1 fait face. Dans ce but, nous résolvons ici le POMDP induit quand est affecté à 2 un $\beta^{2,\varnothing}_{0:}$ uniformément aléatoire, la meilleure réponse résultante étant notée $\beta^{1,\otimes}_{0:}$. À profondeur τ, $\beta^{2,\varnothing}_{0:}$ et $\beta^{1,\otimes}_{0:}$ induisent (i) un état d'occupation OS σ_τ et (ii) un vecteur $\overline{\nu}_\tau$, où $\overline{\nu}_\tau(\theta^1_\tau)$ est la valeur de $\beta^{1,\otimes}_{0:}$ en θ^1_τ (contre $\beta^{2,\varnothing}_{0:}$ et sous $\sigma^{c,1}_\tau$).
Étant données ces stratégies, chaque $\overline{bagV}_\tau$ (respectivement $\overline{bagW}^1_{\tau-1}$) est initialisé avec $\langle \sigma^{c,1}_\tau, \langle \overline{\nu}_\tau, \delta^{2,\varnothing}_\tau \rangle \rangle$ (resp. $\langle \sigma_{\tau-1}, \beta^{2,\varnothing}_{\tau-1}, \langle \overline{\nu}_\tau, \delta^{2,\varnothing}_\tau \rangle \rangle$), où $\delta^{2,\varnothing}_\tau$ est une distribution dégénérée sur le seul élément dans $\overline{bagW}^1_\tau$.
Des initialisations plus avancées pourraient être proposées en améliorant les stratégies de 2, en utilisant une relaxation One-Sided zsPOSG à la place, ou en combinant plusieurs initialisations.

Mise-à-jour de $\overline{V}_\tau$ et $\overline{W}^1_{\tau-1}$ Pour mettre à jour $\overline{bagV}_\tau$ et (si $\tau \geq 1$) $\overline{bagW}^1_{\tau-1}$ simultanément, nous considérons un tuple $\langle \sigma_\tau, \sigma^{c,1}_{\tau-1}, \beta^2_{\tau-1} \rangle$ (partiellement indéfini si $\tau = 0$), et obtenons un couple approprié $\langle \nu^2_\tau, \overline{\delta}^2_\tau \rangle$ en résolvant le LP dual (31), noté $\arg\min_{\delta^2_\tau} \overline{W}^1_\tau(\sigma_\tau, \delta^2_\tau)$, donc reposant sur $\overline{bagV}_{\tau+1}$.
Pour un $\overline{\delta}^2_\tau$ donné, $\nu^2_\tau(\theta^1_\tau, \overline{\delta}^2_\tau)$ est la valeur de la meilleure action a^1 de 1 en supposant que (i) 2 suit la stratégie $\overline{\delta}^2_\tau$ et (ii) le retour espéré à partir de $\tau + 1$ est donné par $\overline{V}_{\tau+1}$ ($= 0$ si $\tau + 1 = H$) :

$$\nu^2_\tau(\theta^1_\tau, \overline{\delta}^2_\tau) = \frac{1}{\sigma^1_{\tau,m}(\theta^1_\tau)} \max_{a^1 \in \mathcal{A}^1} M^{\sigma_\tau}_{((\theta^1_\tau, a^1), .)} \cdot \overline{\delta}^2_\tau. \tag{35}$$

On a alors besoin d'ajouter $\langle \sigma_\tau^{c,1}, \langle \nu_\tau^2(\underline{\delta}_\tau^2), \underline{\delta}_\tau^2 \rangle \rangle$ to $\overline{bagV}_\tau$, et (si $\tau \geq 1$) $\langle \sigma_{\tau-1}, \beta_{\tau-1}^2 \langle \nu_\tau^2(\underline{\delta}_\tau^2), \underline{\delta}_\tau^2 \rangle, \rangle$ à $\overline{bagW}_{\tau-1}^1$.
Ce processus est résumé en ligne 14 sqq. de l'algorithme 1.

Élagage Parce qu'elles ont des formes différentes, $\overline{V}_\tau$ et $\overline{bagW}_{\tau-1}^1$ doivent être élaguées indépendamment. Aussi, l'élagage ne devrait pas casser les stratégies définies récursivement δ_τ^2. Les tuples élagués devraient donc rester stockés en mémoire (si utilisés).
$\overline{V}_\tau$ repose sur une représentation «min-surfaces» (plutôt que «min-planes»), où chaque surface est linéaire en $\sigma_\tau^{m,1}$ et exploite la Lipschitz-continuité. Cela permet d'exploiter (en les inversant) les méthodes d'élagage max-planes pour POMDP comme expliqué dans le résultat suivant.

Theorem 3. *Soit P un opérateur d'élagage* min*-planes (inverse d'un élagage* max*-planes pour POMDP), et $\langle \nu_\tau, \sigma_\tau^{c,1} \rangle \in \overline{bagV}_\tau$. Si P identifie correctement ν_τ comme non-dominé (ou respectivement dominé) sous $\sigma_\tau^{c,1}$ fixe, alors $\langle \nu_\tau, \sigma_\tau^{c,1} \rangle$ est non-dominé (ou resp. dominé) dans $\mathcal{O}_\tau^\sigma$.*

Le fait qu'un test induise des faux positifs (élaguant des éléments non-dominés) ou des faux négatifs (n'élaguant pas des éléments dominés) se propage donc du cadre min-planes au cadre min-surfaces.

Pour sa part, $\overline{W}_\tau^1$ impliquant un terme de récompense qui est bilinéaire (linéaire à la fois en σ_τ et β_τ^1), dériver des techniques d'élagage n'est pas si direct. Cela peut requérir d'utiliser de l'optimisation quadratique, **ou** de majorer le terme de récompense bilinéaire avec un linéaire pour retomber sur le cadre «min-surface» précédent. Alors qu'on s'attend à ce que la résolution de jeux locaux bénéficie de manière significative de l'élagage de $\overline{W}_\tau^1$, nous laissons ce sujet pour des travaux futurs.

5 HSVI pour zs-POSG

Dans cette section, nous cherchons des solutions ϵ-optimales.

5.1 Algorithme

HSVI pour zs-OMG est décrit dans l'algorithme 1. Comme le HSVI de base, il repose sur (i) la génération de trajectoires en agissant avec optimisme (lignes 9+10), c'est-à-dire que le joueur 1 (resp. 2) agit de manière «gourmande» par rapport à $\overline{W}_\tau$ (resp. $\underline{W}_\tau$), et (ii) la mise-à-jour locale des approximateurs majorant et minorant (lignes 12+13). Ici, les calculs des mises-à-jour des valeurs et des stratégies repose sur la résolution de jeux en forme normale décrits par le LP (29). Notons que l'implémentation maintient des états d'occupation *complets* $o_\tau \in \Delta(\mathcal{S} \times \mathbf{\Theta}_\tau)$, ce qui permet de facilement retrouver à la fois les états d'occupation «simples» $\sigma_\tau \in \mathcal{O}_\tau^\sigma = \Delta(\mathbf{\Theta}_\tau)$ et les «croyances» $b(s|\boldsymbol{\theta}_\tau)$. Une différence clef avec l'algorithme de SMITH et SIMMONS réside dans le critère d'arrêt des trajectoires. Dans l'algorithme HSVI de base (pour POMDP), le facteur de branchement fini permet de regarder la convergence de $\overline{V}$ et $\underline{V}$ en chaque point atteignable sous une stratégie optimale. Pour garantir l'ϵ-convergence en σ_0, les trajectoires doivent juste être interrompues quand la largeur courante en σ_τ ($\stackrel{\text{def}}{=} \overline{V}_\tau(\sigma_\tau) - \underline{V}_\tau(\sigma_\tau)$) est plus petite qu'un seuil $\gamma^{-\tau}\epsilon$. (Cela arrive même si $\gamma = 1$ parce que la largeur de l'approximateur est nulle au-delà de H.) Ici, ayant à faire à un facteur de branchement infini, on peut converger vers une solution optimale tout en visitant toujours de nouveaux points de l'espace des états d'occupation. Pour contrer cela, nous bornons la largeur à l'intérieur de boules autour des points visités en exploitant la Lipschitz-continuité de V^*. Ceci est accompli en ajoutant un terme $-\sum_{i=1}^{\tau} 2\rho\lambda_{\tau-i}\gamma^{-i}$ [13] (même si $\gamma = 1$) pour garantir que la largeur est inférieure à $\gamma^\tau\epsilon$ à l'intérieur d'une boule de rayon ρ autour du point courant (ici σ_τ). D'où le seuil

$$thr(\tau) \stackrel{\text{def}}{=} \gamma^{-\tau}\epsilon - \sum_{i=1}^{\tau} 2\rho\lambda_{\tau-i}\gamma^{-i}. \tag{36}$$

Algorithme 1 : zs-OMG-HSVI($b_0, [\epsilon, \rho]$)
[here returning solution strategy $\underline{\delta}_0^1$ for Player 1]

1 **Fct** `zs-OMG-HSVI` $(b_0 \simeq \sigma_0)$
2 Initialize $\overline{V}_{\dots}$ & $\underline{V}_{\dots}$
3 **while** $\left[\overline{V}_0(\sigma_0) - \underline{V}_0(\sigma_0) > thr(0)\right]$ **do**
4 `Explore` $(\sigma_0, 0, -, -)$
5 $\underline{\delta}_0^1 \leftarrow \underset{\langle \tilde{\sigma}_0 = \sigma_0, \langle \nu_0^1, \underline{\delta}_0^1 \rangle, - \rangle \in \underline{bagV}_0}{\arg\max} \left(\sigma_0^{m,2} \cdot \nu_0^1 + \lambda_\tau \cdot 0 \right)$
6 **return** $\underline{\delta}_0^1$
7 **Fct** `Explore` $(\sigma_\tau, \tau, \sigma_{\tau-1}, \boldsymbol{\beta}_{\tau-1})$
8 **if** $\left[\overline{V}_\tau(\sigma) - \underline{V}_\tau(\sigma) > thr(\tau)\right]$ **then**
9 $\overline{\beta}_\tau^1 \leftarrow \arg\max_{\beta_\tau^1} \overline{W}_\tau^1(\sigma, \beta_\tau^1)$
10 $\underline{\beta}_\tau^2 \leftarrow \arg\min_{\beta_\tau^2} \underline{W}_\tau^2(\sigma, \beta_\tau^2)$
11 `Explore` $(T(\sigma_\tau, \overline{\beta}_\tau^1, \underline{\beta}_\tau^2), \tau{+}1, \sigma_\tau, \langle \overline{\beta}_\tau^1, \underline{\beta}_\tau^2 \rangle)$
12 $\overline{\textbf{Update}}(\overline{V}_\tau, \overline{W}_{\tau-1}^1, \langle \sigma_\tau, \sigma_{\tau-1}^{c,1}, \underline{\beta}_{\tau-1}^2 \rangle)$
13 $\underline{\textbf{Update}}(\underline{V}_\tau, \underline{W}_{\tau-1}^2, \langle \sigma_\tau, \sigma_{\tau-1}^{c,2}, \overline{\beta}_{\tau-1}^1 \rangle)$
14 **Fct** $\overline{\textbf{Update}}$ $(\overline{V}_\tau, \overline{W}_{\tau-1}^1, \langle \sigma_\tau, \sigma_{\tau-1}^{c,1}, \beta_{\tau-1}^2 \rangle)$
15 $\overline{\delta}_\tau^2 \leftarrow \arg\min_{\delta_\tau^2} \overline{W}_\tau^1(\sigma_\tau, \delta_\tau^2)$
16 $\nu_\tau^2 \leftarrow \nu^2(\overline{\delta}_\tau^2)$
17 $\overline{bagV}_\tau \leftarrow \overline{bagV}_\tau \cup \{\langle \sigma_\tau^{c,1}, \langle \nu_\tau^2, \overline{\delta}_\tau^2 \rangle \rangle\}$
18 $\overline{bagW}_{\tau-1}^1 \leftarrow \overline{bagW}_{\tau-1}^1 \cup \{\langle \sigma_{\tau-1}, \beta_{\tau-1}^2, \langle \nu_\tau^2, \overline{\delta}_\tau^2 \rangle \rangle\}$

Régler ρ Comme on peut l'observer, cette fonction seuil devrait toujours retourner des valeurs positives, ce qui requiert un ρ suffisamment petit. Pour un problème donné, la valeur maximum possible de ρ dépend des constantes de Lipschitz à chaque pas de temps, lesquelles dépendent

elles-mêmes des majorant et minorant initiaux de la fonction de valeur optimale.

Lemma 5. *En bornant λ_τ par $\lambda^\infty = \frac{1}{2}\frac{1}{1-\gamma}[r_{\max} - r_{\min}]$ quand $\gamma < 1$, et en notant que*

$$thr(\tau) = \begin{cases} \gamma^{-\tau}\epsilon - 2\rho\lambda^\infty \frac{\gamma^{-\tau}-1}{1-\gamma} \text{ if } \gamma < 1, \text{ else} \\ \epsilon - \rho(r_{\max} - r_{\min})(2H + 1 - \tau)\tau, \end{cases} \quad (37)$$

on peut garantir la positivité du seuil en tout $\tau \in 1\,..\,H-1$ en contraignant

$$0 < \rho < \begin{cases} \frac{1-\gamma}{2\lambda^\infty}\epsilon & \text{if } \gamma < 1, \\ \frac{\epsilon}{(r_{\max}-r_{\min})(H+1)H} & \text{if } \gamma = 1. \end{cases} \quad (38)$$

Mais quel est l'effet de régler ρ à de petites ou grandes valeurs ?

— Plus ρ est petit, plus $thr(\tau)$ est grand, plus courtes sont les trajectoires, mais plus petites sont les boules et plus grand devient la densité de points nécessaire autour de la trajectoire optimale, ce qui implique un plus grand nombre de trajectoires requises pour mener à la convergence.

— Plus ρ est grand, plus $thr(\tau)$ est petit, plus longues sont les trajectoires, mais plus grandes sont les boules et moins importante devient la densité de points nécessaire autour de la trajectoire optimale, ce qui implique un plus petit nombre de trajectoires pour mener à la convergence.

5.2 Convergence en temps fini

Theorem 4. *zs-OMG-HSVI (algorithme 1) termine en temps fini avec une ϵ-approximation de $V_0^*(\sigma_0)$.*

Démonstration. (ébauche adaptée de HORÁK et BOŠANSKÝ [12]) Supposons que l'algorithme ne termine pas et génère un nombre infini d'essais (trajectoires) exploratoires. Alors, le nombre d'essais de longueur T (pour un certain $0 \leq T \leq H$) doit être infini. Il est impossible de faire tenir un nombre infini de points d'occupation σ_T satisfaisant $\|\sigma_T - \sigma'_T\|_1 > \rho$ à l'intérieur de $\mathcal{O}_T^\sigma$. Il doit donc y avoir deux essais de longueur T, $\{\sigma_{\tau,1}\}_{\tau=0}^T$ et $\{\sigma_{\tau,2}\}_{\tau=0}^T$, tels que $\|\sigma_{T-1,1} - \sigma_{T-1,2}\|_1 \leq \rho$. On peut ainsi montrer (comme fait en annexe C.1) que le second essai ne devrait pas avoir eu lieu. □

Note : La borne sur le nombre d'itérations dépend du nombre de boules de rayon ρ requis pour couvrir le simplexe des états d'occupation à chaque profondeur.
Par ailleurs, le lemme suivant permet de résoudre aussi des problèmes à horizon infini (quand $\gamma < 1$) en bornant la longueur des trajectoires utilisant la largeur bornée de $\hat{V}$ et la croissance exponentielle de $thr(\tau)$.

Lemma 6. *Quand $\gamma < 1$, en utilisant la constante de Lipschitz indépendante de la profondeur λ^∞, et avec la largeur maximale entre initialisations $W \stackrel{\text{def}}{=} \|\overline{V}^{(0)} - \underline{V}^{(0)}\|_\infty$, la longueur des trajectoires est majorée par*

$$T_{\max} \stackrel{\text{def}}{=} \left\lceil \log_\gamma \frac{\epsilon - \frac{2\rho\lambda^\infty}{1-\gamma}}{W - \frac{2\rho\lambda^\infty}{1-\gamma}} \right\rceil. \quad (39)$$

5.3 Exécution

Comme déjà mentionné, toute stratégie δ_τ^2 dans un tuple stocké garantit au plus (c'est-à-dire «au pire» du point de vue de 2) un retour espéré $\sigma_\tau^{m,1} \cdot \overline{\nu}_{\tau:}^2$ si dans l'état d'occupation associé σ_τ, quelle que soit la stratégie de 1. En particulier, δ_0^2 induit une stratégie qui garantit au pire le retour espéré $\sigma_0^{m,1} \cdot \overline{\nu}_0^2$ si en σ_0 (qui correspond toujours à l'état d'occupation initial). Au moment de l'exécution, si 2 emploie une stratégie $\delta_0^{2,\star} \in \arg\max_{\langle\sigma_0,\langle\overline{\nu}_0,\delta_0^2\rangle,-,-\rangle} \sigma_0^{m,1} \cdot \overline{\nu}_0$, alors son retour espéré est au plus $\overline{V}_0(\sigma_0)$ ($\leq V_0^*(\sigma_0) + \epsilon$) (quelle que soit la stratégie de 1).
Ainsi, résoudre le zs-OMG dérivé fournit à chaque joueur une stratégie solution du zs-POSG original, et chaque joueur peut dériver sa stratégie par lui-même puisque les stratégies à l'équilibre de Nash sont interchangeables (pas besoin de la coordination d'un planificateur central comme pour les Dec-POMDP). Par exemple, l'algo. 1 renvoie une stratégie solution seulement pour 1.

6 Expérimentations

Cette section présente des expérimentations préliminaires pour valider l'approche proposée avec une première version de zsOMG-HSVI. Des résultats complémentaires apparaissent aussi en annexe D.

6.1 Configuration

Bancs d'essai Quatre bancs d'essai ont été utilisés. Recycling Robot et Mabc sont des bancs d'essai Dec-POMDP bien connus (*cf.* `http://masplan.org`) et ont été adaptés à notre cadre compétitif en faisant minimiser (plutôt que maximiser) la fonction objectif par le joueur 2. Competitive Tiger et Adversarial Tiger ont été introduits par WIGGERS [26]. Nous ne considérons que des horizons finis et $\gamma = 1$.

Algorithmes Nous comparons nos algorithmes, c'est-à-dire la version reposant seulement sur la Lipschitz continuité (dénotée OMGHSVI$^{\text{LC}}$) et la version reposant aussi sur la concavité et la convexité (dénotée OMGHSVI$^{\text{LC}}_{\text{CC}}$), avec l'algorithme de l'état de l'art Sequence Form [15], et deux approches heuristiques, *Informed* et *Random*, proposées par WIGGERS [26] et s'appuyant sur les propriétés de concavité et de convexité de la fonction de valeur.
OMGHSVI$^{\text{LC}}$ a été exécuté avec une erreur cible ϵ spécifiée dans la table, $\lambda_\tau = H \cdot (r_{\max} - r_{\min})$, et sans élagage. OMGHSVI$^{\text{LC}}_{\text{CC}}$ a été exécuté avec une erreur $\epsilon = 0.0001$, $\lambda_\tau = H \cdot (r_{\max} - r_{\min})$, ρ au milieu de son intervalle de faisabilité, et un élagage de $\overline{V}_\tau$ et $\underline{V}_\tau$, mais pas de $\overline{W}_\tau^1$ ou $\underline{W}_\tau^2$. Pour les Dec-POMDP, un ingrédient clef du passage à l'échelle de FB-HSVI est la compression sans perte des historiques d'action-observation équivalents dans les états

d'occupation, ce qui réduit leur dimensionalité [9]. Dans notre contexte, nous appliquons plus précisément la compression LPE à σ_τ, laissant la compression plus forte pour des travaux futurs.

Nous avons lancé les expérimentations sur une machine Ubuntu avec processeur Intel i7-10810U 1.10 GHz et 16 GB de mémoire vive disponible. Le code source sera rendu disponible en source ouverte.

6.2 Résultats

Une première observation est que OMGHSVI$^{\text{LC}}_{\text{CC}}$ et OMGHSVI$^{\text{LC}}$ maintiennent tous deux des majorant et minorant valides de la valeur optimale en σ_0, et réduisent l'écart progressivement [2] (*cf.* figures en anx. D). Le tableau 1 montre que (i) OMGHSVI$^{\text{LC}}_{\text{CC}}$ est toujours meilleur que OMGHSVI$^{\text{LC}}$ et les heuristiques de WIGGERS [26], et (ii) à moins de manquer de mémoire, Sequence Form surpasse toujours OMGHSVI$^{\text{LC}}_{\text{CC}}$. Par contre, la compression LPE permet à OMGHSVI$^{\text{LC}}_{\text{CC}}$ d'exploiter la structure de certains jeux et ainsi de générer des trajectoires même pour de grands horizons (par exemple dans Recycling Robot pour $H = 6$, *cf.* anx. D). Plus généralement, nous observons que le nombre d'itérations effectuées par OMGHSVI$^{\text{LC}}_{\text{CC}}$ en 24 h est fortement corrélé avec la qualité de la compression LPE (la compression est maximale pour Recycling Robot compresses et minimale pour Adversarial Tiger). Comme attendu, OMGHSVI$^{\text{LC}}$ s'avère très lent, ne terminant même pas la première itération dans la plupart des cas.

TABLE 1 – Expérimentations comparant 4 solveurs sur divers bancs d'essai. Les valeurs reportées sont les temps de calcul, ou les [écarts] $[\overline{V}_0(\sigma_0) - \underline{V}_0(\sigma_0)]$ si la limite de 24 h est atteinte. «(ni)» (*no improvement*) indique aucune amélioration par rapport aux initialisations en 24 h . «xx» indique un dépassement mémoire. «n/a» indique un résultat non disponible.

Competitive Tiger	H=2	H=3	H=4	H=5
Wiggers Random	[0.56]	[2.67]	[5.81]	[6.97]
Wiggers Informed	[2.07]	[2.33]	[3.61]	xx
OMGHSVI$^{\text{LC}}(1.0)$	[2.8]	(ni)	(ni)	(ni)
OMGHSVI$^{\text{LC}}_{\text{CC}}$	2s	[0.02]	[2.55]	[5.82]
Sequence Form	0.14 s	48 s	14 min	2.5 h
Adversarial Tiger	**H=2**	**H=3**	**H=4**	**H=5**
Wiggers Random	[0.04]	[0.38]	[0.92]	[2.07]
Wiggers Informed	[0.59]	[1.32]	[1.79]	[3.34]
OMGHSVI$^{\text{LC}}(0.1)$	22 min	(ni)	(ni)	(ni)
OMGHSVI$^{\text{LC}}_{\text{CC}}$	1 s	10 s	[0.4]	[1.70]
Sequence Form	0.02 s	0.17 s	3 s	107 s
Recycling Robot	**H=3**	**H=4**	**H=5**	**H=6**
OMGHSVI$^{\text{LC}}$	(ni)	(ni)	(ni)	(ni)
OMGHSVI$^{\text{LC}}_{\text{CC}}$	3 min	[0.01]	[0.93]	[2.67]
Sequence Form	1 s	10 s	1.5 h	xx
Mabc	**H=2**	**H=3**	**H=4**	**H=5**
OMGHSVI$^{\text{LC}}(0.05)$	2s	(ni)	(ni)	(ni)
OMGHSVI$^{\text{LC}}_{\text{CC}}$	0.6 s	17 min	[0.11]	n/a
Sequence Form	0.1 s	1 s	3 s	181 s

7 Discussion

Inspiré par des techniques de résolution de l'état de l'art pour POMDP et Dec-POMDP, nous résolvons ici des zs-POSG en le convertissant en des jeux de Markov à somme nulle sur les états d'occupation, c'est-à-dire des jeux complètement observables qui permettent d'exploiter le principe d'optimalité de Bellman. Nous étendons les propriétés de concavité, convexité et Lipschitz-continuité de V^* et Q^*, et en tirons partie pour proposer des approximateurs à base de points majorant et minorant, ainsi que des opérateurs de mise-à-jour efficace reposant sur la programmation linéaire. Cela permet de dériver une variante de HSVI qui converge en temps fini vers une solution ϵ-optimale, fournissant des stratégies solution (sûres) sous une forme récursive comme sous-produit du processus de résolution. Les expérimentations confirment la faisabilité de cette approche et montrent des résultats améliorés par rapport à des heuristiques apparentées (s'appuyant aussi sur les propriétés de concavité et de convexité).

Cette approche ouvre la voie à une vaste famille de solveurs puisque de nombreuses variantes pourraient être envisagées, par exemple, en employant différents schémas algorithmiques, différents approximateurs, différents opérateurs de mise-à-jour ou différentes techniques d'élagage. Nous avons ainsi aussi évalué une variante ne reposant que sur la Lipschitz-continuité de V^*.

Les travaux futurs incluent : chercher de meilleures initialisations et constantes de Lipschitz, par exemple avec des initialisations POMDP plus avancées ou en s'appuyant sur les One-Sided zsPOSG ; proposer une méthode d'élagage pour $\overline{W}^1_\tau$ et $\underline{W}^2_\tau$; employer des méthodes d'oracle ou d'autres heuristiques pour résoudre des jeux locaux plus vite ; exploiter une compression TPE plutôt que LPE ; et brancher sur des observations publiques (voire sur des informations publiques révélées par la structure de l'état d'occupation).

2. Notons que les écarts ne sont pas comparables entre bancs d'essai puisqu'ils ne sont pas normalisés.

Références

[1] K. ÅSTRÖM. « Optimal control of Markov processes with incomplete state information ». *Journal of Mathematical Analysis and Applications* 10.1 (1965), p. 174-205. ISSN : 0022-247X.

[2] N. BASILICO, G. DE NITTIS et N. GATTI. « A Security Game Combining Patrolling and Alarm–Triggered Responses Under Spatial and Detection Uncertainties ». Dans : *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[3] A. BASU et L. STETTNER. « Finite- and Infinite-Horizon Shapley Games with Nonsymmetric Partial Observation ». *SIAM Journal on Control and Optimization* 53.6 (2015), p. 3584-3619.

[4] B. BOŠANSKÝ, C. KIEKINTVELD, V. LISÝ et M. PĚCHOUČEK. « An Exact Double-Oracle Algorithm for Zero-Sum Extensive-Form Games with Imperfect Information ». *Journal of Artificial Intelligence Research* 51 (2014), p. 829-866. DOI : 10.1613/jair.4477.

[5] N. BROWN et T. SANDHOLM. « Superhuman AI for Heads-Up No-Limit Poker : Libratus Beats Top Professionals ». *Science* 359.6374 (2018), p. 418-424.

[6] O. BUFFET, J. DIBANGOYE, A. DELAGE, A. SAFFIDINE et V. THOMAS. « On Bellman's Optimality Principle for zs-POSGs ». *Computing Research Repository* abs/2006.16395 (2020). arXiv : 2006.16395 [cs.AI].

[7] K. CHATTERJEE et L. DOYEN. « Partial-Observation Stochastic Games : How to Win When Belief Fails ». Dans : t. 15. 2. 2014, p. 16.

[8] H. L. COLE et N. KOCHERLAKOTA. « Dynamic Games with Hidden actions and Hidden States ». *Journal of Economic Theory* 98.1 (2001), p. 114-126.

[9] J. DIBANGOYE, C. AMATO, O. BUFFET et F. CHARPILLET. « Optimally Solving Dec-POMDPs as Continuous-State MDPs ». *Journal of Artificial Intelligence Research* 55 (2016), p. 443-497.

[10] M. K. GHOSH, D. MCDONALD et S. SINHA. « Zero-Sum Stochastic Games with Partial Information ». *Journal of Optimization Theory and Applications* 121.1 (avr. 2004), p. 99-118.

[11] E. A. HANSEN, D. BERNSTEIN et S. ZILBERSTEIN. « Dynamic Programming for Partially Observable Stochastic Games ». Dans : *Proceedings of the Nineteenth National Conference on Artificial Intelligence*. San Jose, CA, 2004.

[12] K. HORÁK et B. BOŠANSKÝ. « Solving Partially Observable Stochastic Games with Public Observations ». Dans : *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. 2019, p. 2029-2036.

[13] K. HORÁK, B. BOŠANSKÝ et M. PĚCHOUČEK. « Heuristic Search Value Iteration for One-Sided Partially Observable Stochastic Games ». Dans : *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 2017, p. 558-564.

[14] D. KOLLER et N. MEGIDDO. « The Complexity of Two-Person Zero-Sum Games in Extensive Form ». *Games and Economic Behavior* 4.4 (1992), p. 528-552. DOI : https://doi.org/10.1016/0899-8256(92)90035-Q.

[15] D. KOLLER, N. MEGIDDO et B. von STENGEL. « Efficient Computation of Equilibria for Extensive Two-Person Games ». *Games and Economic Behavior* 14.51 (1996), p. 220-246.

[16] C. KROER, K. WAUGH, F. KILINÇ-KARZAN et T. SANDHOLM. « Faster Algorithms for Extensive-Form Game Solving via Improved Smoothing Functions ». *Mathematical Programming* 179 (2020), p. 385-417. DOI : 10.1007/s10107-018-1336-7.

[17] H. W. KUHN. « Simplified Two-Person Poker ». Dans : *Contributions to the Theory of Games*. Sous la dir. de H. W. KUHN et A. W. TUCKER. T. 1. Princeton University Press, 1950.

[18] O. MADANI, S. HANKS et A. CONDON. « On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems ». Dans : *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. 1999.

[19] J. von NEUMANN. « Zur Theorie der Gesellschaftsspiele ». *Math. Annalen* 100 (1928).

[20] F. OLIEHOEK et N. VLASSIS. *Dec-POMDPs and extensive form games : equivalence of models and algorithms*. Rapp. tech. IAS-UVA-06-02. Intelligent Systems Laboratory Amsterdam, University of Amsterdam, 2006.

[21] Y. SHOHAM et K. LEYTON-BROWN. *Multiagent Systems : Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009. ISBN : 978-0-521-89943-7.

[22] T. SMITH. « Probabilistic Planning for Robotic Exploration ». Thèse de doct. The Robotics Institute, Carnegie Mellon University, 2007.

[23] T. SMITH et R. SIMMONS. « Point-Based POMDP Algorithms : Improved Analysis and Implementation ». Dans : *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. 2005, p. 542-549.

[24] B. von STENGEL. « Efficient Computation of Behavior Strategies ». *Games and Economic Behavior* 14.50 (1996), p. 220-246.

[25] D. SZER, F. CHARPILLET et S. ZILBERSTEIN. « MAA* : A Heuristic Search Algorithm for Solving Decentralized POMDPs ». Dans : *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. 2005, p. 576-583.

[26] A. WIGGERS. « Structure in the Value Function of Two-Player Zero-Sum Games of Incomplete Information ». Mém. de mast. University of Amsterdam, 2015.

[27] A. WIGGERS, F. OLIEHOEK et D. ROIJERS. « Structure in the Value Function of Two-Player Zero-Sum Games of Incomplete Information ». *Computing Research Repository* abs/1606.06888 (2016).

[28] A. WIGGERS, F. OLIEHOEK et D. ROIJERS. « Structure in the Value Function of Two-player Zero-sum Games of Incomplete Information ». Dans : *Proceedings of the Twenty-second European Conference on Artificial Intelligence*. The Hague, The Netherlands, 2016, p. 1628-1629. DOI : 10.3233/978-1-61499-672-9-1628.

[29] M. ZINKEVICH, M. JOHANSON, M. BOWLING et C. PICCIONE. « Regret Minimization in Games with Incomplete Information ». Dans : *Advances in Neural Information Processing Systems 20*. 2007.

Les annexes qui suivent fournissent :

— quelques résultats théoriques utiles (sans démonstrations et sans commentaires), y compris en répétant des résultats présentés dans le corps de l'article, et

— quelques résultats expérimentaux supplémentaires.

A État de l'art

A.1 Résoudre des POSG comme des Occupancy MG

Lemma 1. (énoncé initialement en page 4) *$\sigma_{\boldsymbol{\beta}_{0:\tau-1}}$, pris avec $\boldsymbol{\beta}_\tau$, est une statistique suffisante pour calculer (i) le prochain état d'occupation, $\sigma_{\boldsymbol{\beta}_{0:\tau}}$, et (ii) l'espérance de récompense à τ :* $\mathbb{E}\left[R_\tau \mid \boldsymbol{\beta}_{0:\tau-1} \oplus \boldsymbol{\beta}_\tau\right]$.

A.2 Introduction des jeux locaux

Lemma 7. *$T^1_m(\sigma_\tau, \boldsymbol{\beta}_\tau)$ est linéaire en σ_τ et β^1_τ.*

Lemma 8. *$T^1_c(\sigma_\tau, \boldsymbol{\beta}_\tau)$ est indépendant de β^1_τ.*

Lemma 2. (énoncé initialement en page 5) *$Q^*_\tau(\sigma_\tau, \boldsymbol{\beta}_\tau)$ est concave en β^1_τ et convexe en β^2_τ.*

B Propriétés et approximation de V^*

B.1 Propriétés de V^*

Linéarité et Lipschitz-continuité de $T(\sigma_\tau, \beta^1_\tau, \beta^2_\tau)$.

Lemma 3. (énoncé initialement en page 5) *À profondeur τ, $T(\sigma_\tau, \boldsymbol{\beta}_\tau)$ est linéaire en β^1_τ, β^2_τ, et σ_τ, où $\boldsymbol{\beta}_\tau = \langle \beta^1_\tau, \beta^2_\tau \rangle$. Elle est plus précisément 1-Lipschitz-continue (1-LC) en σ_τ (en norme 1), c'est-à-dire que, pour tous σ_τ, σ'_τ :*

$$\|T(\sigma'_\tau, \boldsymbol{\beta}_\tau) - T(\sigma_\tau, \boldsymbol{\beta}_\tau)\|_1 \leq 1 \cdot \|\sigma'_\tau - \sigma_\tau\|_1. \tag{14}$$

Lipschitz-Continuité de V^*.

Lemma 4. (énoncé initialement en page 5) *À profondeur τ, $V_\tau(\sigma_\tau, \boldsymbol{\beta}_{\tau:})$ est linéaire en σ_τ.*

Theorem 2. (énoncé initialement en page 5) *Soit $h_\tau \stackrel{\text{def}}{=} \frac{1-\gamma^{H-\tau}}{1-\gamma}$ si $\gamma < 1$, sinon $h_\tau \stackrel{\text{def}}{=} H - \tau$ (si $\gamma = 1$). Alors $V^*_\tau(\sigma_\tau)$ est λ_τ-Lipschitz-continue en σ_τ à toute profondeur $\tau \in \{0\,..\,H-1\}$, où $\lambda_\tau = \frac{1}{2} h_\tau \, (r_{\max} - r_{\min})$.*

B.2 Approximateurs

$\overline{W}^1_\tau$ et $\underline{W}^2_\tau$.

Lemma 9. *Considérant que les vecteurs $\nu_{[\sigma^{c,1}_H, \beta^2_{H:}]}$ sont des vecteurs nuls, nous avons, pour tout $\tau \in \{0..H-1\}$:*

$$W^{1,*}_\tau(\sigma_\tau, \beta^1_\tau) = \min_{\beta^2_\tau, \langle \beta^2_{\tau+1:}, \nu_{[T^1_c(\sigma_\tau, \beta^2_\tau), \beta^2_{\tau+1:}]} \rangle} \beta^1_\tau \cdot \left[r(\sigma_\tau, \cdot, \beta^2_\tau) + \gamma T^1_m(\sigma_\tau, \cdot, \beta^2_\tau) \cdot \nu_{[T^1_c(\sigma_\tau, \beta^2_\tau), \beta^2_{\tau+1:}]} \right]. \tag{40}$$

C HSVI pour zs-POSG

Lemma 5. (énoncé initialement en page 8) *En bornant λ_τ par $\lambda^\infty = \frac{1}{2} \frac{1}{1-\gamma} \left[r_{\max} - r_{\min} \right]$ quand $\gamma < 1$, et en notant que*

$$thr(\tau) = \begin{cases} \gamma^{-\tau} \epsilon - 2\rho\lambda^\infty \frac{\gamma^{-\tau}-1}{1-\gamma} \text{ if } \gamma < 1, \text{ else} \\ \epsilon - \rho(r_{\max} - r_{\min})(2H + 1 - \tau)\tau, \end{cases} \tag{37}$$

on peut garantir la positivité du seuil en tout $\tau \in 1\,..\,H-1$ en contraignant

$$0 < \rho < \begin{cases} \frac{1-\gamma}{2\lambda^\infty} \epsilon & \text{if } \gamma < 1, \\ \frac{\epsilon}{(r_{\max} - r_{\min})(H+1)H} & \text{if } \gamma = 1. \end{cases} \tag{38}$$

Lemma 6. (énoncé initialement en page 8) *Quand $\gamma < 1$, en utilisant la constante de Lipschitz indépendante de la profondeur λ^∞, et avec la largeur maximale entre initialisations $W \stackrel{\text{def}}{=} \|\overline{V}^{(0)} - \underline{V}^{(0)}\|_\infty$, la longueur des trajectoires est majorée par*

$$T_{\max} \stackrel{\text{def}}{=} \left\lceil \log_\gamma \frac{\epsilon - \frac{2\rho\lambda^\infty}{1-\gamma}}{W - \frac{2\rho\lambda^\infty}{1-\gamma}} \right\rceil. \tag{39}$$

TABLE 2 – Nombres d'états/actions/observations pour chaque banc d'essai

	$\mathcal{S}$	$\mathcal{A}^1$	$\mathcal{A}^2$	$\mathcal{O}^1$	$\mathcal{O}^2$
Competitive Tiger	2	4	4	3	3
Adversarial Tiger	2	3	2	2	2
Recycling Robot	4	3	3	2	2
Mabc	4	2	2	2	2

C.1 Convergence

Lemma 10. *Soit $(\sigma_0, \ldots, \sigma_{\tau+1})$ une trajectoire complète générée par zs-OMG-HSVI et $\boldsymbol{\beta}_\tau$ la règle de décision comportementale jointe qui a induit la dernière transition, c'est-à-dire que $\sigma_{\tau+1} = T(\sigma_\tau, \boldsymbol{\beta}_\tau)$. Alors, après mise-à-jour de $\overline{W}^1_\tau$ et $\underline{W}^2_\tau$, nous avons que $\overline{W}^1_\tau(\sigma_\tau, \beta^1_\tau) - \underline{W}^2_\tau(\sigma_\tau, \beta^2_\tau) \leq \gamma thr(\tau + 1)$.*

Lemma 11 (Évolution monotone de $\overline{W}^1_\tau$ et $\underline{W}^2_\tau$)**.** *Soient $K\overline{W}^1_\tau$ et $K\underline{W}^2_\tau$ les approximateurs après une mise-à-jour en σ_τ avec les règles de décision comportementales $\langle \overline{\beta}^1_\tau, \underline{\beta}^2_\tau \rangle$ (respectivement associées aux vecteurs $\overline{\nu}^2_{\tau+1}$ et $\underline{\nu}^1_{\tau+1}$). Soient aussi $K^{(n+1)}\overline{W}^1_\tau$ et $K^{(n+1)}\underline{W}^2_\tau$ les mêmes approximateurs après n autres mises-à-jour (dans divers états d'occupation). Alors,*

$$\max_{\beta^1_\tau} K^{(n+1)}\overline{W}^1_\tau(\sigma_\tau, \beta^1_\tau) \leq \max_{\beta^1_\tau} K\overline{W}^1_\tau(\sigma_\tau, \beta^1_\tau) \leq \overline{W}^1_\tau(\sigma_\tau, \overline{\beta}^1_\tau) \quad \textit{et} \tag{41}$$

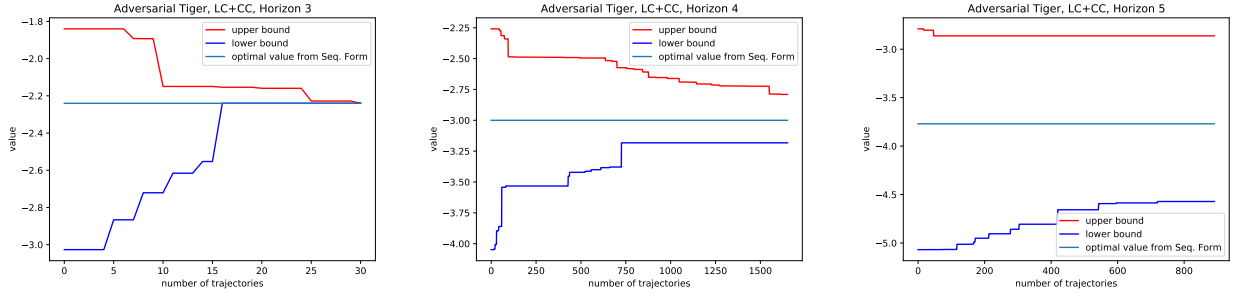
$$\min_{\beta^2_\tau} K^{(n+1)}\underline{W}^2_\tau(\sigma_\tau, \beta^2_\tau) \geq \min_{\beta^2_\tau} K\underline{W}^2_\tau(\sigma_\tau, \beta^2_\tau) \geq \underline{W}^2_\tau(\sigma_\tau, \underline{\beta}^2_\tau). \tag{42}$$

Lemma 12. *Après avoir mis-à-jour, dans l'ordre, $\overline{W}^1_\tau$ et $\overline{V}_\tau$, nous avons $K\overline{V}_\tau(\sigma_\tau) \leq \max_{\beta^1_\tau} K\overline{W}^1_\tau(\sigma_\tau, \beta^1_\tau)$.*
Après avoir mis-à-jour, dans l'ordre, $\underline{W}^2_\tau$ et $\underline{V}_\tau$, nous avons $K\underline{V}_\tau(\sigma_\tau) \geq \min_{\beta^2_\tau} K\underline{W}^2_\tau(\sigma_\tau, \beta^2_\tau)$.

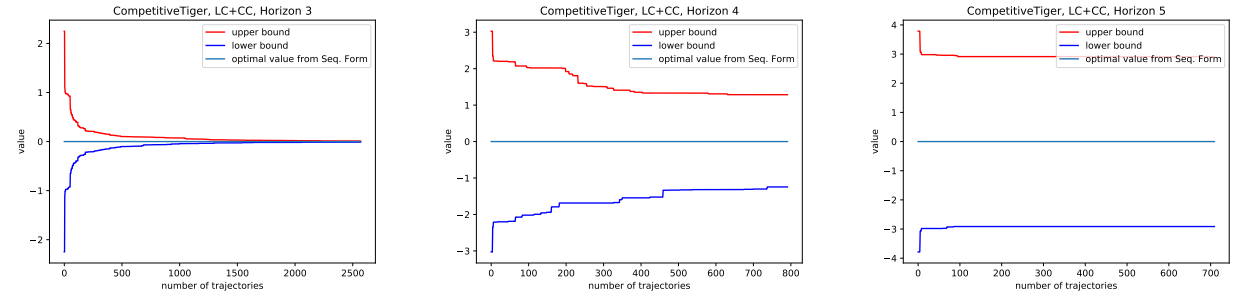
Theorem 4. (énoncé initialement en page 8) *zs-OMG-HSVI (algorithme 1) termine en temps fini avec une ϵ-approximation de $V^*_0(\sigma_0)$.*

D Expérimentations

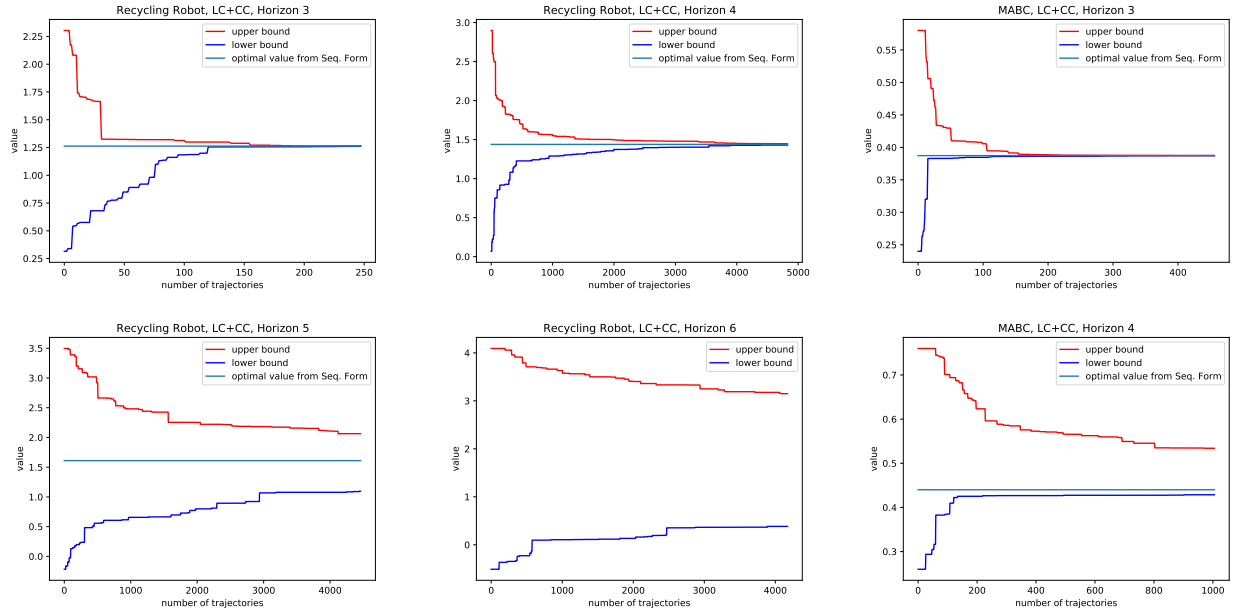
Cette section fournit (i) des informations concernant les bancs d'essai considérés dans le tableau 2 et (ii) des résultats supplémentaires.
Les graphes de la figure 1 montrent comment les valeurs majorantes et minorantes en σ_0 (c'est-à-dire $\overline{V}_0(\sigma_0)$ et $\underline{V}_0(\sigma_0)$) évoluent en fonction du nombre d'itérations, ici en considérant les mêmes bancs d'essai et horizons temporels que dans la section 6 (sauf pour $H = 2$).
Comme attendu, ces bornes convergent de manière monotone vers la valeur optimale (ici fournies par Sequence Form dans tous les cas sauf Recycling Robot pour $H = 6$). Cette convergence serait symétrique dans Competitive Tiger, le seul jeu symétrique, si l'algorithme brise les symétries de manière biaisé quand plusieurs situations équivalentes sont possibles.
Notons aussi qu'on s'attend à ce que les durées des itérations augmentent avec l'horizon temporel, de sorte qu'il est surprenant d'observer, à la fois dans Competitive Tiger et Recycling Robot, des nombres similaires d'itérations en 24 heures pour les horizons au-delà de 4. Ce phénomène est actuellement en cours d'étude.



(a) Adversarial Tiger



(b) Competitive Tiger



(c) Recycling Robot (d) Mabc

FIGURE 1 – Évolution des valeurs majorante et minorante $\overline{V}_0(\sigma_0)$ (en rouge) et $\underline{V}_0(\sigma_0)$ (en bleu) de OMGHSVI$^{\text{LC}}_{\text{CC}}$ pour les différents bancs d'essai en fonction du nombre d'itérations (de trajectoires générées). Valeur optimale trouvée par Sequence Form en vert pour référence (si disponible).

Explicit Representations of Persistency for Propositional Action Theories

Sergej Scheck Alexandre Niveau Bruno Zanuttini

Normandie Univ.; UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

sergej.scheck,alexandre.niveau,bruno.zanuttini@unicaen.fr

Résumé

Nous envisageons d'enrichir la représentation des actions en logique propositionnelle par des opérateurs syntaxiques pour représenter la persistance des variables. Ceci est motivé par le fait que le problème du cadre n'est pas résolu de manière satisfaisante par les langages propositionnels, tels que les diagrammes de décision binaires ou DNF. Nous introduisons deux de ces opérateurs, permettant de représenter différents types de persistance, et considérons les langages obtenus à partir de la logique propositionnelle en les ajoutant à n'importe quel niveau d'imbrication. Nous étudions les langages résultantes du point de vue de leur concision relative et de la complexité de la décision de successeur. Nous montrons une image intéressante de divers résultats de complexité.

Mots Clef

Planification, compilation de connaissances, théories d'actions, persistance, problème du cadre, circonscription

Abstract

We consider enriching the representation of actions in propositional logic by syntactic operators for representing the persistency of variables. This is motivated by the fact that the frame problem is not satisfactorily solved by propositional languages, such as binary decision diagrams or DNF. We introduce two such operators, allowing to represent different kinds of persistency, and consider the languages obtained from propositional logic by adding them at any level of nesting. We study the resulting languages from the point of view of their relative succinctness and the complexity of deciding successorship. We show an interesting picture of diverse complexity results.

Keywords

Planning, knowledge compilation, action theories, persistency, frame problem, circumscription

1 Introduction

In automated planning, a central aspect of the description of problems is the formal representation of actions. Such representations are needed for specifying the available actions, and for the planners to operate on them while computing a plan. **PDDL** [15] is a standard language for this. More generally, the formal representation of actions is central to reasoning about actions, programs, and change using logic. Frameworks for this include proposals as diverse as the Situation Calculus [18], the Event Calculus [13], **PDL** [9], **DL-PA** [3], and many others.

We view such languages as either *imperative* or *declarative*. Declarative languages allow one to specify the properties of situations, actions, events, while imperative languages concentrate on how the effects are brought about. In this view, the Situation and Event Calculi are declarative, as well as **PDL**, while **PDDL** and **DL-PA** are imperative.

On the other hand, a well-known question when representing action and change is how to succinctly specify the *non-effects* of actions, that is, to ensure that the specification precludes fluents to change value while this is not intended. This is known as the *frame problem*. While imperative languages naturally come with a solution to the frame problem, because operational semantics literally transform a situation into another one, the frame problem is crucial in declarative languages, and it has been thoroughly studied, in particular for the Situation Calculus [18].

We are interested here in the frame problem for actions specified in the simple language of *propositional action theories*, that is, as Boolean formulas describing the possible combinations of values for fluents *before* and *after* the action is taken. Though this language is very simple, it is indeed used in automated planning, because it makes operations on sets of states (aka *belief states*) conceptually simple [6, 5, 20].

We consider action theories represented in Negation Normal Form (NNF), which encompasses representations usually used like ordered binary decision diagrams or formulas in disjunctive normal form. Such theories are adequate for representing (purely) nondeterministic actions, which lie at the core of fully observable nondeterministic planning and conformant planning [19, 1, 10, 16, 20, 11]. Our contribution is to propose two different operators for representing the persistency of fluents, and to consider the language of NNF actions theories as enriched by one or the other. The originality of this contribution lies in the facts that (i) the language is *enriched* with an operator, which, we argue, allows one to specify the persistency of variables more naturally and succinctly than expressions in the plain underly-

ing logic (like successor-state axioms for the Situation Calculus), and (ii) we allow the operators to occur anywhere in the formula, including nested occurrences, which again facilitates the description of actions.
Our operators are one whose interpretation is dependent on the *syntax* of the action in its scope, and one whose interpretation depends only on its semantics (as a relation between the states before and after the action). The "semantic" one corresponds to interpreting the action in its scope under *circumscription* [14], here used as a semantics of minimal change through the action.
We consider the resulting extensions of the language of NNF action theories in the formal framework of the knowledge compilation map [7]. This framework deals with the study of formal languages under the point of view of queries (how efficient is it to answer various queries depending on the language?), transformations (how efficient is it to transform or combine different representations in a given language?), and succinctness (how concise is it to represent knowledge in each language?). We focus on queries related to automated planning (deciding whether the action can lead from a state to another one, whether it is applicable at some state) and on succinctness issues.
Naturally, there is a tradeoff between tractability of queries and succinctness of the languages. We give a complete picture for the languages which we consider. Precisely, we show that the syntactic operator can be added to NNF action theories without changing complexity of queries nor succinctness, since it can be compiled away in polynomial time; this shows that actions can be specified in the richer language without harming further calculations. On the other hand, we show that the semantic operator yields a more succinct language when allowed only at the root of expressions, and even more succinct when allowed everywhere, but that the complexity of answering queries increases accordingly.

2 Preliminaries

We consider a countable set of propositional *state* variables $\mathbb{P} = \{p_i \mid i \in \mathbb{N}\}$. Let $P \subset \mathbb{P}$ be a finite set of state variables; a subset of P is called a *P-state*, or simply a *state*. The intended interpretation of a state $s \in 2^P$ is the assignment to P in which all variables in s are true, and all variables in $P \setminus s$ are false. For instance, for $P = \{p_1, p_2, p_3\}$, $s = \{p_1, p_3\}$ denotes the state in which p_1, p_3 are true and p_2 is false. We write $\mathrm{V}(\varphi)$ for the set of variables occuring in an expression φ; note that expressions may involve both variables in $\mathbb{P}$ and variables not in $\mathbb{P}$, so in general we do *not* have $\mathrm{V}(\varphi) \subseteq \mathbb{P}$.

Actions We consider (purely) nondeterministic actions, *i.e.*, actions with which a single state may have several successors.

Definition 1. Let $P \subset \mathbb{P}$ be a finite set of variables. A *P-action* is a mapping a from 2^P to $2^{(2^P)}$. The states in $a(s)$ are called *a-successors* of s.

Note that $a(s)$ is defined for all states s. We will consider a to be *applicable* in s if and only if $a(s) \neq \emptyset$ holds.

Definition 2. An *action language* is an ordered pair $\langle L, I\rangle$, where L is a set of expressions and I is a partial function on $L \times 2^{\mathbb{P}}$ such that, when defined for $\alpha \in L$ and $P \subset \mathbb{P}$, $I(\alpha, P)$ is a P-action.

We call the expressions in L *action descriptions*, and call I the *interpretation function* of the language. Observe that those sets P's such that $I(\alpha, P)$ is defined are *a priori* not related to $\mathrm{V}(\alpha)$; α may involve auxiliary variables (not in $\mathbb{P}$) which are not part of the state descriptions, and dually, a state may assign variables of $\mathbb{P}$ which do not occur in α. If L, I, P are fixed or clear from the context, then we write $\alpha(s)$ instead of $I(\alpha, P)(s)$ for the set of all α-successors of s. In this case we call P the *scope* of α. In this article we will consider action descriptions α which are intendedly constructed to ensure that $I(\alpha, P)$ is defined.

Definition 3. A *translation* from an action language $\langle L_1, I_1\rangle$ to another language $\langle L_2, I_2\rangle$ is a function $f : L_1 \times 2^{\mathbb{P}} \to L_2$ satisfying $I_1(\alpha, P) = I_2(f(\alpha, P), P)$ for all $\alpha \in L_1$ and $P \subset \mathbb{P}$ such that $I_1(\alpha, P)$ is defined.

In words, this means that the L_1-expression α and the L_2-expression $f(\alpha, P)$ describe the same P-action. Again, when P is clear from the context, we write $f(\alpha)$ for $f(\alpha, P)$. The translation f is said to be *polynomial-time* if it can be computed in time polynomial in the size of α and P, and *polynomial-size* if the size of $f(\alpha, P)$ is bounded by a fixed polynomial in the size of α and P. Clearly, a polynomial-time translation is necessarily also a polynomial-size one, but the converse is not true in general.

Logic A Boolean formula φ over a set Q of variables is in *negation normal form* (NNF) if it is built up from literals using conjunctions and disjunctions, *i.e.*, if it is generated by the grammar $\varphi ::= q \mid \neg q \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$, where q ranges over Q. Similarly to other expressions, Q may involve state variables (in $\mathbb{P}$) and other variables (not in $\mathbb{P}$).
It is important to note that a formula φ with $\mathrm{V}(\varphi) \subseteq Q$ for some set of variables Q can be viewed as a formula over Q (and the truth value of the corresponding Boolean function does not depend on the variables in $Q \setminus \mathrm{V}(\varphi)$). For a Boolean formula φ over Q and an assignment t to the variables in Q, we write $t \models \varphi$ if φ evaluates to true under the assignment t.
For readability, we sometimes use the symbols $\leftrightarrow$ and $\rightarrow$ in Boolean formulas and still call them **NNF** formulas. Indeed, it will always be the case that there are equivalent **NNF** formulas of the same size, up to a polynomial.
We always use notation $s, t, \ldots$ for states, $\alpha, \beta, \ldots$ for action descriptions, and $\varphi, \psi, \ldots$ for logical formulas.

Action theories We define the action language of (NNF) action theories, whose extensions we are going to study. To prepare the definition we associate an auxiliary variable

$p' \notin \mathbb{P}$ to each variable $p \in \mathbb{P}$; p' denotes the value of p *after* the action took place, while p denotes the value *before*.

Definition 4. An **NNFAT** *action description* is a Boolean formula α in NNF over $P_\alpha \cup \{p' \mid p \in P_\alpha\}$ for some set of state propositions $P_\alpha \subset \mathbb{P}$. The interpretation of an **NNFAT** action description α is defined for all $P \subseteq \mathbb{P}$ such that $P \supseteq \mathrm{V}(\alpha) \cap \mathbb{P}$ (that is, when all state propositions have a value) by

$$\forall s \subseteq P\colon\ I(\alpha, P)(s) = \{s' \mid (s, s') \models \alpha\}$$

where $(s, s') := s \cup \{p' \mid p \in s'\}$ is the assignment to $P \cup \{p' \mid p \in P_\alpha\}$ induced by s, s'. For $P \not\supseteq \mathrm{V}(\alpha) \cap \mathbb{P}$, $I(\alpha, P)$ is not defined. In words, an **NNFAT** expression represents the set of all ordered pairs $\langle s, s' \rangle$ such that s' is a successor of s, as a Boolean formula over variables in $\mathbb{P} \cup \{p' \mid p \in \mathbb{P}\}$.
Importantly, **NNFAT** does not assume persistency of values, so that if, for example, a variable does not appear at all in an **NNFAT** expression, then this means that its value after the execution of the action can be arbitrary.

Example 5. Let $P = \{p_1, p_2, p_3\}$, $s_1 = \emptyset$, $s_2 = \{p_1\}$, and $\alpha := p_1 \vee (p_1' \wedge p_2')$, which can be read "when p_1 is false before the action is taken, both p_1 and p_2 become true after the action, and when p_1 is true anything can occur". Then $\alpha(s_1) = \{\{p_1, p_2\}, \{p_1, p_2, p_3\}\}$, and $\alpha(s_2) = 2^P$.

Circuit representation Since we study the succinctness of languages, it is crucial to define the *size* of action descriptions. For this we use the same setting as typically used in knowledge compilation studies about propositional logic [7], and assume that all action descriptions α are represented by the directed acyclic graph, or *circuit*, obtained from the syntactic tree of α by iteratively identifying the roots of two isomorphic subexpressions to each other, until no more reduction is possible (like for binary decision diagrams [4]). Clearly, for all expressions α, the circuit associated to α in this manner is unique, and it can be computed in polynomial time from the plain expression or from a nonreduced circuit.

3 Frame operators

Our proposal is to enrich **NNFAT** with operators for expressing persistency of variable values. Operators in action languages are used to construct new action descriptions from existing ones, and they can be roughly divided into two types: the interpretation of a *syntactic* operator depends on its argument action descriptions, while that of a *semantic* operator depends on the actions but not on their description.
It is important to recall that **NNF** is a complete language for propositional logic and hence, that **NNFAT** is able to express any action. As a consequence, by *enriching* **NNFAT** we mean providing languages in which it is more convenient to express actions, as we will illustrate, but there is no action which those enriched languages can encode, that **NNFAT** cannot encode itself.

Semantic frame operator The semantic frame operator which we study builds on *circumscription* [14], which is a nonmonotonic semantics for formulas enforcing a form of closed-world assumption, and especially by *propositional* circumscription [8, 17]. However, by introducing an *operator* for this interpretation, we allow circumscription to be enforced only on some parts of an expression.
Let α be an action description and s be a P-state. For all partitions $\{X, V, F\}$ of P, we introduce the operator $\mathsf{C}_{X,V,F}$ so that $\mathsf{C}_{X,V,F}(\alpha)(s)$ chooses those α-successors of s which change variables from X minimally among all states with the same values over F.
Precisely, we define a state s' to be *preferred* to a state s'' with respect to a state s and to X, V, F, which we write $s' \prec^s_{X,V,F} s''$, if $s' \cap F = s'' \cap F$ and $(s''\Delta s) \cap X \subset (s'\Delta s) \cap X$ hold, where Δ denotes symmetric difference for sets.[1]

Definition 6. The action language $\mathbf{NNFAT_C}$ is the language $\langle L_\mathbf{C}, I_\mathbf{C} \rangle$, where the expressions with scope P in $L_\mathbf{C}$ are defined by the grammar

$$\alpha ::= p \mid p' \mid \neg p \mid \neg p' \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \mathsf{C}_{X,V,F}(\alpha),$$

where p ranges over P and $\langle X, V, F \rangle$ over partitions of P, and $I_\mathbf{C}$ is defined as for **NNFAT**, extended with

$$I_\mathbf{C}(\mathsf{C}_{X,V,F}(\alpha), P)(s) = \{s' \in I_\mathbf{C}(\alpha, P)(s) \mid \nexists s'' \in I_\mathbf{C}(\alpha, P)(s) : s'' \prec^s_{X,V,F} s'\}$$

Example 7. Let $P = \{p_1, \ldots, p_5\}$, $X = \{p_1, p_2\}$, $V = \{p_3\}$, and $F = \{p_4, p_5\}$. Let $\alpha = (p_1' \vee p_3') \wedge (p_2' \vee p_4') \wedge (p_5')$ and $s = \emptyset$. Then $\{p_1, p_2, p_5\}$ is an α-successor, but not a $\mathsf{C}_{X,V,F}(\alpha)$-successor, of s, because $s'' = \{p_2, p_3, p_5\}$ is also an α-successor of s with $s'' \cap F = \{p_5\} = s' \cap F$ and $(s''\Delta s) \cap X = \{p_2\} \subset \{p_1, p_2\} = (s'\Delta s) \cap X$. On the other hand, s'' is a $\mathsf{C}_{X,V,F}(\alpha)$-successor of s even though $s''' = \{p_3, p_4, p_5\}$ changes fewer values over X, since s'' and s''' differ over F and hence are incomparable with each other.

It can be seen that the $\mathsf{C}_{X,V,F}$ operator is convenient in particular for expressing actions which involve external causes (to be put in the set F) of changes of values for variables of interest (set X), in the presence of ramifications (set V).

Example 8. Consider encoding the action of driving from work to home. A particularly succinct description is

$$\mathsf{C}_{\{\mathsf{home}\},\{\mathsf{at\_work}\},\{\mathsf{flat\_tire},\mathsf{engine\_ok}\}} \Big((\mathsf{engine\_ok}' \wedge \neg \mathsf{flat\_tire}') \rightarrow \mathsf{home}') \wedge (\mathsf{home}' \leftrightarrow \neg \mathsf{at\_work}') \Big)$$

Consider $s = \{\mathsf{engine\_ok}, \mathsf{at\_work}\}$. Minimization of change over $\{\mathsf{home}\}$ entails that when the causes are not met (for instance, when $\mathsf{engine\_ok}'$ is false), among the possible successors of s only the ones with $\neg\mathsf{home}'$ are retained, ignoring the ramification $\mathsf{at\_work}'$; successors with

[1] As mnemonics, variables in V may vary; those in F are fixed.

home' are not retained, reflecting the fact that there is no "proof" provided by the action that home should change value. On the other hand, due to their presence in the set F, all combinations of causes will be retained. Precisely, the successors of s are $\{\mathsf{engine\_ok}, \mathsf{at\_work}, \mathsf{flat\_tire}\}$, $\{\mathsf{at\_work}\}$, $\{\mathsf{at\_work}, \mathsf{flat\_tire}\}$, and $\{\mathsf{engine\_ok}, \mathsf{home}\}$.

Syntactic frame operator We now define a *syntactic* operator, for representing the notion of persistency of languages like **PDDL**, where a variable does not change value if there is no *explicit* reason for this. Concretely, we want an operator F such that $\{p\}$ is an $\mathsf{F}(p' \vee \neg p')$-successor of $s = \emptyset$, but not an $\mathsf{F}(q' \vee \neg q')$-successor for $q \neq p$, although $p' \vee \neg p'$ describes the same action as $q' \vee \neg q'$. Therefore we need to formalize the intuition that some change is "explicitly mentioned" in an action description.
For this, we refine the notion of successor by considering the *effects* which apply in a state, themselves decomposed into *explicit* and *implicit* effects.

Definition 9. An *effect* (over $P \subseteq \mathbb{P}$) is a quadruple $\langle e^+, e^-, i^+, i^- \rangle$ such that e^+, e^-, i^+, i^- are pairwise disjoint subsets of P. The *explicit* (resp. *implicit*) part of such an effect is the pair $\langle e^+, e^- \rangle$ (resp. $\langle i^+, i^- \rangle$).

The positive ($e^+ \cup e^-$) and negative ($e^- \cup i^-$) are similar to add- and del-lists in STRIPS: $\langle e^+, e^-, i^+, i^- \rangle$ provokes a transition from a state s to $s' = (s' \cup e^+ \cup i^+) \setminus (e^- \cup i^-)$.
We now define effects for **NNFAT** action descriptions. For combinations of effects via $\wedge$, let $\varepsilon_1 = \langle e_1^+, e_1^-, i_1^+, i_1^- \rangle$ and $\varepsilon_2 = \langle e_2^+, e_2^-, i_2^+, i_2^- \rangle$. If $e_1^+ \cup i_1^+ = e_2^+ \cup i_2^+$ holds then we write $\varepsilon_1 \approx \varepsilon_2$ and set $\varepsilon_1 + \varepsilon_2 := \langle e_1^+ \cup e_2^+, e_1^- \cup e_2^-, i_1^+ \cap i_1^-, i_2^+ \cap i_2^- \rangle$. Intuitively, $\varepsilon_1 \approx \varepsilon_2$ means that they provoke exactly the same transitions, but possibly with different explicit/implicit changes, and $\varepsilon_1 + \varepsilon_2$ is the effect which makes explicit any change which is explicit in one of them.

Definition 10. Let α be an **NNFAT** action description and s be a state. Then the set of *effects* of α in s, written $\mathrm{E}(\alpha, s)$, is defined inductively by

$$\mathrm{E}(p, s) = \{\langle \emptyset, \emptyset, A, B \rangle \mid A, B \subseteq P\} \text{ for } s \models p,$$
$$\mathrm{E}(p, s) = \emptyset \text{ for } s \not\models p;$$
$$\text{and dually for } \mathrm{E}(\neg p, s)$$

$$\mathrm{E}(p', s) = \begin{array}{l} \{\langle \emptyset, \emptyset, A, B \rangle \mid A, B \subseteq P \setminus \{p\}\} \\ \cup \{\langle \{p\}, \emptyset, A, B \rangle \mid A, B \subseteq P \setminus \{p\}\} \end{array} \quad \text{for } s \models p,$$
$$\mathrm{E}(p', s) = \{\langle \{p\}, \emptyset, A, B \rangle \mid A, B \subseteq P \setminus \{p\}\} \text{ for } s \not\models p;$$
$$\text{and dually for } \mathrm{E}(\neg p', s)$$

$$\mathrm{E}(\alpha_1 \wedge \alpha_2, s)$$
$$= \{\varepsilon_1 + \varepsilon_2 \mid \varepsilon_1 \in \mathrm{E}(\alpha_1, s), \varepsilon_2 \in \mathrm{E}(\alpha_2, s), \varepsilon_1 \approx \varepsilon_2\};$$

$$\mathrm{E}(\alpha_1 \vee \alpha_2, s) = \mathrm{E}(\alpha_1, s) \cup \mathrm{E}(\alpha_2, s) \cup \mathrm{E}(\alpha_1 \wedge \alpha_2, s).$$

where A, B are always disjoint from each other.

Intuitively, the first case says that the action p is not applicable if s does not satisfy p (second line), and otherwise imposes no constraint on the successor state, but moreover does not set any value *explicitly*: a transition may occur from, say, $\{p, q\}$ to $\{r\}$, but then the positive effects $A = \{r\}$ and negative effects $B = \{p, q\}$ are considered to be *implicit*.
For atomic actions of the form p', we distinguish two cases. When s does not already satisfy p, then all effects of p' *explicitly* set p. Constrastingly, when s already satisfies p, then the action p' leaves the value unchanged, and we include both effects with an explicit setting of p (to the same value) and effects with no setting of p at all (neither implicit nor explicit). Of course, for fixed A, B, those two effects provoke a transition from s to the same successor s'. Nevertheless, including the effects which do not set p at all turns out to be necessary for $\wedge$ to behave as expected in the extension of **NNFAT** with the F operator (to be defined soon).
Finally, it is worth noting that we include the effects of $\alpha_1 \wedge \alpha_2$ in those of $\alpha_1 \vee \alpha_2$. Of course, the *transitions* of $\alpha_1 \wedge \alpha_2$ are already included in those of α_1 and in those of α_2, but not necessarily with the same explicit parts. For instance, for $\alpha_1 = p'$, $\alpha_2 = q'$, and $s = \emptyset$, $s' = \{p, q\}$ is both an α_1- and an α_2-successor of s, but the "fully" explicit effect $\langle \{p, q\}, \emptyset, \emptyset, \emptyset \rangle$ is only one of $\alpha_1 \wedge \alpha_2$.
With this in hand, we can define the *framing* operator F_X. Intuitively, $\mathsf{F}_X(\alpha)$ retains only those effects of α which include no implicit effect on variables of X. As can be seen, this is equivalent to removing variables of X from the implicit part of all effects. So we define $\mathrm{E}(\mathsf{F}_X(\alpha), s)$ to be

$$\{\langle e^+, e^-, i^+ \setminus X, i^- \setminus X \rangle \mid \langle e^+, e^-, i^+, i^- \rangle \in \mathrm{E}(\alpha, s)\}$$

Definition 11. The action language $\mathbf{NNFAT_F}$ is the language $\langle L_{\mathbf{F}}, I_{\mathbf{F}} \rangle$, where the expressions with scope P in $L_{\mathbf{F}}$ are defined by the grammar

$$\alpha ::= p \mid p' \mid \neg p \mid \neg p' \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \mathsf{F}_X(\alpha),$$

where p ranges over P and X over subsets of P, and $I_{\mathbf{F}}$ is defined for all α, s, $I_{\mathbf{F}}(\alpha, P)(s)$ is defined to be

$$\{(s \cup e^+ \cup i^+) \setminus (e^- \cup i^-) \mid \langle e^+, e^-, i^+, i^- \rangle \in \mathrm{E}(\alpha, s)\}$$

Example 12. Consider the action of leaving one's bike in a garage for them to repair exactly one wheel, but not knowing which one in advance.[2] The action may have two effects: making the front wheel or the back wheel ok (in both cases not affecting the other). However, in the process of repairing the back wheel, it might occur that the gear is changed. Additionally, in no case would the brakes be affected. Such an action could be encoded by

$$\mathsf{F}_{\mathsf{brakes}}\big(\mathsf{F}_{\mathsf{f\_wheel\_ok}}(\mathsf{b\_wheel\_ok}') \vee \mathsf{F}_{\mathsf{b\_wheel\_ok,gear}}(\mathsf{f\_wheel\_ok}')\big)$$

[2] *E.g.*, knowing only that they are not aligned which each other.

Importantly, in general, pushing all occurrences of F to the root of the expression changes its interpretation. For instance, in Example 12, this would yield the expression

$$\mathsf{F}_{\mathsf{brakes,f\_wheel\_ok,b\_wheel\_ok,gear}}\big(\mathsf{b\_wheel\_ok}' \vee \mathsf{f\_wheel\_ok}'\big)$$

according to which the gear can never change value.
Another important observation is that F_X is *not* a special case of $\mathsf{C}_{X,V,F}$. It might seem that F_X is nothing more than $\mathsf{C}_{X,\emptyset,P\setminus X}$. However, taking $P = \{p\}$, $\alpha = p' \vee \neg p'$, and $s = \emptyset$, it can be seen that both $\emptyset$ and $\{p\}$ are $\mathsf{F}_X(\alpha)$-successors of s, while only $\emptyset$ is a $\mathsf{C}_{X,\emptyset,P\setminus X}(\alpha)$-successor. Indeed, $\mathsf{F}_X(\alpha)$ takes into account the fact that both alternatives are explicitly mentioned in the formula, while $\mathsf{C}_{X,\emptyset,P\setminus X}(\alpha)$ considers only the semantics of α and hence, is equivalent to $\mathsf{C}_{X,\emptyset,P\setminus X}(\top)$.

Restriction on circuits In addition to $\mathbf{NNFAT_C}$ and $\mathbf{NNFAT_F}$, we will also study their natural restrictions where the operator $\mathsf{C}_{X,V,F}$ or F_X, respectively, occurs only at the root of expressions. Namely, $\mathbf{NNFAT_C}$ is $\mathbf{NNFAT}$ augmented only with expressions of the form $\mathsf{C}_{X,V,F}(\alpha)$, where α is an $\mathbf{NNFAT}$ action description. We denote by $\mathbf{NNFAT_{rC}}$ the resulting language. Similarly, we define the restricted language $\mathbf{NNFAT_{rF}}$. Observe that $\mathbf{PDDL}$, for instance, can be seen as a language without framing, together with an (implicit) operator F_P at the root of all expressions.

4 Compiling the Syntactic Operator Away

In this section, we show that the syntactic operator F_X can be eliminated from an $\mathbf{NNFAT_F}$ expression to yield an expression in $\mathbf{NNFAT}$ which has the same interpretation and size (up to a polynomial). Moreover, the elimination can be done in polynomial time. This means that $\mathbf{NNFAT_F}$ can be used as a convenient language for describing actions, still the algorithms which manipulate those descriptions (e.g. planners) need not be extended to cope with F, since all its occurrences can simply be eliminated in polynomial time and space.
Let us mention that the equivalent result is rather obvious for plain (tree-like) representations of formulas, but that we consider here *circuit representations*, in which a single subcircuit may occur in an exponential number of paths.
Our procedure essentially amounts to replacing all occurrences of F with subformulas similar to successor-state axioms. Before we show the result, we need two lemmas (the proofs are by induction on the structure of α).
The first lemma defines an expression $\mathrm{Expl}(\alpha, p)$, and shows that this expression states that if p changes its value via α, then there is at least one effect which does it explicitly. For states s, s' and action description α, write $\mathrm{E}(\alpha, s, s')$ for the set of effects which lead from s to s': $\mathrm{E}(\alpha, s, s') := \{\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s) \mid s' = (s \cup e^+ \cup i^+) \setminus (e^- \cup i^-)\}$.

Lemma 13. *Let α be a $\mathbf{NNFAT_F}$ action description with scope P, $s, s' \subseteq P$ be two states and $p \in s\Delta s'$. Then there is an effect $\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s, s')$ with $p \in e^+_e \cup e^-_e$ if and only if $(s, s') \models \mathrm{Expl}(\alpha, p)$, where the $\mathbf{NNFAT}$ expression $\mathrm{Expl}(\alpha, p)$ is defined to be*

- *α for $\alpha = p'$ or $\alpha = \neg p'$,*
- *$\bot$ for $\alpha = q'$ or $\alpha = \neg q'$ with $q \neq p$,*
- *$\bot$ for $\alpha = q$ or $\alpha = \neg q$, for all variables q,*
- *$(\mathrm{Expl}(\beta, p) \wedge \gamma) \vee (\beta \wedge \mathrm{Expl}(\gamma, p))$ for $\alpha = \beta \wedge \gamma$,*
- *$\mathrm{Expl}(\beta, p) \vee \mathrm{Expl}(\gamma, p)$ for $\alpha = \beta \vee \gamma$,*
- *$\bigwedge_{x \in X \cup \{p\}} \big((x \leftrightarrow x') \vee \mathrm{Expl}(\beta, x)\big)$ for $\alpha = \mathsf{F}_X(\beta)$.*

The second lemma states properties of sets of effects.

Lemma 14. *Let α be an $\mathbf{NNFAT_F}$ action description with scope $P \supseteq s$ and $\varepsilon_1, \varepsilon_2 \in \mathrm{E}(\alpha, s, s')$. Then it holds:*

1. *If $\varepsilon_1 \approx \varepsilon_2$ then $\varepsilon_1 + \varepsilon_2 \in \mathrm{E}(\alpha, s)$*
2. *There is at least one $\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s, s')$ such that $e^+ \cup i^+ = s' \setminus s$ and $e^- \cup i^- = s \setminus s'$*

Proposition 15. $\mathbf{NNFAT_F}$ *is translatable into* $\mathbf{NNFAT}$ *in polynomial time.*

Proof. Let α be a $\mathbf{NNFAT_F}$ action description with scope P. The translation $f(\alpha)$ is obtained by replacing each node $\mathsf{F}_X(\beta)$ (with $X \subseteq P$) of the circuit of α by $\beta \wedge \bigwedge_{p \in X} \big((p \leftrightarrow p') \vee \mathrm{Expl}(\beta, p)\big)$ and keeping the other nodes.
The circuit of $\mathrm{Expl}(\beta, p)$ can be computed in polynomial time (in each step we create a bounded amount of edges and nodes). Thus $f(\alpha)$ can be computed in polynomial time, too.
Now we show that $f(\alpha)$ describes the same action as α. Suppose that $f(\beta) \equiv \beta$ and $(s, s') \models f(\alpha) = \beta \wedge \bigwedge_{p \in X} \big((p \leftrightarrow p') \vee \mathrm{Expl}(\beta, p)\big)$. Then $s' \in \beta(s)$, and $(s, s') \models \mathrm{Expl}(\beta, p)$ for all $p \in X$. Then by lemmas 13 and 14 for every $p \in (s'\Delta s) \cap X$ there exists an effect $\langle e^+_{\beta,p}, e^-_{\beta,p}, i^+_{\beta,p}, i^-_{\beta,p}\rangle$ with $p \in e^+_{\beta,p} \cup e^-_{\beta,p}$ which mentions only variables from $s\Delta s'$ and by lemma 14 the sum $\langle e^+, e^-, i^+, i^-\rangle$ of these effects is as well in $\mathrm{E}(\beta, s, s')$ and $(s\Delta s') \cap X \subseteq e^+ \cup e^-$ and thus $\langle e^+, e^-, i^+, i^-\rangle \in \mathrm{E}(\alpha, s)$. This implies $s' \in (\mathsf{F}_X(\beta))(s)$. Conversely, if $s' \in (\mathsf{F}_X(\beta))(s)$ then there exists an effect $\langle e^+, e^-, i^+, i^-\rangle$ of β in s witnessing this with $(s\Delta s') \cap X \subseteq e^+ \cup e^-$. Therefore, by Lemma 13 all $\mathrm{Expl}(\beta, p)$ with $p \in (s\Delta s') \cap X$ from the definition of $f(\alpha)$ are satisfied by (s, s'), and for the rest the expression $p \leftrightarrow p'$ is satisfied. And β is satisfied by the inductive assumption. For $\alpha = \alpha_1 \wedge \alpha_2$ or $\alpha = \alpha_1 \vee \alpha_2$ we trivially have that $f(\alpha)$ describes the same action as α if the claim was proven for α_1 and α_2. □

5 Complexity of queries

We now turn to studying the complexity of *queries* to expressions. We concentrate on two natural queries for planning: checking the existence of a transition, and deciding applicability of an action in a state. These queries arguably are at the basis of most other reasonable queries.
Let $\langle L, I\rangle$ be a fixed action language, and let α denote an expression in L, $P \subseteq \mathbb{P}$ denote a set of variables such that $I(\alpha, P)$ is defined, and s, s' denote two P-states.

Definition 16. The decision problem SUCC takes as input α, P, s, s', and asks whether $s' \in \alpha(s)$.

Definition 17. The decision problem APPLIC takes as input α, P, s, and asks whether $\alpha(s) \neq \emptyset$.

SUCC for **NNFAT** amounts to model-checking of an NNF formula, and for $\mathbf{NNFAT_F}$ the complexity follows from Proposition 15.

Proposition 18. SUCC *is in* **P** *for* **NNFAT** *and* $\mathbf{NNFAT_F}$.

To prepare further results we introduce a notation.

Notation 19. Let $n \in \mathbb{N}$ and $X_n = \{x_1, \ldots, x_n\}$ be a set of variables. Observe that there are a cubic number N_n of clauses of length 3 over X_n. We fix an arbitrary enumeration $\gamma_1, \gamma_2, \ldots, \gamma_{N_n}$ of all these clauses, and we define $P_n \subset \mathbb{P}$ to be the set of state variables $\{p_1, p_2, \ldots, p_{N_n}\}$. Write $\ell \in \gamma_i$ if the literal ℓ occurs in the clause γ_i. Then to any 3-CNF formula φ we associate the P_n-state $s(\varphi) = \{p_i \mid i \in \{1, \ldots, N_n\}, \gamma_i \in \varphi\}$, and dually, to any P_n-state s, we associate the 3-CNF formula over X_n, written $\varphi(s)$, which contains exactly those clauses γ_i for which $p_i \in s$ holds.We set $\psi_n := \bigwedge_{i=1}^{N_n}(\neg p_i \vee \bigvee_{\ell\in\gamma_i} \ell)$. In words, ψ_n is satisfied by an assignment t to $P_n \cup \{x_1, \ldots, x_n\}$ if and only if the 3-CNF over $\{x_1, \ldots, x_n\}$ encoded by $t \cap P_n$, is satisfied by the assignment to $\{x_1, \ldots, x_n\}$ encoded by $t \cap \{x_1, \ldots, x_n\}$.

Example 20. Consider an enumeration of all clauses over $X_2 = \{x_1, x_2\}$ which starts with $\gamma_1 = (x_1 \vee x_1 \vee x_2), \gamma_2 = (x_1 \vee x_1 \vee \neg x_2), \gamma_3 = (x_1 \vee \neg x_1 \vee x_2), \ldots$ Then $\varphi = (x_1 \vee x_1 \vee x_2) \wedge (x_1 \vee \neg x_1 \vee x_2)$ is encoded by $s(\varphi) = \{p_1, p_3\}$.

For a formula ψ over the variables $\{q_i \mid j \in J\}$, write ψ' for the formula obtained by replacing all variables q_j by q'_j.

Proposition 21. SUCC *is* $\mathbf{coNP}$*-complete for* $\mathbf{NNFAT_{rC}}$.

Proof. We reduce the non-satisfiability of a 3-CNF to SUCC. Using Notation 19 for P_n and ψ_n, let $\alpha_n = C_{X_n,\emptyset,P_n}(\psi'_n \vee (\bigwedge_{i=1}^n x'_i))$. Take a 3-CNF φ over X_n which is not satisfied by the assignment "$\forall i : x_i = \top$". If φ is unsatisfiable, then for all $t \subseteq X_n$, $(s(\varphi) \cup t)$ falsifies ψ'_n, hence the only model m via $\psi'_n \vee (\bigwedge_{i=1}^n x_i)$ which respects $m \cap P_n = s(\varphi)$ is $s(\varphi) \cup X_n$. Conversely, if φ is satisfiable then let $t \subseteq X_n$ be a satisfying assignment. By assumption we have $t \neq X_n$ and hence, $t \subsetneq X_n$. On the other hand, by definition of ψ_n (Notation 19), $s(\varphi) \cup t$ satisfies α_n, and it follows that $s(\varphi) \cup t$ cannot be minimal over X_n. We have shown that φ is unsatisfiable if and only if $s(\varphi) \cup X_n$ is an α_n-successor of $\emptyset$. □

Proposition 22. SUCC *is* **PSPACE**-*complete for* $\mathbf{NNFAT_C}$.

Proof. We modify Notation 19 by introducing for every variable x_j two variables q_j, r_j and write $q_j \in \gamma_i$ if $x_j \in \gamma_i$, and $r_j \in \gamma_i$ if $\neg x_j \in \gamma_i$. We set $Q_n := \{q_j, r_j \mid 1 \leq j \leq n\}$. $S_n := Q_n \cup \{p_1, \ldots, p_{N_n}\}$. We obtain β^n_{n+1} by replacing all x_j in ψ_n by q'_j and all $\neg x_j$ by r'_j and then define recursively

$$\beta^n_i := \mathsf{C}_{\{q_i,r_i\},\emptyset,S_n\setminus\{q_i,r_i\}}\big((q_i \wedge r_i) \vee (q_i \wedge \beta^n_{i+1}) \vee (r_i \wedge \beta^n_{i+1})\big)$$

Let $\Phi := \forall x_1 : \exists x_2 : \ldots : \forall x_n : \varphi$, which is equivalent to $\forall x_1 : \neg(\forall x_2 : \neg(\ldots \neg(\forall x_n : \varphi)\ldots)$, be a quantified Boolean formula with a 3-CNF φ. Deciding the validity of such formulas is obviously **PSPACE**-complete. We claim that Φ is true if and only if $s(\varphi) \cup Q_n \in \beta^n_1(s(\varphi))$. Indeed, first observe that for all i and all states $s \subseteq S_n \setminus \{q_i, r_i\}$, $t \subseteq \{p_1, \ldots, p_{N_n}\}$: $s \cup \{q_i, r_i\} \in \beta^n_i(t) \Leftrightarrow s \cup \{q_i\}, s \cup \{r_i\} \notin \beta^n_{i+1}(t)$. We set $V_i := \{q_j, r_j \mid i \leq j \leq n\}$ and $W_i := \{q_i, r_i\}$ and it follows with $t := s(\varphi)$

$$\begin{aligned}
&s(\varphi) \cup Q_n \in \beta^n_1(s(\varphi)) \\
\Leftrightarrow &s(\varphi) \cup V_2 \cup \{q_1\}, s(\varphi) \cup V_2 \cup \{r_1\} \notin \beta^n_2(s(\varphi)) \\
\Leftrightarrow &\forall z_1 \in W_1 : \neg(s(\varphi) \cup V_2 \cup \{z_1\} \in \beta^n_2(s(\varphi))) \\
&\quad \ldots \\
\Leftrightarrow &\forall z_1 \in W_1 : \neg\big(\forall z_2 \in W_2 : \neg(\forall z_3 \in W_3 : \neg \ldots \\
&\quad (s(\varphi) \cup \{z_1, \ldots, z_n\} \in \beta^n_{n+1}(s(\varphi)))))\big)
\end{aligned}$$

$s(\varphi) \cup \{z_1, \ldots, z_n\} \in \beta^n_{n+1}(s(\varphi))$ in the last line is equivalent to φ being true under the assignment defined by $x_i := (z_i = q_i)$. We have proven the claim and thus **PSPACE**-hardness of SUCC. For membership: SUCC can be reduced to deciding the truth of a fully quantified boolean formula (because $s' \in \mathsf{C}_{X,V,F}(\alpha)(s) \Leftrightarrow \forall s'' : ((s'' \Delta s) \cap X \subsetneq (s' \Delta s) \cap X \wedge s'' \cap F = s' \cap F \Rightarrow s'' \notin \alpha(s))$). □

Proposition 23. APPLIC *is* $\mathbf{NP}$*-complete for* **NNFAT** *and* $\mathbf{NNFAT_{rC}}$ *and* **PSPACE**-*complete for* $\mathbf{NNFAT_C}$.

Proof. Satisfiability of a 3-CNF φ can be reduced to applicability in **NNFAT** by replacing each x by x' and checking whether the obtained action description describes an action which is applicable in $s = \emptyset$.
For $\mathbf{NNFAT_{rC}}$ we observe that $\mathsf{C}_{X,V,F}(\alpha)$ is applicable in s if and only if α is applicable in s.
For **PSPACE**-hardness in $\mathbf{NNFAT_C}$: $s' \in \alpha(s)$ if and only if $\alpha \wedge \bigwedge_{p\in s'} p' \wedge \bigwedge_{p\notin s'} \neg p'$ is applicable in s. For membership: to check for applicability of α in s we need to check for all s' whether $s' \in \alpha(s)$. □

6 Succinctness

Recall that all languages are fully expressive, so we use the following definition [7].

Definition 24. A language $\mathbf{L}_1$ is *at least as succinct as* $\mathbf{L}_2$ if there exists a polynomial-size translation from $\mathbf{L}_2$ into $\mathbf{L}_1$.

Our separation results rely on yet unproven assumptions on nonuniform complexity classes. Recall that $\mathbf{P/poly}$ (resp. $\mathbf{coNP/poly}$) is the class of all decision problems such that for all $n \in \mathbb{N}$, there is a polytime algorithm (resp. a nondeterministic polytime algorithm for the complement) which decides the problem for all inputs of size n [2]. The assumptions $\mathbf{coNP} \not\subseteq \mathbf{P/poly}$ and $\mathbf{PSPACE} \not\subseteq \mathbf{coNP/poly}$ which we use are standard ones; in particular, $\mathbf{coNP} \subseteq \mathbf{P/poly}$ would imply a collapse of the polynomial hierarchy at the second level (Karp-Lipton theorem), and $\mathbf{PSPACE} \subseteq \mathbf{coNP/poly}$ would imply a collapse at the third level [21].
We first observe that since $\mathbf{NNFAT}$ is translatable into $\mathbf{NNFAT_F}$ via the identity function, and $\mathbf{NNFAT_F}$ is a superlanguage of $\mathbf{NNFAT}$, so they are equally succinct.

Proposition 25. *If* $\mathbf{coNP} \not\subseteq \mathbf{P/poly}$ *then* $\mathbf{NNFAT_{rC}}$ *is strictly more succinct than* $\mathbf{NNFAT}$.

Proof. Recall from the proof of proposition 21 that $s(\varphi) \cup X_n$ is an α_n-successor of $s(\varphi)$ if and only if φ (assumed not to be satisfied by assigning $\top$ to all variables) is unsatisfiable, and that α_n depends only on the number n of variables in φ (not on φ itself). Now suppose that there exists a poly-size translation f from $\mathbf{NNFAT_{rC}}$ into $\mathbf{NNFAT}$. Then we can check whether φ is unsatisfiable by checking if it is not satisfied by the all-$\top$ assignment, and whether $s(\varphi) \cup X_n$ is an $f(\alpha_n)$-successor of $s(\varphi)$. This gives a nonuniform polytime algorithm (Proposition 18) for non-satisfiability, which is a $\mathbf{coNP}$-complete problem. □

The next proposition says that nesting of $\mathsf{C}_{X,V,F}$ operators contributes to succinctness. The proof is very similar to that of Proposition 25 .

Proposition 26. *If* $\mathbf{PSPACE} \not\subseteq \mathbf{coNP/poly}$ *then* $\mathbf{NNFAT_{rC}}$ *is strictly less succinct than* $\mathbf{NNFAT_C}$.

Proof. We use β_1^n from the proof of Proposition 22. Suppose there exists a poly-size translation f from $\mathbf{NNFAT_C}$ into $\mathbf{NNFAT_{rC}}$. Then like in the proof of Proposition 25, by checking whether $s(\varphi) \cup Q_n$ is an $f(\alpha_n)$-successor of $s(\varphi)$ we decide the validity of a QBF with a nonuniform $\mathbf{coNP}$ algorithm, implying $\mathbf{PSPACE} \subseteq \mathbf{coNP/poly}$. □

7 Conclusion

We studied extensions of $\mathbf{NNF}$ action theories with operators expressing two different types of persistency of variables, with the goal of enriching the language. We gave a picture of the resulting languages *à la* knowledge compilation map. It turns out that using a frame operator resembling that of PDDL at any level of nesting does not change time nor space complexity; hence this operator can be used when specifying actions, then compiled away efficiently so as to use algorithms designed for (standard) NNF action theories. The languages resulting for our second operator (related to the interpretation of formulas under circumscription) are more succinct but also have a greater complexity for basic queries.
Our results raise new open knowledge compilation-related questions. For example, we are interested in comparing these new languages to already well-known languages like variants of $\mathbf{PDDL}$ or $\mathbf{DL\text{-}PPA}$ [12] in terms of succinctness. We are also interested in the complexity of queries other than studied here, e.g. whether all successors of a state via a given sequential plan satisfy some property. Our long-term goal is to study action description languages as defined by allowed operators or constructs, so as to get a complete picture.

Acknowledgements

This work has been supported by the French National Research Agency (ANR) through project PING/ACK (ANR-18-CE40-0011).

References

[1] Alexandre Albore, Héctor Palacios, and Hector Geffner. Compiling uncertainty away in non-deterministic conformant planning. In *Proc. 19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215, pages 465–470, 2010.

[2] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[3] P. Balbiani, A. Herzig, and N. Troquard. Dynamic logic of propositional assignments: A well-behaved variant of pdl. In *2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 143–152, 2013.

[4] Randal E Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys (CSUR)*, 24(3):293–318, 1992.

[5] Daniel Bryce, Subbarao Kambhampati, and David E. Smith. Planning graph heuristics for belief space search. *Journal of Artificial Intelligence Research*, 26:35–99, 2006.

[6] Alessandro Cimatti and Marco Roveri. Conformant planning via symbolic model checking. *Journal of Artificial Intelligence Research*, 13:305–338, 2000.

[7] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[8] Thomas Eiter and Georg Gottlob. Propositional circumscription and extended closed-world reasoning are πp2-complete. *Theoretical Computer Science*, 114(2):231–245, 1993.

[9] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194 – 211, 1979.

[10] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[11] Tomas Geffner and Hector Geffner. Compact policies for fully observable non-deterministic planning as SAT. In *Proc. 28th International Conference on Automated Planning and Scheduling (ICAPS 2018)*, pages 88–96, 2018.

[12] Andreas Herzig, Frédéric Maris, and Julien Vianey. Dynamic logic of parallel propositional assignments and its applications to planning. In *Proc. 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5576–5582, 2019.

[13] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Gener. Comput.*, 4(1):67–95, 1986.

[14] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial intelligence*, 13(1-2):27–39, 1980.

[15] Drew McDermott. PDDL–the planning domain definition language. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998. Available at: `www.cs.yale.edu/homes/dvm` (consulted on 2020/03/16).

[16] Christian J. Muise, Sheila A. McIlraith, and Vaishak Belle. Non-deterministic planning with conditional effects. In *Proc. 24th International Conference on Automated Planning and Scheduling (ICAPS 2014)*, pages 370—374, 2014.

[17] Gustav Nordh. A trichotomy in the complexity of propositional circumscription. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 257–269. Springer, 2005.

[18] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial and Mathematical Theory of Computation*, 1991.

[19] Jussi Rintanen. Complexity of planning with partial observability. In *Proc. 14th International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 345–354, 2004.

[20] Son Thanh To, Tran Cao Son, and Enrico Pontelli. A generic approach to planning in the presence of incomplete information: Theory and implementation. *Artificial Intelligence*, 227:1–51, 2015.

[21] Chee K Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical computer science*, 26(3):287–300, 1983.

Résolution de Dec-POMDP à horizon infini à l'aide de contrôleurs à états finis dans JESP

Yang You[1] Vincent Thomas[1] Francis Colas[1] Olivier Buffet[1]

[1] Université de Lorraine, INRIA, CNRS, LORIA, F-54000 Nancy, France

prénom.nom@loria.fr

Résumé

Cet article s'intéresse à la résolution de problèmes de planification collaborative formalisés comme des POMDP décentralisés (Dec-POMDP) en cherchant des équilibres de Nash, c'est-à-dire des situations dans lesquelles la politique de chaque agent est une meilleure réponse aux politiques (fixes) des autres agents. Alors que l'algorithme joint equilibrium-based search for policies *(JESP) fait ceci dans le cadre d'horizons finis en se reposant sur des arbres-politiques, nous proposons ici d'adapter JESP aux Dec-POMDP à horizon infini en représentant les politiques des agents par des contrôleurs à états finis. Dans cet article, nous (1) expliquons comment transformer un Dec-POMDP avec $N-1$ contrôleurs à états finis fixés en un POMDP à horizon infini dont la solution est une meilleure réponse du $N^{ième}$ agent ; (2) proposons une variante de JESP, appelée inf-JESP, reposant sur cette transformation pour résoudre des Dec-POMDP à horizon infini ; (3) introduisons des initialisations heuristiques pour JESP visant à conduire à de bonnes solutions ; et (4) conduisons une évaluation empirique de notre approche sur des bancs d'essais de l'état de l'art.*

Mots Clef

Dec-POMDP, JESP, contrôleurs à états finis, équilibre de Nash

Abstract

This paper looks at solving collaborative planning problems formalized as Decentralized POMDPs (Dec-POMDPs) by searching for Nash equilibria, i.e.*, situations where each agent's policy is a best response to the other agents' (fixed) policies. While the* joint equilibrium-based search for policies *(JESP) algorithm does this in the finite-horizon setting relying on policy trees, we propose here to adapt JESP to infinite-horizon Dec-POMDPs by using Finite State Controller policy representations. In this article, we (1) explain how to turn a Dec-POMDP with $N-1$ fixed finite state controllers into an infinite-horizon POMDP whose solution is a best response of the N^{th} agent ; (2) propose a JESP variant based on this transformation, called inf-JESP, for solving infinite-horizon Dec-POMDPs ; (3) introduce heuristic initializations for JESP aiming at deterministically leading to good solutions ; and (4) conduct experiments on state-of-the-art benchmark problems to evaluate our approach.*

Keywords

Dec-POMDP, JESP, finite state controllers, Nash equilibrium

1 Introduction

Les problèmes de décisions markoviens partiellement observables décentralisés (Dec-POMDP) visent à représenter des problèmes de décision séquentielle multi-agents dans lesquels l'objectif est de concevoir simultanément les politiques de plusieurs agents de sorte que leur exécution décentralisée collecte le plus de récompenses cumulées possibles. Pour être exécutable, chaque politique d'agent doit ainsi reposer sur son historique individuel, à savoir la séquence des ses actions et observations passées.

Il a été démontré que résoudre un Dec-POMDP à horizon fini est de complexité NEXP au pire cas [5], même pour deux agents, ce qui limite l'efficacité possible des solveurs optimaux pour des Dec-POMDP génériques. Les principales difficultés de la résolution de Dec-POMDP sont liées à deux faits : (i) l'état du système évolue selon les actions de tous les agents ; et (ii) l'action effectuée par chaque agent ne peut dépendre que de son propre historique. Ainsi, toutes les politiques sont interdépendantes et chaque agent a besoin de considérer les historiques potentiellement rencontrés par les autres agents, ainsi que leurs politiques, pour prendre une décision de manière optimale.

Pour contourner ces interdépendances pendant le processus d'optimisation, l'algorithme JESP (*joint equilibrium-based search for policies*) [17] recherche des solutions sous la forme d'équilibres de Nash, la politique de chaque agent étant une *meilleure réponse* aux politiques des autres agents. Il accomplit cela pour des problèmes à horizon fini (en employant des représentations arborescentes des politiques) en optimisant individuellement la politique de chaque agent l'un après l'autre, les politiques des autres

agents étant figées, jusqu'à convergence, c'est-à-dire en répétant le processus jusqu'à ce qu'aucune amélioration ne soit possible.

Cet algorithme fait toutefois face à deux inconvénients. Premièrement, les équilibres de Nash sont des optima locaux et non globaux. Deuxièmement, il ne traite que des problèmes à horizon fini et fournit des arbres politiques, lesquels ne peuvent pas être employés avec des horizons infinis.

Dans cet article, nous abordons ce second inconvénient et proposons une façon de résoudre les problèmes à horizon infini à travers une approche JESP appelée *inf-JESP* pour "*infinite-horizon JESP*". Son point de départ est de reposer non sur des arbres politiques, mais sur des contrôleurs à états finis (FSC pour "*Finite State Controler*"), une représentation courante pour les POMDP à horizon infini, ce qui, à notre connaissance n'a pas été fait jusque là. Dans ce but, nous proposons une manière de concevoir le POMDP rencontré par un agent quand on fige les politiques FSC des autres agents et qu'on les combine avec le modèle Dec-POMDP. De là, nous utilisons un solveur de POMDP pour trouver un FSC qui soit une meilleure réponse individuelle, et intégrons cette étape dans un schéma algorithmique de type JESP.

Pour être plus précis, nous étendons et améliorons la méthode JESP sur trois aspects : (1) les représentations arborescentes sont remplacées par des FSC pour résoudre les problèmes à horizon infini et pour concevoir chaque POMDP intermédiaire sans avoir à considérer des distributions sur les historiques possibles des autres agents (mais seulement sur les nœuds internes de leurs FSC) ; (2) des solveurs de POMDP de l'état de l'art sont utilisés à chaque étape (dans ce travail, nous utilisons SARSOP [14]) pour approximer la fonction de valeur optimale d'un POMDP et, sur la base du travail de GRZEŚ et al. [10], un FSC est dérivé de l'approximation résultante ; (3) une nouvelle méthode d'initialisation heuristique pour JESP est proposée, dans laquelle des FSC individuels sont extraits d'une politique jointe obtenue en résolvant un POMDP multiagent plus simple (MPOMDP) [23] dans lequel tous les agents sont contrôlés par une entité unique qui a accès à toutes les observations reçues. Ces FSC individuels extraits peuvent être utilisés pour initialiser notre algorithme et fournir des solutions de façon déterministe.

En suivant ces directions, nous espérons que l'emploi de FSC aidera à concevoir des politiques à la fois dotées de représentations compactes, mais aussi plus faciles à exécuter et à comprendre (comme présenté dans [10]) que le JESP classique

Dans la section 2, nous discutons des travaux connexes sur les contrôleurs à états finis et les méthodes de résolution de Dec-POMDP existantes. La section 3 définit formellement les Dec-POMDP et les FSC. Dans la section 4, nous (1) expliquons comment combiner des FSC avec un Dec-POMDP pour générer le POMDP requis à chaque itération de inf-JESP ; (2) puis décrivons l'algorithme inf-JESP global dédié à la résolution de Dec-POMDP à horizon infini ; et (3) présentons une méthode d'initialisation heuristique pour les algorithmes de la famille de JESP. Enfin, des résultats expérimentaux et leur analyse sont présentés en section 5 avant de conclure en section 6.

2 Travaux connexes

Résolution de Dec-POMDP Un premier type d'approche pour résoudre des Dec-POMDP consiste à transformer le Dec-POMDP en un problème de plus court chemin déterministe, comme le fait Multi-Agent A* (MAA*) [25], ou même un processus de décision markovien déterministe, en utilisant une statistique suffisante contenant des informations pertinentes concernant l'état du processus. Ainsi, plusieurs approches ont été proposées pour résoudre des Dec-POMDP en utilisant les *états d'occupation* définis comme la distribution de probabilité sur l'état réel du système et les possibles historiques d'actions et observations passées des différents agents. FB-HSVI [9] emploie le schéma algorithmique HSVI [24], c'est-à-dire une recherche heuristique dans l'espace des états d'occupation, permettant ainsi de dériver une politique jointe ϵ-optimale en temps fini. Reposant sur la même idée, MACDERMED et ISBELL [15] ont proposé un algorithme appelé PBVI-BB pour transformer un Dec-POMDP en un POMDP avec un nombre fini d'états de croyance qui peut être résolu par des méthodes à base de points. Le principal intérêt de ces approches est qu'elles peuvent donner des solutions aussi proches de l'optimum que désiré, mais la taille de l'espace d'état pour le problème correspondant explose quand le nombre d'actions et d'observations croît, nécessitant d'importantes ressources computationnelles pour obtenir une solution proche de l'optimum.

Un second type d'approche consiste à explorer l'espace des politiques jointes en optimisant simultanément les politiques paramétrées de tous les agents. Pour des Dec-POMDP à horizon infini, ces approches représentent les politiques (à mémoire bornée) de chaque agent de manière compacte comme un FSC, rendant possible de rechercher directement dans l'espace des FSC à nombre de nœuds fini. Profitant de cette représentation, AMATO, BERNSTEIN et ZILBERSTEIN [2] proposent d'optimiser directement les paramètres des FSC en représentant le problème Dec-POMDP comme un problème de programmation non-linéaire (NLP), employant des outils de NLP pour calculer la politique jointe optimale. Cette approche a été améliorée en employant des FSC sous la forme de machines de Mealy à la place de machines de Moore, dérivant ainsi le solveur MealyNLP [3]. D'autres approches [13, 20, 21] utilisent des techniques *Expectation-Maximization* en reformulant le problème d'optimisation Dec-POMDP en un problème d'inférence consistant à estimer les meilleurs paramètres des FSC afin de maximiser la probabilité de générer des récompenses, ce qui conduit à des politiques jointes sous-optimales (du fait de limitations d'Expectation-Maximization). En particulier, PeriEM

[21] travaille avec des FSC périodiques (comme Peri, un algorithme d'amélioration de FSC). Comme les descentes de gradient pour POMDP (i) s'étendent naturellement aux Dec-POMDP [26], et (ii) peuvent servir à optimiser les paramètres d'un FSC [16, 1], ils permettraient aussi d'optimiser plusieurs FSC dans un cadre multi-agent.

Un troisième type d'approche se concentre sur des heuristiques pour concevoir une politique jointe. C'est le cas de JESP (joint equilibrium-based search for policies), proposé par NAIR et al. [17]. Celui-ci repose sur l'observation qu'une politique jointe optimale est à l'évidence un équilibre de Nash, puisqu'aucun agent ne peut, seul, améliorer la valeur de cette politique jointe. Même si le contraire n'est pas vrai, JESP propose de chercher des équilibres de Nash en optimisant la politique d'un agent à la fois, considérant que les politiques des autres agents sont figées, et ce, jusqu'à ce qu'aucune amélioration ne soit possible. Cette optimisation est réalisée en résolvant à chaque itération un problème de décision mono-agent modélisé comme un processus de décision markovien partiellement observable (POMDP) combinant la dynamique du Dec-POMDP et les politiques, connues, des autres agents. Dans ce problème, l'agent à optimiser n'a accès qu'à ses propres observations et doit raisonner, à travers des distributions de probabilité, sur l'état caché du système et les historiques d'action et d'observation des autres agents. En transformant un problème de décision séquentiel multi-agent en une séquence de plusieurs problèmes de décision mono-agents à résoudre, JESP réduit fortement la complexité de la résolution, mais se restreint à trouver des optima locaux. De plus, dans l'algorihtme JESP original, la résolution de POMDP est accomplie soit par une recherche exhaustive, soit par une approche par programmation dynamique, les politiques étant représentées par des arbres politiques. Dans ce cas, les historiques des autres agents devraient être représentées comme une variable cachée du POMDP mono-agent, limitant JESP à des problèmes à horizon fini tout en requérant un espace d'états dont la taille croît exponentiellement avec l'horizon considéré. Nous proposons d'adapter JESP à des problèmes à horizon infini en utilisant une représentation FSC pour la politique de chaque agent.

Parmi les approches de résolution de Dec-POMDP, Dec-BPI (*Decentralized Bounded Policy Iteration*) [6, 7] est étroitement lié à notre proposition. Il repose sur une représentation par FSC stochastiques complétée par un mécanisme de corrélation permettant aux agents d'avoir accès aux mêmes nombres pseudo-aléatoires, et propose une approche similaire à JESP en se concentrant sur l'amélioration des paramètres d'un nœud de FSC (ou du mécanisme de corrélation) à la fois en utilisant un programme linéaire. À l'inverse, nous proposons de ne pas modifier qu'un FSC localement mais, à chaque itération d'inf-JESP, de dériver un FSC complètement nouveau solution induit par les FSC des autres agents. Notons que, comme JESP, Dec-BPI ne peut fournir qu'un équilibre de Nash sans aucune garantie de trouver une solution globalement optimale. Un des principaux avantages de Dec-BPI est, comme la version théorique de notre algorithme, de se concentrer sur des FSC de taille fixe, alors que la version que nous implémentons relâche cette contrainte, au risque d'obtenir de grands FSC (voir sections 3 et 5).

Représentation et évaluation des politiques FSC Puisque JESP repose sur la résolution de POMDP, cette section discute de contributions employées dans ce travail, en se concentrant sur les représentations FSC pour les politiques POMDP à horizon infini. De tels FSC sont compacts, et ont ouverts de nouvelles directions de recherche pour la résolution de POMDPs en optimisant directement leurs paramètres.

Une des contributions principales dans ce contexte est itération sur les politiques pour POMDP [11, 12]. Dans ces articles, HANSEN propose un algorithme d'itération sur les politiques reposant sur des politiques FSC (ainsi qu'une recherche heuristique). Suivant le schéma algorithmique d'itération sur les politiques, itération sur les politiques pour POMDPs repose sur deux étapes : (i) une étape d'évaluation consistant à estimer la valeur de la politique FSC courante, et (ii) une étape d'amélioration mettant à jour la politique FSC à l'aide de cette évaluation à travers l'addition de nouveaux nœuds et l'élagage ou le remplacement de nœuds dominés. On notera que, si l'étape d'amélioration de la politique peut être très coûteuse en temps de calcul, l'étape d'évaluation de politique peut être effectuée très efficacement en résolvant un système d'équations linéaires [11]. Dans le présent article, nous réutilisons la même technique d'évaluation des FSC obtenus à chaque itération de JESP.

Notons que des descentes de gradients peuvent aussi être employées pour optimiser les paramètres d'un FSC (stochastique) de taille donnée, comme le font MEULEAU et al., ABERDEEN [16, 1].

Plus récemment, GRZEŚ et al. [10] ont expliqué comment dériver un FSC à partir d'une fonction de valeur et fournit une méthode de compression de contrôleur par suppression des nœuds redondants ou dominés. Cela permet d'utiliser des solveurs de POMDP de l'état de l'art pour calculer une fonction de valeur tout en exprimant néanmoins la politique finale comme un FSC pour une exécution de politique moins gourmande en énergie. Dans le présent article, nous utilisons des variantes des mêmes techniques (i) pour calculer, à chaque itération d'inf-JESP, un FSC à partir de l'ensemble d'α-vecteurs solution d'un POMDP (voir section 4.3), ainsi que (ii) pour munir les agents du Dec-POMDP de politiques initiales (voir section 4.4).

3 État de l'art

3.1 Dec-POMDP

La recherche de comportements collaboratifs optimaux pour un groupe d'agents sous dynamique stochastique et observabilité partielle est typiquement formalisé comme un *processus de décision markovien partiellement observable*

décentralisé (Dec-POMDP).

Definition 1. *Un **Dec-POMDP** avec $|\mathcal{I}|$ agents est représenté par un tuple $M \equiv \langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, T, O, R, b_0, H, \gamma \rangle$, où :*

- *$\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$ est un ensemble fini d'**agents** ;*
- *$\mathcal{S}$ est un ensemble fini d'**états** ;*
- *$\mathcal{A}^i$ est l'ensemble fini d'**actions** de l'agent i ; $\mathcal{A} = \bigtimes_i \mathcal{A}^i$ est l'ensemble fini des actions jointes ;*
- *Ω^i est l'ensemble fini des **observations** de l'agent i ; $\Omega = \bigtimes_i \Omega^i$ est l'ensemble fini des observations jointes ;*
- *$T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ dénote la **fonction de transition** ; $T(s, a, s')$ est la probabilité de transiter vers le prochain état s' depuis l'état s si l'action jointe a est effectuée ;*
- *$O : \mathcal{A} \times \mathcal{S} \times \Omega \to \mathbb{R}$ est la **fonction d'observation** ; $O(a, s', o)$ est la probabilité d'observer o si l'action jointe a est effectuée et l'état résultant est s' ;*
- *$R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ est la **fonction de récompense** ; $R(s, a)$ donne la récompense immédiate reçue pour l'exécution de l'action jointe a dans l'état s ;*
- *b_0 est la **distribution de probabilité initiale** (ou croyance initiale) sur les états ;*
- *$H \in \mathbb{N} \cup \{\infty\}$ est l'**horizon temporel** (potentiellement infini) ;*
- *$\gamma \in (0, 1)$ est le **facteur d'atténuation**, lequel définit l'atténuation appliquée aux récompenses futures.*

Chaque agent i peut être muni d'une politique d'action π^i qui associe à chaque historique d'action-observation possible une action à effectuer. L'objectif est alors de trouver une politique jointe $\pi = \langle \pi^1, \ldots, \pi^{|\mathcal{S}|} \rangle$ qui maximise le critère de performance, ici l'espérance du retour atténué à partir de b_0 :

$$V_H^\pi(b_0) \stackrel{\text{def}}{=} \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^{-t} r(S_t, A_t) \mid S_0 \sim b_0, \pi\right].$$

3.2 POMDP

Dans ce travail, nous nous intéresserons à des solutions de Dec-POMDP sous la forme d'*équilibres*, c'est-à-dire de situations dans lesquelles chaque agent i suit une politique qui est une *meilleure réponse* étant données les politiques fixes des autres agents, notées $\pi^{\neq i}$, ce qui induit un Dec-POMDP mono-agent, c'est-à-dire un POMDP. Dans le cas d'un POMDP, une politique optimale existe dont l'entrée est la croyance b, c'est-à-dire la distribution de probabilité sur les états étant donné l'historique d'action-observation courant. Pour un horizon h fini, la valeur de la politique π à l'état de croyance b est définie récursivement comme suit :

$$V_h^\pi(b) = r(b, \pi(b)) + \gamma \sum_o Pr(o \mid b, \pi(b)) V_{h-1}^\pi(b^{a,o}),$$

où (i) $r(b, a) = \sum_s b(s) \cdot r(s, a)$, (ii) $Pr(o \mid b, \pi(b))$ dépend de la dynamique, et (iii) $b^{a,o}$ est l'état de croyance mis-à-jour après exécution de a et perception de o.

Estimer V^* permet de dériver une politique quasi-optimale, et repose souvent sur l'équation d'optimalité de Bellman :

$$V_h^*(b) = \max_a \left[r(b, a) + \gamma \sum_o Pr(o \mid b, a) V_{h-1}^*(b^{a,o}) \right].$$

Pour un horizon fini h, V_h^* est linéaire par morceaux est convexe (PWLC) en b. Quand l'horizon est infini, $V^*(= V_\infty^*)$ peut ainsi être approché par une enveloppe supérieure d'hyperplans—appelés α-vecteurs $\alpha \in \Gamma$.

3.3 Contrôleurs à états finis

Dans les POMDP comme dans les Dec-POMDP, les politiques solutions peuvent aussi être cherchées sous la forme de *contrôleurs à états finis* (FSC) (aussi appelés *graphes politiques* [16]), c'est-à-dire des automates dont les transitions d'un nœud interne au suivant dépendent de l'observation reçue et génèrent les actions à accomplir.

Definition 2. *Un **FSC** est représenté par un tuple $fsc \equiv \langle N, \eta, \psi \rangle$, où :*

- *N est un ensemble fini de nœuds, avec n_0 le nœud initial ;*
- *$\eta : N \times \Omega \times N \to \mathbb{R}$ est la fonction de transition entre nœuds du FSC ; $\eta(n, o, n') \stackrel{\text{def}}{=} Pr(n' | n, o')$ est la probabilité de transiter vers le nœud $n' \in N$ depuis le nœud $n \in N$ si l'observation $o' \in \Omega_i$ est perçue ;la notation $n' = \eta(n, o)$ est aussi employée quand cette transition est déterministe ;*
- *$\psi : N \times \mathcal{A} \to \mathbb{R}$ est la fonction de sélection d'action du FSC ; $\psi(n, a) \stackrel{\text{def}}{=} Pr(a|n)$ est la probabilité de choisir l'action $a \in \mathcal{A}$ dans le nœud n ; la notation $a = \psi(n)$ est aussi employée quand cette fonction est déterministe.*

HANSEN [11] a établi que, dans un POMDP donné, calculer la fonction de valeur d'un FSC déterministe (η et ψ étant tous deux déterministes) nécessite de résoudre le système d'équations linéaires suivant, avec un α-vector par nœud interne :

$$\alpha_s^n = R(s, a_n) + \gamma \sum_{s', o} T(s, a_n, s') O(a_n, s', o) \alpha_{s'}^{\eta(n,o)}, \tag{1}$$

où n est l'index d'un nœud de fsc, et $a_n \stackrel{\text{def}}{=} \psi(n)$. Ce système d'équations linéaires peut être résolu par un processus itératif en tirant profit du théorème du point fixe. Ce processus est typiquement interrompu quand le résidu de Bellman (la plus grande variation de valeur) est en-dessous d'un seuil ϵ donné, de sorte que la valeur calculée est à $\frac{\epsilon}{1-\gamma}$ de la vraie valeur de la politique FSC.

4 Infinite-horizon JESP

inf-JESP repose sur une procédure de recherche locale principale, laquelle est typiquement relancée plusieurs fois avec des initialisations aléatoires différentes pour converger vers différents optima locaux. Cette recherche locale, présentée en section 4.1, de manière itérative (i) définit un POMDP meilleur-réponse pour chaque agent sur la base des politiques des autres agents (voir section 4.2), et (ii) le résout pour extraire et évaluer le FSC associé (voir section 4.3). Nous expérimentons aussi avec une initialisation non aléatoire de inf-JESP, laquelle est décrite en section 4.4.

4.1 Algorithme principal

La politique de chaque agent est représentée par un FSC déterministe. Pour contrôler le coût computationnel de chaque itération, nous bornons la taille des FSC solution avec un paramètre $K \in \mathbb{N}^*$ (de sorte que le nombre de FSC considérés est fini). La recherche locale commence ainsi avec $|\mathcal{I}|$ FSC de taille au plus K générés aléatoirement dans fsc. Ensuite, elle boucle sur les agents, chaque itération tentant d'améliorer la politique d'un agent i en trouvant (line 7) un FSC fsc'_i de taille K qui soit une meilleure réponse aux FSC $fsc_{\neq i}$ courants (figés) des $|\mathcal{I}| - 1$ autres agents (dénotés $\neq i$). La section 4.2 détaille comment le problème auquel est confronté l'agent i est formalisé comme un POMDP et résolu. La ligne 8 repose sur l'équation 1 pour évaluer la solution $\langle fsc'_i, fsc_{\neq i}\rangle$ (en b_0), à moins que le solveur de POMDP en ligne 7 ne fournisse cette information. Ensuite, si une meilleure solution a été trouvée, fsc'_i remplace fsc_i dans fsc. Le processus s'arrête quand le nombre d'itérations consécutives sans améliorations, $\#ni$, atteint $|\mathcal{I}|$.

Algorithme 1 : ∞-horizon JESP

```
1  [Input :] K : FSC size | fsc : initial FSCs
2  Fct LocalSearch (K, fsc ≝ ⟨fsc_1, ..., fsc_|I|⟩)
3      v_bestL ← eval(fsc)
4      #ni ← 0          // #(iterations w/o improvement)
5      i ← 0                        // Id of current agent
6      repeat
7          fsc'_i ← Solve2FSC (fsc_≠i, K)
8          v ← eval(fsc'_i, fsc_≠i)
9          if v ≥ v_bestL then
10             fsc_i ← fsc'_i
11             v_bestL ← v
12             #ni ← 0
13         else
14             #ni ← #ni + 1
15         i ← (i + 1)  mod |I|
16     until #ni = |I|
17     return ⟨fsc, v_bestL⟩
```

Propriétés Pour l'agent i, à chacun des K nœuds de son FSC est associée une action de $\mathcal{A}_i$, et à chacun de ses $K \times |\Omega|$ arcs est associé un nœud de FSC, de sorte que le nombre de FSC *déterministes* possibles $|\mathcal{FSC}_i|$ est majoré par $|\mathcal{A}_i|^K \cdot K^{K\times|\Omega|}$.[1] Nous pouvons ainsi supposer ici que chaque POMDP rencontré est résolu optimalement, ce qui conduit aux propriétés suivantes.

Proposition 1. *La recherche locale de inf-JESP converge en un nombre fini d'itérations à un équilibre de Nash.*

Démonstration. La recherche n'accepte que des solutions de qualité croissante, de sorte que le nombre d'itérations (sur tous les agents) est majoré par le nombre (fini) de solutions possibles : $|\mathcal{FSC}| = \prod_i |\mathcal{FSC}_i|$.
La recherche s'interrompt quand le FSC de chaque agent est une meilleure réponse aux FSC des autres agents, c'est-à-dire quand un équilibre de Nash est atteint. □

Notons que ces équilibres ne sont que des optima locaux. Permettre une infinité de redémarrages aléatoires garantit de converger vers un optimum local avec probabilité 1. Évidemment, l'ensemble des équilibres de Nash dépend de l'ensemble des politiques considérées, donc du paramètre K dans notre cadre. Augmenter K donne accès à plus de politiques (sans en enlever), et ainsi à des équilibres de Nash potentiellement meilleurs.
En pratique (voir section4.2), nous utiliserons un solveur de POMDP sous-optimal dans **Solve2FSC** (ligne 7) dans lequel les tailles des FSC ne sont pas contraintes par une borne K. Si ce solveur de POMDP retourne une solution fsc'_i moins bonne que fsc_i, elle sera ignorée, de sorte que les améliorations monotones sont préservées, et la recherche s'arrêtera toujours nécessairement en temps fini. Nous ne perdrons que la propriété que les solutions obtenues sont des équilibres de Nash. Elles peuvent être proches d'équilibres de Nash si le solveur de POMDP retourne des solutions ϵ-optimales. On parlera simplement d'*équilibre*.

4.2 POMDP meilleure réponse pour l'agent i

Solution à horizon fini (JESP) Dans JESP, quand on raisonne sur la politique de l'agent i en considérant les politiques $\pi_{\neq i}$ des autres agents connues et figées, l'agent i doit maintenir une croyance b^t_{JESP} sur le tuple $e^t = \langle s^t, \vec{\omega}^t_{\neq i}\rangle$, où s^t dénote l'état courant et $\vec{\omega}^t_{\neq i}$ les historiques d'action-observation des autres agents. Cette croyance sur e^t est une statistique suffisante parce que, en l'utilisant, l'agent i peut déduire la distribution de probabilité sur l'état courant du système et inférer les actions des autres agents $a^t_{\neq i} = \pi_{\neq i}(\vec{\omega}^t_{\neq i})$.
Toutefois, dans les problèmes à horizon infini, l'agent i ne peut considérer les historiques d'action-observation complets $\vec{\omega}^t_{\neq i}$ des autres agents parce que leur nombre croît exponentiellement avec le temps, de sorte que l'espace d'états

1. Le nombre exact est plus petit du fait de symétries et parce que, dans certains FSC, tous les nœuds internes ne sont pas atteignables.

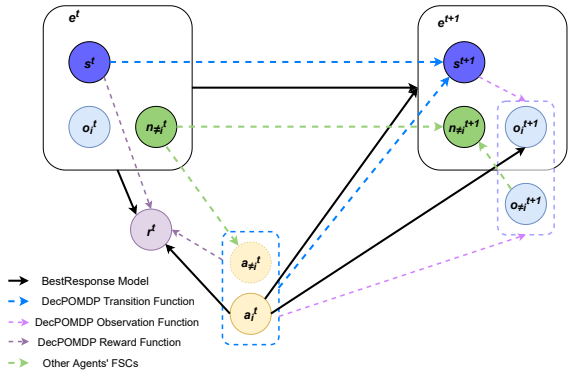


FIGURE 1 – Structure du POMDP meilleure-réponse : Les flèches noires décrivent un POMDP standard ; Les flèches bleues montrent la fonction de transition T du Dec-POMDP ; Les flèches roses montrent la fonction d'observation O du Dec-POMDP ; Les flèches violettes montrent la fonction de récompense R ; Les flèches vertes montrent l'évolution des nœuds internes des agents $\neq i$.

est infini. Avec des politiques représentées par des FSC, l'agent i peut raisonner sur les nœuds internes des autres agents au lieu des historiques d'action-observation. Mais combiner des FSC avec un Dec-POMDP n'est pas trivial (voir annexe A, page 12).

Solution à horizon infini Plusieurs POMDP meilleure-réponse peuvent être conçus. Les ensembles d'actions et d'observations sont imposés ($\mathcal{A}_i$ et Ω_i), mais plusieurs choix sont possibles pour l'ensemble des états *étendus*. Ici, l'état étendu $e^t \in \mathcal{E}$ contient (i) s^t, l'état courant du Dec-POMDP, (ii) $n^t_{\neq i} \equiv \langle n^t_1, \ldots, n^t_{i-1}, n^t_{i+1}, \ldots, n^t_n \rangle$, les nœuds des FSC des $|\mathcal{I}| - 1$ autres (agents $\neq i$) au pas de temps courant, et (iii) o^t_i, l'observation courante de l'agent i. Nous avons ainsi $\mathcal{E} \stackrel{\text{def}}{=} \mathcal{S} \times N_{\neq i} \times \Omega_i$. Étant donnée une action a^t_i, l'état étendu $e^t \equiv \langle s^t, n^t_{\neq i}, o^t_i \rangle$ évolue selon les étapes suivantes (voir figure 1) :

1. d'abord, chaque agent $j \neq i$ sélectionne son action a_j selon son nœud de FSC courant ;
2. ensuite l'état s^t transitionne vers s^{t+1} selon la fonction de transition du Dec-POMDP, l'action a^i de l'agent i et les actions des agents $\neq i$ ($a^t_{\neq i}$), c'est-à-dire l'action jointe $a^t \equiv \langle a^t_i, a^t_{\neq i} \rangle$; et
3. les nœuds FSC des agents j (y compris i) évoluent conjointement selon leurs observations (pas nécessairement indépendantes) o^{t+1}_j, lesquelles peuvent être inférées de s^{t+1} et de l'action jointe a^t.

Ce choix de conception pour l'état étendu induit un POMDP parce que les propriétés suivantes sont satisfaites : (i) il induit un processus markovien (contrôlé par l'action) ; (ii) l'observation dépend de l'action et du prochain état étendu ; et (iii) la récompense dépend de l'état et de l'action. En effet, dériver les fonctions de transition, d'observation et de récompense pour ce POMDP meilleure-réponse (voir annexe A) conduit à :

$$T_e(e^t, a^t_i, e^{t+1}) = Pr(e^{t+1}|e^t, a^t_i)$$
$$= \sum_{o^{t+1}_{\neq i}} T(s^t, \langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}) \cdot \eta_{\neq i}(n^t_{\neq i}, o^{t+1}_{\neq i}, n^{t+1}_{\neq i})$$
$$\cdot O(s^{t+1}, \langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, \langle o^{t+1}_{\neq i}, o^{t+1}_i \rangle),$$
$$O_e(a^t_i, e^{t+1}_i, o^{t+1}_i) = Pr(o^{t+1}_i|a^t_i, e^{t+1}_i)$$
$$= Pr(o^{t+1}_i|a^t_i, \langle s^{t+1}, n^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle) = \mathbf{1}_{o^{t+1}_i = \tilde{o}^{t+1}_i},$$
$$r_e(e^t, a^t_i) = r(s^t, a^t_i, \psi_{\neq i}(n^t_{\neq i})),$$

où $\eta_{\neq i}(n^t_{\neq i}, o^{t+1}_{\neq i}, n^{t+1}_{\neq i}) = \prod_{j \neq i} \eta(n^t_j, \tilde{o}^{t+1}_j, n^{t+1}_j)$ et $\psi_{\neq i}(n^t_{\neq i}) = a^t_{\neq i}$.
Ce POMDP meilleure-réponse peut être vu comme un MDP à observabilité mixte (MOMDP [19, 4]), puisqu'une des variables d'état est complètement observable. Aussi, $\mathcal{E}$ peut être rendu plus petit en enlevant les états étendus impossibles. En particulier, comme on peut l'observer dans la fonction de transition T_e, une observation o^t_i peut être impossible sous un certain état s^t (du fait de la fonction d'observation).[2] L'efficacité de ce processus d'éliminination d'états (étendus) dépend évidemment beaucoup du Dec-POMDP considéré, le pire cas étant quand toutes les observations ont toujours une probabilité d'occurrence non nulle.

4.3 Obtenir et évaluer fsc_i

Après avoir construit le POMDP (meilleure-réponse) pour l'agent i, un solveur de POMDP est requis pour obtenir la politique de l'agent i. Il y a différents types de solveurs. Un premier type donne directement un FSC qui peut être utilisé pour construire le prochain POMDP meilleure-réponse [11]. Un second type approche la fonction de valeur optimale, comme les algorithmes à base de points tels que PBVI [22], HSVI [24], et SARSOP [14]. Pour tirer profit de ces solveurs modernes, nous choisissons cette seconde méthode pour résoudre le POMDP meilleure-réponse et obtenir un ensemble Γ d'α-vecteurs.

Algorithme Nous utilisons l'algorithme 2, qui est similaire à ce qui a été proposé par GRZEŚ et al. [10], pour transformer une fonction de valeur approchée (sous la forme d'un ensemble d'α-vecteurs Γ) en un FSC. Il créé d'abord un nœud initial n_0 à partir de l'état de croyance initial b_0 et de Γ (ligne 2), et l'ajoute à la fois au nouveau FSC (N) et à une file (G). Ensuite, l'algorithme traite chaque nœud $n = \langle \alpha, b, a \rangle$ de G (premier entré d'abord), avec chaque observation possible o_i comme suit (les observations de probabilité nulle induisant des boucles (ligne 18)). Une mise-à-jour de l'état de croyance produit $b^{o_i}_{a_i}$ avant d'identifier $\alpha_i \in \Gamma$, l'α-vecteur dominant à $b^{o_i}_{a_i}$. Le nœud

2. Il peut aussi y avoir des restrictions dues aux FSC. En fait, une approche plus générale serait de ne considérer dans $\mathcal{E}$ que les états étendus atteignables depuis l'état de croyance initial.

$n' \in N$ qui contient α_i est extrait s'il existe (ligne 16), sinon un nouveau nœud $n' \stackrel{\text{def}}{=} \langle \alpha_i, b^{o_i}_{a_i} \rangle$ est créé (ligne 12) et ajouté à la fois à N et G. Un arc est ajouté entre n et n', de label o_i (ligne 19).

Comme dans le travail de GRZEŚ et al., et comme déjà mentionné, des boucles sont ajoutées quand une observation improbable $Pr(o_i|b, a_i) = 0$ est rencontrée. Cela peut arriver parce que, quand on construit le FSC, chaque nœud n_i est associé à un tuple $\langle \alpha_i, b_i \rangle$, et les arcs sortant de n_i seront créés seulement pour les observations qui ont une probabilité non-nulle dans b_i. Or, durant l'exécution, n_i peut être atteint via un état de croyance b' différent depuis lequel a_i (l'action attachée à α_i, donc à n_i) peut induire des observations "inattendues". [3] Ajouter des boucles est une façon de munir l'agent d'une stratégie par défaut quand une observation inattendue a lieu.

Il y a toutefois deux différences dans notre méthode par rapport au travail de GRZEŚ et al. :

— Premièrement, dans leur travail, le nombre de nœuds $|N|$ du FSC compilé est égal au nombre d'α-vecteurs dans $|\Gamma|$. Dans notre méthode, le nombre de nœuds est réduit puisque nous n'explorons que les états de croyance atteignables depuis l'état de croyance initial b_0, et ne stockons qu'un α-vector par état de croyance.

— Deuxièmement, dans notre cas, nous n'avons pas une mais deux raisons d'ajouter des boucles. La première raison est la même que dans les travaux de GRZEŚ et al. La seconde raison est que, dans l'approche JESP, chaque FSC d'un agent i est obtenu en considérant les FSC des agents $\neq i$ figés. Toutefois, modifier les FSC des autres agents va conduire à un nouveau POMDP du point de vue de i, avec potentiellement de nouvelles observations rencontrées depuis certains nœuds de fsc_i.

Notons que cet algorithme ne borne pas le nombre résultant de nœuds internes. À la place, nous comptons sur (i) le fait que SARSOP retourne un nombre fini d'α-vecteurs, et (ii) la production de (significativement) moins de $|\Gamma|$ nœuds internes par l'extraction de FSC.

Une fois fsc_i obtenu, l'étape suivante est l'évaluation commune de toutes les politiques des agents dans le Dec-POMDP. Dans ce but, il est suffisant d'évaluer fsc_i dans le POMDP meilleure-réponse correspondant, puisque celui-ci combine dans un seul modèle le Dec-POMDP et les FSC des agents $\neq i$. Ici, nous employons la technique d'évaluation de FSC décrite en section 3.3.

4.4 Initialisations MPOMDP

Commme mentionné précédemment, l'initialisation d'inf-JESP est importante. Même si inf-JESP améliore la valeur du FSC joint à chaque itération, des initialisations aléatoires peuvent souvent conduire à de mauvais optima locaux. Nous souhaitons donc investiguer si des heuristiques d'initialisation non-aléatoires permettent de trouver de bonnes solutions vites et fiablement. Notre méthode part de l'hypothèse d'observations publiques dans le Dec-POMDP, de sorte que nous sommes face à un POMDP multi-agent (MPOMDP) [23], c'est-à-dire un problème formellement équivalent à un POMDP et donc résolu à l'aide d'un solveur de POMDP. Nous extrayons des FSC individuels de la politique MPOMDP résultante comme détaillé ci-dessous, et les employons pour initialiser inf-JESP.

Algorithme 2 : Compilation d'un ensemble Γ d'α-vecteurs en un contrôleur à états finis $\langle N, \eta, \psi \rangle$

```
1  [Input :] Γ : α-vector set
2  Start node n_0 ← node(arg max_{α∈Γ} α · b_0, b_0)
3  N ← {n_0}
4  G.pushback(n_0)
5  while |G| > 0 do
6      n ← G.popfront()
7      (b, a_i) ← (n.b, ψ(n))
8      forall o_i ∈ Ω_i do
9          if Pr(o_i|b, a_i) > 0 then
10             α_i ← argmax_{α∈Γ} α · b^{o_i}_{a_i}
11             if α_i ∉ N then
12                 n' ← node(α_i, b^{o_i}_{a_i})
13                 N ← N ∪ {n'}
14                 G.pushback(n')
15             else
16                 n' ← N(α_i)
17         else
18             n' ← n
19         η(n, o_i) ← n'
20 return ⟨N, η, ψ⟩
```

FSC initial basé-MPOMDP stochastique (M-S) – L'algorithme 3 extrait une politique fsc_i pour l'agent i d'une politique MPOMDP. Cette approche est similaire à l'algorithme 2, la différence principale étant que les observations et actions des agents $\neq i$ devraient être aussi considérées pour calculer le prochain état de croyance. Ce n'est toutefois pas vraiment faisable puisque l'agent i ne les connaît pas. Pour résoudre ce problème, étant donné l'agent i, considérons un nœud courant $n = \langle \alpha, b \rangle$ et une observation individuelle o_i (de probabilité d'occurrence non nulle). La solution MPOMDP spécifie une action jointe $a = \langle a_i, a_{\neq i} \rangle$ en b, a étant l'action associée à α. Nous supposons arbitrairement que (i) chaque transition stochastique possible du FSC (de n et associée à l'observation o_i) correspond à une observation jointe possible $o_{\neq i}$ des autres agents, (ii) une telle transition a lieu avec probabilité $Pr(o_{\neq i} \mid b, a, o_i)$, et (iii) elle "conduit" à une nouvelle croyance $b_a^{\langle o_i, o_{\neq i} \rangle}$ et ainsi à un nœud n' attaché à l'α-vecteur MPOMDP dominant (α_i, *cf.* ligne 12) en ce point. Comme seulement un nœud FSC devrait correspondre à un α-vecteur

3. Cela arrivera si un état s a une probabilité nulle dans b_i mais pas dans b' et peut induire cette observation quand a_i est effectuée.

donné, un nouveau n' attaché à α_i n'est créé que si nécessaire (lignes 13, 14 et 18). Comme de multiples observations jointes $o_{\neq i}$ peuvent conduitre au même n', les probabilités de transition correspondantes sont cumulées dans $\eta(n, o_i, n')$ (ligne 19). Les observations jointes $o_{\neq i}$ de probabilité nulle étant donnés b, a et o_i sont ignorées. Les observations individuelles o_i de probabilité nulle étant donnés b et a induisent la création d'une boucle (ligne 21).
Notons que les FSC résultants sont stochastiques, et que certains d'entre eux pourraient ne pas être améliorables, de sorte que, dans ce cas, la solution retournée par inf-JESP pourrait contenur des FSC stochastiques.

FSC initial basé-MPOMDP déterministe (M-D) – Une variante très simple de cette méthode est de supposer que la seule transition possible de n sous o_i correspond à l'observation jointe la plus probable $o_{\neq i}$ des autres agents. La transition entre nœuds est alors déterministe ($\eta(n, o_i, n') = 1$).

Notes : (1) Comme (i) ces méthodes d'initialisation heuristiques et (ii) la recherche locale d'inf-JESP sont déterministes, utiliser une procédure déterministe pour dériver des FSC meilleures-réponses induit un algorithme déterministe pour lequel les redémarrages sont inutiles. (2) Ces méthodes d'initialisation heuristiques peuvent aussi être adaptées au cadre à horizon fini de JESP en remplaçant la représentation de politique par des arbres à horizon fini.

Algorithme 3 : Extraction de fsc_i pour l'agent i à partir de l'ensemble Γ d'α-vecteurs solution du MPOMDP

```
1  [Input :] i : agent | Γ : MPOMDP α-vector set
2  Start node n_0 ← node(argmax_{α∈Γ} α · b_0, b_0)
3  N ← {n_0}
4  G.pushback(n_0)
5  while |G| > 0 do
6      n ← G.popfront()
7      (b, a) ← (n.b, ψ(n))
8      forall o_i ∈ Ω_i do
9          if Pr(o_i|b, a) > 0 then
10             forall o_{≠i} ∈ Ω_{≠i} do
11                 if Pr(o_{≠i}|o_i, b, a) > 0 then
12                     α_i ← argmax_α(b_a^{⟨o_i,o_{≠i}⟩}, Γ)
13                     if α_i ∉ N then
14                         n' ← node(α_i, b_a^{⟨o_i,o_{≠i}⟩})
15                         N ← N ∪ {n'}
16                         G.pushback(n')
17                     else
18                         n' ← N(α_i)
19                     η(n, o_i, n') ←
                         η(n, o_i, n') + Pr(o_{≠i}|o_i, b, a)
20         else
21             η(n, o_i, n) ← 1
22 return ⟨N, η, ψ⟩
```

5 Expérimentations

Dans cette section, nous appliquons l'algorithm inf-JESP sur divers bancs d'essai Dec-POMDP standards. Comme il y a différents choix d'initialisations, nous comparons différents réglages de inf-JESP. Nous incluons aussi les résultats de plusieurs algorithmes de l'état de l'art pour résoudre des Dec-POMDP à horizon infini.

5.1 Méthode de comparaison

Nous utilisons des problèmes Dec-POMDP standards pour évaluer notre algorithme et ses variantes : DecTiger, Recycling, Meeting in a 3×3 grid (*aka* Grid), Box Pushing, et Mars Rover. Tous ces problèmes sont disponibles en ligne [4].
Nous comparons les différentes variantes d'inf-JESP avec des solveurs Dec-POMDP de l'état de l'art, à savoir : FB-HSVI [9], Peri [21], PeriEM [21], PBVI-BB [15] et MealyNLP [3].
Nous ignorons l'algorithme JESP original puisqu'il ne peut pas gérer des Dec-POMDPs à horizon infini. Nous comparons les résultats avec Dec-BPI séparément parce que nous ne pouvons qu'estimer les valeurs à partir de graphes empiriques sur quelques bancs d'essai [6].

5.2 Réglages des algorithmes

Nous avons utilisé SARSOP [14] comme solveur POMDP fournissant un ensemble solution d'α-vecteurs. Nous fixons un temps limite de 5 s à SARSOP, et un résidu de Bellman $= 0,001$ à la fois pour l'évaluation de FSC et la précision de SARSOP.
Nous avons testé quatre différentes implémentations d'inf-JESP. Les première et deuxième, IJ(M-D) et IJ(M-S), sont inf-JESP initialisé avec les heuristiques M-D et M-S (*cf.* section 4.4) et sans redémarrage parce que, sans options de randomisation et malgré le temps limite de 5 s, SARSOP se conduit de manière déterministe (comme observé empiriquement). Les troisième et quatrième, IJ(R-1_{runs}) et IJ(R-100_{runs}), sont inf-JESP avec des initialisations aléatoires et différents nombres de redémarrages (1 (re)démarrage et 100 redémarrages).
Par exemple, "IJ(R-100_5)" signifie 5 exécutions d'inf-JESP, chacun avec 100 redémarrages. Toutes les initialisations aléatoires sont bornées avec 5 nœuds pour chaque FSC. Pour chaque redémarrage, nous fixons une limite de temps de 7200 s. À cause du temps limité, nous n'avons pas effectué de tests avec redémarrage aléatoire sur les domaines Box-Pushing and Mars Rover.
Les expérimentations avec inf-JESP ont été conduites sur un portable doté d'un CPU i5-1.6 GHz et 8 GB de RAM.

5.3 Résultats

Comparaison avec les algorithmes de l'état de l'art. La table 1 présente les résultats obtenus sur les 5 problèmes Dec-POMDP standards employés comme bancs d'essai. Pour inf-JESP avec initialisation aléatoire et redémarrages,

4. http://masplan.org/problem

TABLE 1 – Comparaison de différents algorithmes en termes de taille finale des FSC, nombre d'itérations requis, temps et valeur, sur 5 bancs d'essai Dec-POMDP à horizon infini, avec $\gamma = 0.9$ dans chaque cas. R-R_N : initialisations aléatoires avec R redémarrages (évalué avec N exécutions). M-S and M-D : initialisations FSC basées-MPOMDP stochastiques et déterministes.

Alg.	FSC size	#Iterations	Time (s)	Value
DecTiger ($\|\mathcal{I}\|=2, \|\mathcal{S}\|=2, \|\mathcal{A}^i\|=3, \|\mathcal{Z}^i\|=2$)				
FB-HSVI			153,7	13,448
PBVI-BB				13,448
Peri			220	13,45
IJ(R-100_5)	$34_{\pm 34} \times 34_{\pm 34}$	22 ± 3	7361,8	13,30
IJ(M-D)	75×81	27	157	13,10
IJ(M-S)	75×79	27	164	13,10
PeriEM			6450	9,42
MealyNLP			29	$-1,49$
IJ(R-1_{500})	$52_{\pm 61} \times 53_{\pm 65}$	12 ± 10	73,62	$-67,81$
Recycling ($\|\mathcal{I}\|=2, \|\mathcal{S}\|=4, \|\mathcal{A}^i\|=3, \|\mathcal{Z}^i\|=2$)				
FB-HSVI			2,6	31,929
PBVI-BB				31,929
MealyNLP			0	31,928
Peri			77	31,84
IJ(R-100_{10})	$3_{\pm 0} \times 3_{\pm 0}$	3 ± 0	22,3	31,92
PeriEM			272	31,80
IJ(R-1_{1000})	$5_{\pm 8} \times 5_{\pm 8}$	3 ± 0	0,223	29,70
IJ(M-S)	8×8	3	0	26,57
IJ(M-D)	4×4	3	0	25,65
Grid3*3 ($\|\mathcal{I}\|=2, \|\mathcal{S}\|=81, \|\mathcal{A}^i\|=5, \|\mathcal{Z}^i\|=9$)				
IJ(M-D)	8×8	5	13	5,81
IJ(M-S)	8×8	7	88	5,81
IJ(R-100_5)	$9_{\pm 1} \times 12_{\pm 4}$	3 ± 0	3788,4	5,81
FB-HSVI			67	5,802
IJ(R-1_{500})	$11_{\pm 14} \times 14_{\pm 24}$	4 ± 1	37,88	5,40
Peri			9714	4,64
Box-pushing ($\|\mathcal{I}\|=2, \|\mathcal{S}\|=100, \|\mathcal{A}^i\|=4, \|\mathcal{Z}^i\|=5$)				
FB-HSVI			1715,1	224,43
PBVI-BB				224,12
IJ(M-S)	158×197	6	2443	220,26
IJ(M-D)	78×288	4	1564	202,94
Peri			5675	148,65
MealyNLP			774	143,14
PeriEM			7164	106,65
Mars Rover ($\|\mathcal{I}\|=2, \|\mathcal{S}\|=256, \|\mathcal{A}^i\|=6, \|\mathcal{Z}^i\|=8$)				
FB-HSVI			74,31	26,94
IJ(M-D)	19×49	7	1432	26,91
IJ(M-S)	41×41	6	891	24,20
Peri			6088	24,13
MealyNLP			396	19,67
PeriEM			7132	18,13

à chaque exécution nous avons gardé la plus haute valeur parmi les redémarrages, et ensuite calculé la moyenne de ces valeurs sur les exécutions effectuées (1 exécution = 100 redémarrages). Les cinq colonnes de la table fournissent : (*Alg.*) les différents algorithmes comparés, (*FSC size*) les tailles finales des FSC après convergence (pour les approches inf-JESP), (*#Iterations*) le nombre d'itérations requises pour atteindre un équilibre (pour les approches inf-JESP), (*Time*) le temps de calcul, (*Value*) la valeur finale de la politique jointe. Les résultats donnés pour les variantes de inf-JESP sont des minorants, de sorte que la valeur réelle des FSC obtenus pourrait être plus élevée (avec un différence maximum de $0,001$).

En termes de valeur finale atteinte, inf-JESP peut avoir des solutions raisonnablement bonnes dans la plupart des domaines avec différentes méthodes d'initialisation. Même si les valeurs obtenues avec inf-JESP ne peuvent pas concurrencer celles atteintes par FB-HSVI, dans la plupart des domaines l'écart est relativement faible. Nous avons aussi comparé nos résultats avec Dec-BPI en estimant les valeurs qu'il peut atteindre à travers les figures présentées dans [7] (page 123). Même si Dec-BPI n'a été testé que sur trois problèmes (à savoir, DecTiger, Grid, et Box-Pushing), inf-JESP surpasse Dec-BPI dans les trois cas. Il faut aussi noter que les résultats de Dec-BPI dépendent fortement de la taille des FSC considérés et que Dec-BPI rencontre des difficultés quand les FSC ont beaucoup de nœuds, comme on peut le voir sur le problème DecTiger [7].

Concernant le temps de résolution, IJ(M-S), IJ(M-D) et IJ(R-1_{500}/R-1_{1000}) ont des comportements différents selon le problème considéré. Par exemple, dans le problème DecTiger, IJ(M-S) et IJ(M-D) ont presque le même temps de résolution, et tous deux ont besoin de plus de temps que IJ(R-1_{500}). Par contre, pour le problème Grid, IJ(M-D) a besoin de significativement moins de temps que IJ(M-S) et IJ(R-1_{500}). Dans l'ensemble, IJ(M-D) et IJ(M-S) peuvent donner de bonnes solutions en un temps acceptable.

Étude d'inf-JESP. La figure 2 (gauche) présente l'évolution des valeurs des FSC pendant l'exécution de JESP en fonction du nombre d'itérations. La courbe bleue présente cette évolution en moyenne (ainsi que l'écart-type correspondant) pendant des exécutions initialisées aléatoirement, alors que les courbes rouges et vertes présentent les résultats déterministes avec les initialisation M-S et M-D. Pour les initialisations aléatoires, il est à noter que, à chaque itération, seuls les FSC n'ayant pas convergé à un optimum local sont considérés dans la moyenne. Cela explique pourquoi cette moyenne peut (i) être plus élevée que la distribution sur les valeurs finales (à gauche), et (ii) décroître, puisqu'une exécution avec une solution de haute valeur peut s'être arrêtée juste avant. Néanmoins, cette figure souligne clairement que la recherche locale d'inf-JESP avec initialisations aléatoires améliore effectivement la valeur du FSC à chaque itération jusqu'à convergence à une solution localement optimale. Elle montre aussi que les initialisations MPOMDP sont des heuristiques plutôt bonnes comparées aux initialisations aléatoires, mais, comme attendu, inf-JESP avec initialisation MPOMDP n'atteint pas toujours un optimum global (comme pour le problème Recycling).

La partie droite de la figure 2 présente la distribution des valeurs des FSC finaux pour différentes exécutions de la recherche locale d'inf-JESP avec initialisations aléatoires. Elle souligne donc la fréquence d'accès à des valeurs spécifiques après convergence d'inf-JESP(R-1_1). Il faut d'abord noter que, dans certaines exécutions, inf-JESP(R-1_1) a été capable d'atteindre un optimum global même si les FSC

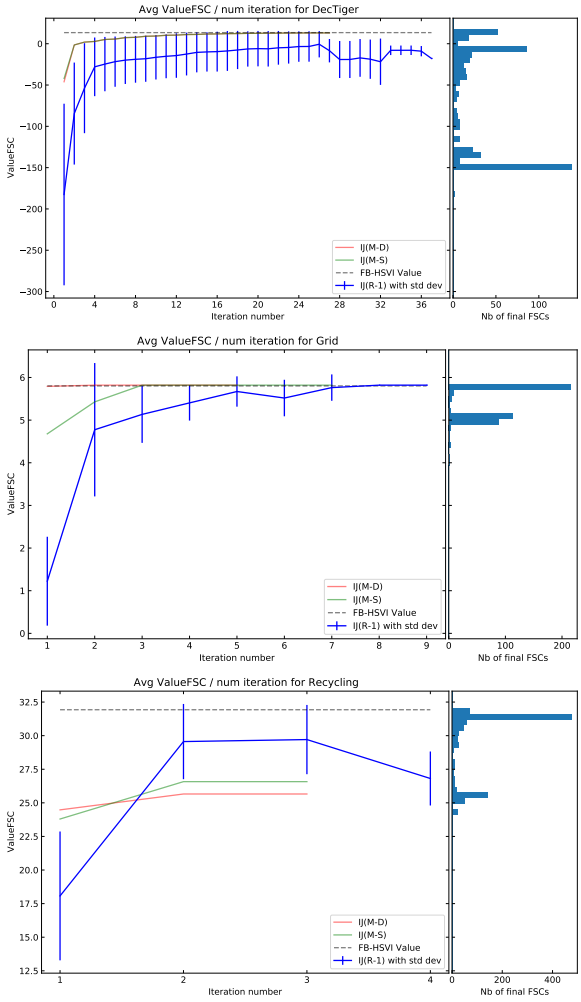


FIGURE 2 – Valeurs des politiques FSC jointes pour les bancs d'essai considérés (de haut en bas) : DecTiger, Grid and Recycling. La partie gauche de chaque figure présente l'évolution (pendant une exécution) de la valeur du FSC joint à chaque itération de inf-JESP(R-1_1) (moyenne et écart-type en bleu), et des algorithmes déterministes inf-JESPM-D (en rouge) et inf-JESPM-S (en vert). La partie droite présente la distribution des valeurs des FSC joints obtenus après convergence de inf-JESP(R-1_1). La ligne tiretée représente la valeur obtenue avec FB-HSVI.

initiaux sont limités à seulement 5 nœuds. Si cela arrive dans chacun de ces 3 problèmes, cela pourrait ne pas être le cas pour d'autres Dec-POMDP. Toutefois, cela démontre quand même que, dans certains contextes, inf-JESP peut tirer parti de petits FSC pour produire des politiques jointes utiles. En fonction du problème, les FSC joints avec des valeurs comparables à la politique optimale peuvent être difficiles à atteindre, comme pour les problèmes DecTiger ou Recycling où il semble facile d'atteindre une solution quasi-optimale sans avoir une forte probabilité d'atteindre un FSC joint globalement optimal. Cette distribution donne aussi une idée générale du nombre de redémarrages nécessaires pour atteindre un optimum global avec forte probabilité. Par exemple, inf-JESP aurait probablement besoin de moins de redémarrages pour trouver un FSC joint optimal pour le problème Grid que pour DecTiger.

Résultats complémentaires. Les expérimentations conduites ont aussi montré qu'inf-JESP exhibe des comportements différents selon le problème abordé. Les résultats obtenus sont décrits en détails dans l'annexe B. Il est à noter que l'élimination d'états (comme expliqué en section 4.2) se comporte différemment selon le problème considéré. Elle peut être très efficace, comme pour le problème Grid, où elle réduit le nombre d'états du POMDP meilleure-réponse par un facteur 10 à chaque itération. Par contre, comme attendu, l'élimination d'états est sans effet sur le problème DecTiger parce que n'importe quelle observation peut être reçue depuis n'importe quel état de ce Dec-POMDP. Nous avons aussi examiné les tailles des FSC finaux obtenus par inf-JESP avec initialisations aléatoires. Nous observons que ceux dont les valeurs sont élevées ne sont pas ceux qui requièrent le plus grand nombre d'itérations ou ayant le plus grand nombre de nœuds. Des FSC de petite taille semblent aussi suffisants pour induire de hautes valeurs, ce qui ouvre une direction de recherche intéressante pour combiner inf-JESP avec de la compression de FSC.

6 Conclusion

Cet article apporte deux contributions principales.
Premièrement, nous avons proposé un nouveau solveur de Dec-POMDP à horizon infini appelé inf-JESP. Ce solveur repose sur l'approche JESP et une nouvelle formalisation qui combine un problème Dec-POMDP avec des FSC connus pour construire un POMDP mono-agent meilleure-réponse. Nous avons traité la limitation de JESP aux horizons finis en tirant profit de solveurs à base de points modernes tels que SARSOP. Nous avons comparé les résultats d'inf-JESP avec plusieurs solveurs de Dec-POMDP de l'état de l'art à travers des expérimentations sur cinq bancs d'essai. Même si, comme JESP, inf-JESP ne converge que vers des optima locaux, les résultats obtenus montrent qu'inf-JESP avec des initialisations aléatoires ou basées-MPOMDP peut quand même fournir des politiques jointes utiles pour les cinq problèmes considérés. Notre nouvelle formalisation de POMDP meilleure-réponse pourrait aussi être utile en elle-même (hors d'inf-JESP), dans des cadres où les comportements d'autres agents sont définis indépendamment, comme dans des jeux [18] ou pour l'interaction homme-robot [8].
Deuxièmement, nous avons fourni une méthode pour extraire des politiques individuelles (FSC) de l'ensemble solution Γ d'un MPOMDP pour initialiser inf-JESP. Cette approche peut être facilement adaptée à JESP pour des horizons finis. Des résultats empiriques ont montré que cette méthode d'initialisation peut, dans certains cas, atteindre des solutions de bonne qualité avec une valeur plus éle-

vée que la solution moyenne d'inf-JESP avec initialisation aléatoire. Cette approche ne marche toutefois pas toujours. Dans le problème Recycling, elle est pire que la valeur moyenne d'inf-JESP avec initialisations aléatoires. Comment dériver de meilleures heuristiques (éventuellement randomisées) de politiques MPODMP reste une question ouverte.
Les travaux futurs incluent des expérimentations avec (i) des FSC de taille bornée ; (ii) différentes formalisations des POMDP meilleures-réponses ; (iii) différentes durées limites pour SARSOP ; (iv) des redémarrages parallèles.

Références

[1] D. ABERDEEN. « Policy-Gradient Algorithms for Partially Observable Markov Decision Processes ». Thèse de doct. Canberra, Australia : The Australian National University, mar. 2003.

[2] C. AMATO, D. S. BERNSTEIN et S. ZILBERSTEIN. « Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs ». *Journal of Autonomous Agents and Multi-Agent Systems* 21.3 (2010), p. 293-320. DOI : 10.1007/s10458-009-9103-z.

[3] C. AMATO, B. BONET et S. ZILBERSTEIN. « Finite-State Controllers Based on Mealy Machines for Centralized and Decentralized POMDPs ». Dans : *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. Sous la dir. de M. FOX et D. POOLE. AAAI Press, 2010.

[4] M. ARAYA-LÓPEZ, V. THOMAS, O. BUFFET et F. CHARPILLET. « A Closer Look at MOMDPs ». Dans : *Proceedings of the Twenty-Second IEEE International Conference on Tools with Artificial Intelligence (ICTAI-10)*. Arras, France, 2010.

[5] D. BERNSTEIN, R. GIVAN, N. IMMERMAN et S. ZILBERSTEIN. « The Complexity of Decentralized Control of Markov Decision Processes ». *Mathematics of Operations Research* 27.4 (2002), p. 819-840.

[6] D. S. BERNSTEIN, C. AMATO, E. A. HANSEN et S. ZILBERSTEIN. « Policy Iteration for Decentralized Control of Markov Decision Processes ». *Journal of Artificial Intelligence Research* 34 (2009), p. 89-132. DOI : 10.1613/jair.2667.

[7] D. S. BERNSTEIN, E. A. HANSEN et S. ZILBERSTEIN. « Bounded Policy Iteration for Decentralized POMDPs ». Dans : *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 2005.

[8] A. BESTICK, R. BAJCSY et A. DRAGAN. « Implicitly Assisting Humans to Choose Good Grasps in Robot to Human Handovers ». Dans : *2016 International Symposium on Experimental Robotics*. Mar. 2017, p. 341-354. ISBN : 978-3-319-50114-7. DOI : 10.1007/978-3-319-50115-4_30.

[9] J. DIBANGOYE, C. AMATO, O. BUFFET et F. CHARPILLET. « Optimally Solving Dec-POMDPs as Continuous-State MDPs ». *Journal of Artificial Intelligence Research* 55 (2016), p. 443-497.

[10] M. GRZEŚ, P. POUPART, X. YANG et J. HOEY. « Energy Efficient Execution of POMDP Policies ». *IEEE Transactions on Cybernetics* 45 (2015), p. 2484-2497. DOI : 10.1109/TCYB.2014.2375817.

[11] E. HANSEN. « An Improved Policy Iteration Algorithm for Partially Observable MDPs ». Dans : *Advances in Neural Information Processing Systems*. Sous la dir. de M. JORDAN, M. KEARNS et S. SOLLA. T. 10. MIT Press, 1998, p. 1015-1021.

[12] E. HANSEN. « Solving POMDPs by Searching in Policy Space ». Dans : *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. 1998.

[13] A. KUMAR, S. ZILBERSTEIN et M. TOUSSAINT. « Probabilistic Inference Techniques for Scalable Multiagent Decision Making ». *Journal of Artificial Intelligence Research* 53 (2015), p. 223-270. DOI : 10.1613/jair.4649.

[14] H. KURNIAWATI, D. HSU et W. LEE. « SARSOP : Efficient point-based POMDP planning by approximating optimally reachable belief spaces ». Dans : *Robotics : Science and Systems IV*. 2008.

[15] L. C. MACDERMED et C. ISBELL. « Point Based Value Iteration with Optimal Belief Compression for Dec-POMDPs ». Dans : *Advances in Neural Information Processing Systems 26*. 2013.

[16] N. MEULEAU, K.-E. KIM, L. KAELBLING et A. CASSANDRA. « Solving POMDPs by searching the space of finite policies ». Dans : *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*. 1999, p. 417-426.

[17] R. NAIR, M. TAMBE, M. YOKOO, D. PYNADATH et S. MARSELLA. « Taming decentralized POMDPs : Towards efficient policy computation for multiagent settings ». Dans : *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*. 2003.

[18] F. OLIEHOEK, M. SPAAN et N. VLASSIS. « Best-response play in partially observable card games ». *Benelearn 2005 : Proceedings of the 14th Annual Machine Learning Conference of Belgium and the Netherlands* (jan. 2005).

[19] S. ONG, S. PNG, D. HSU et W. LEE. « POMDPs for robotic tasks with mixed observability ». Dans : *Proceedings of Robotics : Science and Systems V (RSS'09)*. 2009.

[20] J. PAJARINEN et J. PELTONEN. « Efficient Planning for Factored Infinite-Horizon DEC-POMDPs ». Dans : *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. AAAI Press, juil. 2011, p. 325-331. DOI : 10.5591/978-1-57735-516-8/IJCAI11-064.

[21] J. PAJARINEN et J. PELTONEN. « Periodic Finite State Controllers for Efficient POMDP and DEC-POMDP Planning ». Dans : *Advances in Neural Information Processing Systems 24*. Sous la dir. de J. SHAWE-TAYLOR, R. ZEMEL, P. BARTLETT, F. PEREIRA et K. Q. WEINBERGER. T. 24. Curran Associates, Inc., 2011.

[22] J. PINEAU, G. GORDON et S. THRUN. « Point-Based Value Iteration : An Anytime Algorithm for POMDPs ». Dans : *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. IJCAI'03. Acapulco, Mexico : Morgan Kaufmann Publishers Inc., 2003, p. 1025-1030.

[23] D. V. PYNADATH et M. TAMBE. « The Communicative Multiagent Team Decision Problem : Analyzing Teamwork Theories and Models ». *Journal of Artificial Intelligence Research* 16 (juin 2002), p. 389-423. ISSN : 1076-9757. DOI : 10.1613/jair.1024.

[24] T. SMITH et R. SIMMONS. « Heuristic Search Value Iteration for POMDPs ». Dans : *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*. UAI '04. Banff, Canada : AUAI Press, 2004, p. 520-527. ISBN : 0974903906.

[25] D. SZER, F. CHARPILLET et S. ZILBERSTEIN. « MAA* : A Heuristic Search Algorithm for Solving Decentralized POMDPs ». Dans : *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. 2005, p. 576-583.

[26] N. TAO, J. BAXTER et L. WEAVER. « A Multi-Agent, Policy-Gradient approach to Network Routing ». Dans : *Proceedings of the Eighteenth International Conference on Machine Learning*. 2001.

A Notes sur les formalisations POMDP candidates

Nous regardons plus en détails comment, étant donnés le Dec-POMDP et les FSC de tous les agents sauf i, on peut dériver un POMDP meilleure-réponse valide du point de vue de l'agent i. Notons d'abord que, dans la formalisation POMDP :

- nous n'avons **aucun choix** pour l'observation (o_i^t) et pour l'action (a_i^t), qui sont des pré-requis ;
- le **seul choix** qui peut être fait est celui des variables mises dans l'état e_i^t ;
- les fonctions de transition, d'observation et de récompense sont des **conséquences** de ce choix.

Notons aussi que, dans une formalisation POMDP, les variables d'état et d'observation doivent être distinguées. On ne peut pas écrire que X est une variable d'état observée (contrairement au formalisme MOMDP). Dans ce qui suit, il est ainsi important de distinguer les différents types de variables aléatoires. En particulier,

- O_i^t dénote toujours (évidemment) une *variable d'observation*, mais
- $\tilde{O}_i^t$ dénote soit une *variable intermédiaire*[5], soit une *variable d'état* (qui, dans les deux cas, est fortement dépendante de la variable d'observation O_t^i puisque leurs valeurs sont toujours égales).

Le principal problème dans la conception de notre POMDP meilleure-réponse est de vérifier que les dépendances dans les fonctions de transition, d'observation, et de récompense sont appropriées (voir figure 3 (haut gauche)). Dans les paragraphes qui suivent, nous considérons 3 choix pour l'*état étendu* du POMDP, vérifions s'ils induisent effectivement des POMDP valides, et dérivons les fonctions de transition, d'observation et de récompense induites le cas échéant.

$e^t = \langle s^t, n_{\neq i}^t \rangle$ **? –** Pour montrer que $e^t = \langle s^t, n_{\neq i}^t \rangle$ n'induit pas un POMDP convenable, écrivons la fonction de transition :

$$
\begin{aligned}
&T_e(e^t, a_i^t, e^{t+1}) \stackrel{\text{def}}{=} Pr(e^{t+1}|e^t, a_i^t) \\
&= Pr(s^{t+1}, n_{\neq i}^{t+1}|s^t, n_{\neq i}^t, a_i^t) \\
&= \sum_{\tilde{o}^{t+1}} Pr(s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}^{t+1}|s^t, n_{\neq i}^t, a_i^t) \\
&= \sum_{\tilde{o}^{t+1}} Pr(n_{\neq i}^{t+1}|s^t, n_{\neq i}^t, a_i^t, s^{t+1}, \tilde{o}^{t+1}) \\
&\quad \cdot Pr(s^{t+1}, \tilde{o}^{t+1}|s^t, n_{\neq i}^t, a_i^t) \\
&= \sum_{\tilde{o}^{t+1}} Pr(n_{\neq i}^{t+1}|s^t, n_{\neq i}^t, a_i^t, s^{t+1}, \tilde{o}^{t+1}) \\
&\quad \cdot Pr(\tilde{o}^{t+1}|s^t, n_{\neq i}^t, a_i^t, s^{t+1}) \cdot Pr(s^{t+1}|s^t, n_{\neq i}^t, a_i^t) \\
&= \sum_{\tilde{o}^{t+1}} \left(\prod_{j \neq i} \eta(n_j^t, \tilde{o}_j^{t+1}, n_j^{t+1}) \right) \\
&\cdot O(\langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}, \tilde{o}^{t+1}) \cdot T(s^t, \langle \psi_{\neq i}(n_{\neq i}^t), a_i^t \rangle, s^{t+1}),
\end{aligned}
$$

où $\tilde{O}^{t+1}$ est une variable temporaire, pas une variable d'état ou d'observation. Ici, comme illustré par la figure 3 (haut droite), le problème est que la variable d'observation O_i^{t+1} n'est pas indépendante de E^t étant donné E^{t+1} et A^t.

$e^t = \langle s^t, n_{\neq i}^t, \tilde{o}_i^t \rangle$ **? –** Nous corrigeons la première tentative en ajoutant une variable d'état $\tilde{O}_i^t$, définissant ainsi $e^t = \langle s^t, n_{\neq i}^t, \tilde{o}_i^t \rangle$, d'où :

$$
\begin{aligned}
&T_e(e^t, a_i^t, e^{t+1}) \stackrel{\text{def}}{=} Pr(e^{t+1}|e^t, a_i^t) \\
&= Pr(s^{t+1}, n_{\neq i}^{t+1}, \tilde{o}_i^{t+1}|s^t, n_{\neq i}^t, \tilde{o}_i^t, a_i^t)
\end{aligned}
$$

5. Une telle variable n'apparaît usuellement pas dans le formalisme POMDP, mais est requise ici pour calculer les fonctions de transition et d'observation à l'aide du modèle Dec-POMDP et des FSC.

$$= \sum_{o^{t+1}_{\neq i}} Pr(s^{t+1}, n^{t+1}_{\neq i}, o^{t+1}_{i-1}, \tilde{o}^t_i | s^t, n^t_{\neq i}, \tilde{o}^t_i, a^t_i)$$
$$= \sum_{o^{t+1}_{\neq i}} Pr(n^{t+1}_{\neq i} | s^t, n^t_{\neq i}, \tilde{o}^t_i, a^t_i, s^{t+1}, o^{t+1})$$
$$\cdot Pr(s^{t+1}, o^{t+1}_{\neq i}, \tilde{o}^{t+1}_i | s^t, n^t_{\neq i}, \tilde{o}^t_i, a^t_i)$$
$$= \sum_{o^{t+1}_{\neq i}} Pr(n^{t+1}_{\neq i} | s^t, n^t_{\neq i}, \tilde{o}^t_i, a^t_i, s^{t+1}, o^{t+1})$$
$$\cdot Pr(o^{t+1}_{\neq i}, \tilde{o}^{t+1}_i | s^t, n^t_{\neq i}, \tilde{o}^t_i, a^t_i, s^{t+1}) \cdot Pr(s^{t+1} | s^t, n^t_{\neq i}, \tilde{o}^t_i, a^t_i)$$
$$= \sum_{o^{t+1}_{\neq i}} \Big(\prod_{j \neq i} \eta(n^t_j, o^{t+1}_j, n^{t+1}_j) \Big)$$
$$\cdot O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, o^{t+1}) \cdot T(s^t, \langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}).$$

La formule ci-dessus ne soulève pas le même problème que précédemment puisque $\tilde{O}^t_i$ est une variable d'état, et la variable d'observation O^t_i ne dépend plus maintenant de l'état précédent étant donné le nouveau et l'action (*cf.* figure 3 (bas gauche)). Nous avons aussi la fonction d'observation suivante :

$$O_e(a^t_i, e^{t+1}_i, o^{t+1}_i) \stackrel{\text{def}}{=} Pr(o^{t+1}_i | a^t_i, e^{t+1}_i)$$
$$= Pr(o^{t+1}_i | a^t_i, \langle s^{t+1}, n^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle) = \mathbf{1}_{o^{t+1}_i = \tilde{o}^{t+1}_i},$$

et la fonction de récompense triviale :

$$r_e(e^t, a^t_i) = r(s^t, a^t_i, \psi_{\neq i}(n^t_{\neq i})).$$

$e^t = \langle s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i} \rangle$ **?** – Dans ce travail, nous avons aussi considéré un troisième choix pour l'état étendu, défini comme $e^t = \langle s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i} \rangle$, où $\tilde{O}^t_{\neq i}$ est une variable d'état (pas d'observation) correspondant aux observations des autres agents au temps t :

$$T_e(e^t, a^t_i, e^{t+1}) \stackrel{\text{def}}{=} Pr(e^{t+1} | e^t, a^t_i)$$
$$= Pr(s^{t+1}, n^t_{\neq i}, \tilde{o}^{t+1}_{\neq i} | s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i)$$
$$= Pr(\tilde{o}^{t+1}_{\neq i} | s^{t+1}, n^t_{\neq i}, s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i)$$
$$\cdot Pr(s^{t+1}, n^t_{\neq i} | s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i)$$
$$= Pr(\tilde{o}^{t+1}_{\neq i} | s^{t+1}, n^t_{\neq i}, s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i)$$
$$\cdot Pr(s^{t+1} | n^t_{\neq i}, s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i)$$
$$\cdot Pr(n^t_{\neq i} | s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}, a^t_i)$$
$$= Pr(s^{t+1} | s^t, n^t_{\neq i}, a^t_i) \cdot Pr(n^t_{\neq i} | n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i})$$
$$\cdot Pr(\tilde{o}^{t+1}_{\neq i} | s^{t+1}, n^t_{\neq i}, a^t_i)$$
$$= Pr(s^{t+1} | s^t, n^t_{\neq i}, a^t_i) \cdot Pr(n^t_{\neq i} | n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i})$$
$$\cdot \sum_{\tilde{o}^{t+1}_i} Pr(\tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i | s^{t+1}, n^t_{\neq i}, a^t_i)$$
$$= T(s^t, \langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}) \cdot \prod_{j \neq i} \eta(n^{t-1}_j, o^t_j, n^t_j)$$
$$\cdot \sum_{\tilde{o}^{t+1}_i} O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle).$$

La formule ci-dessus ne soulève pas de problèmes. Comme illustré par la figure 3 (bas droite), la variable d'observation O^{t+1}_i dépend de E^t étant donnés E^{t+1} et A^t. Nous avons aussi la fonction d'observation suivante :

$$O_e(a^t_i, e^{t+1}_i, o^{t+1}_i) \stackrel{\text{def}}{=} Pr(o^{t+1}_i | a^t_i, e^{t+1}_i)$$
$$= Pr(o^{t+1}_i | a^t_i, \langle s^{t+1}, n^t_{\neq i}, \tilde{o}^{t+1}_{\neq i} \rangle)$$
$$= \frac{Pr(\tilde{o}^{t+1}_{\neq i}, o^{t+1}_i, s^{t+1}, n^t_{\neq i}, a^t_i)}{Pr(\tilde{o}^{t+1}_{\neq i}, s^{t+1}, n^t_{\neq i}, a^t_i)}$$
$$= \frac{Pr(\tilde{o}^{t+1}_{\neq i}, o^{t+1}_i | s^{t+1}, n^t_{\neq i}, a^t_i)}{\sum_{o^{t+1}_i} Pr(\tilde{o}^{t+1}_{\neq i}, o^{t+1}_i | s^{t+1} n^t_{\neq i}, a^t_i)}$$
$$= \frac{O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, o^{t+1}_i \rangle)}{\sum_{\tilde{o}^{t+1}_i} O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle)}.$$

On peut voir que le dénominateur est identique à la dernière partie de la fonction de transition. En pratique, lors du calcul des probabilités de transition, nous stockerons les valeurs pour $\sum_{\tilde{o}^{t+1}_i} O(\langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle, s^{t+1}, \langle \tilde{o}^{t+1}_{\neq i}, \tilde{o}^{t+1}_i \rangle)$ de manière à les réutiliser los du calcul de la fonction d'observation. Finalement, la fonction de récompense est obtenue avec :

$$r_e(e^t, a^t_i) = r(\langle s^t, n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i} \rangle, a^t_i)$$
$$= \sum_{n^t_{\neq i}} Pr(n^t_{\neq i} | n^{t-1}_{\neq i}, \tilde{o}^t_{\neq i}) \cdot r(s^t, n^{t-1}_{\neq i}, a^t_i)$$
$$= \sum_{n^t_{\neq i}} \Big(\prod_{j \neq i} \eta(n^{t-1}_j, \tilde{o}^t_j, n^t_j) \Big) \cdot r(s^t, \langle \psi_{\neq i}(n^t_{\neq i}), a^t_i \rangle).$$

Différentes formalisations conduisent à différents espaces d'états (étendus) avec différentes tailles, de sorte que les complexités en temps et en espace de la génération de ce POMDP meilleure-réponse ou de sa résolution dépendront de ce choix de formalisation. Le meilleur choix dépendra du Dec-POMDP considéré, et éventuellement des FSC courants. Nous avons opté pour la plus simple des deux formalisations présentées ci-dessus (que nous appelons "formalisation MOMDP" parce qu'une des variables d'état est complètement observable), mais avons observé peu de différences en pratique dans nos expérimentations.

B Figures supplémentaires

Concernant l'**élimination d'états** (comme expliquée en section 4.2), il se trouve que, pour le POMDP meilleure-réponse "MOMDP", le ratio du nombre initial d'états (étendus) sur le nombre d'états après élimination dépend du problème considéré mais pas de l'exécution de inf-JESP (voir figure 4). Ce ratio est de 1 pour DecTiger, 9 pour Grid, et 2 pour Recycling. Ces différences sont dues à la stochasticité du processus d'observation, laquelle limite voire prévient l'élimination d'états. Actuellement, l'élimination d'états repose sur la probabilité de générer une observation donnée depuis un état en considérant les actions possibles. Toutefois, pour le problème DecTiger, chaque observation

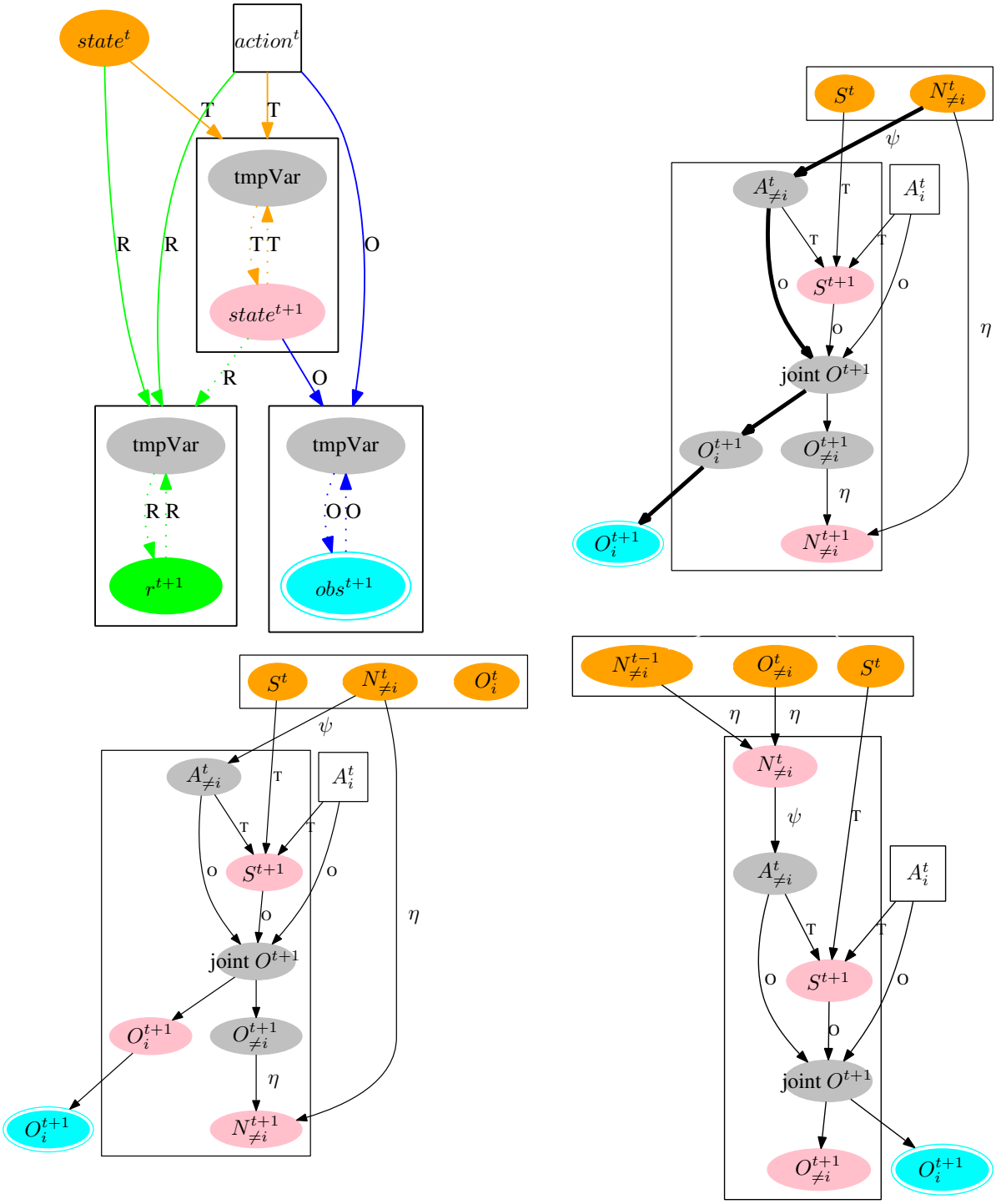


FIGURE 3 – (haut gauche) Dépendances standard d'un POMDP, incluant des variables intermédiaires employées pour calculer la fonction de transition (seulement) ; et 2 formalisations candidates pour un POMDP meilleure-réponse avec : (haut droite) $e^t_i \stackrel{\text{def}}{=} \langle s^t, n^t_{\neq i} \rangle$, (bas gauche) $e^t_i \stackrel{\text{def}}{=} \langle s^t, n^t_{\neq i}, o^t_i \rangle$, et (bas droite) $e^t_i \stackrel{\text{def}}{=} \langle s^t, n^{t-1}_{\neq i}, o^t_{\neq i} \rangle$.

peut être générée quelle que soit l'action considérée, ce qui empêche toute élimination d'état (contrairement aux problèmes Grid et Recycling).

Concernant la **taille des FSC finaux** obtenus après convergence d'une exécution d'inf-JESP avec initialisation aléatoire (voir figure 5), nous avons aussi observé différents comportements. Appliqué au problème DecTiger, inf-JESP conduit à une distribution très dispersée des tailles des FSC finaux. Appliqué au problème Recycling, inf-JESP conduit aussi à une distribution dispersée des tailles des FSC fi-

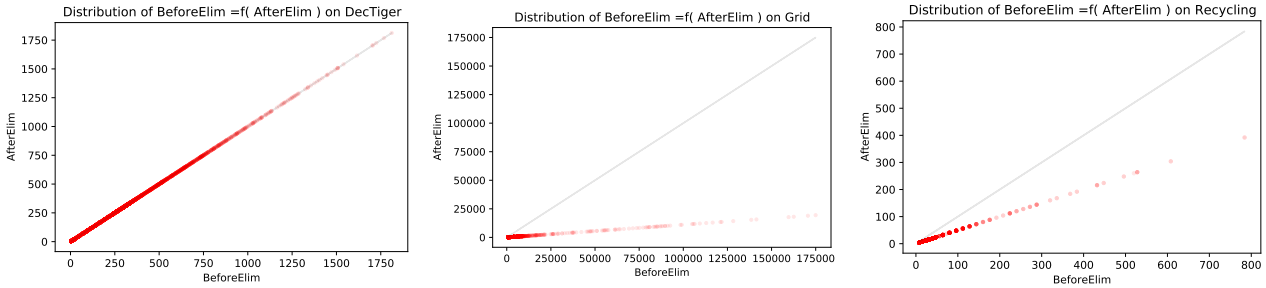


FIGURE 4 – Effet de l'élimination d'états pour les problèmes DecTiger, Grid et Recycling (de gauche à droite). Chaque figure trace, pour chaque POMDP meilleure-réponse obtenu en exécutant inf-JESP(R-1_1), le nombre d'états de ce POMDP après élimination en fonction de ce nombre avant élimination.

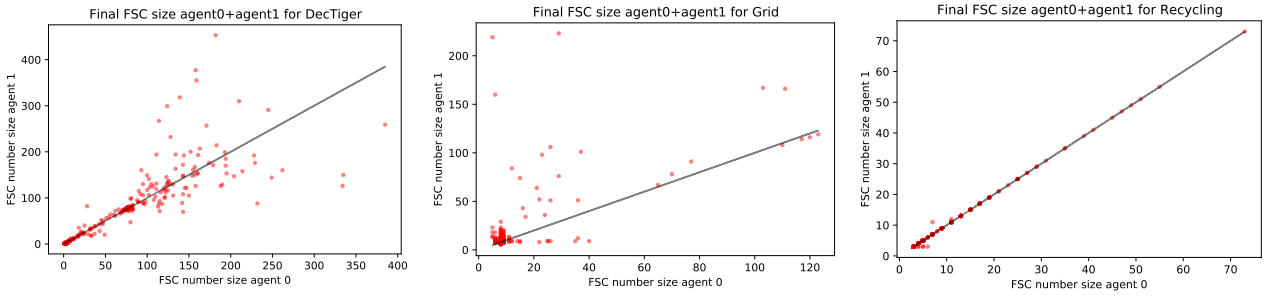


FIGURE 5 – Tailles de FSC finaux obtenus avec inf-JESP(R-1_1) après convergence pour les problèmes DecTiger, Grid et Recycling (de gauche à droite). chaque point correspond à une paire contenant les tailles des FSC des deux agents.

naux, mais les tailles des FSC des deux agents sont symétriques. Appliqué au problème Grid, inf-JESP génère un grand nombre de paires de FSC de taille 10 avec parfois d'énormes variations et avec une tendance à l'assymétrie, le premier agent optimisé ayant une tendance à avoir plus de nœuds dans son FSC. Ces résultats nécessitent des investigations supplémentaires pour les comprendre clairement et voir s'il est possible de les lier à la nature du domaine abordé.

Il faut aussi noté que les **tailles des FSC** ne croissent pas de manière monotone pendant une itération d'inf-JESP (données non présentées ici). Parfois la taille du FSC calculé pendant une amélioration d'inf-JESP décroît, ce qui signifie que la solution au POMDP meilleure-réponse est un FSC de plus petite taille mais de plus grande valeur (comme observé couramment dans le problème DecTiger). En regardant les FSC obtenus à la fin d'une exécution d'inf-JESP, un autre phénomène observé est que plus le nombre d'itérations requis pour atteindre l'équilibre est grand, plus petits sont les FSC finaux (voir figure 6). Mais cela ne signifie pas que la valeur associée est plus grande (voir figure 7).

Finalement, la figure 8 présente les valeurs obtenues en fonction de la somme des tailles des FSC obtenus par inf-JESP(R-1_1) après convergence. On peut observer que de petits FSC semblent suffisants pour générer une valeur élevée, ce qui ouvre de nouvelles directions combinant inf-JESP avec la compression de FSC.

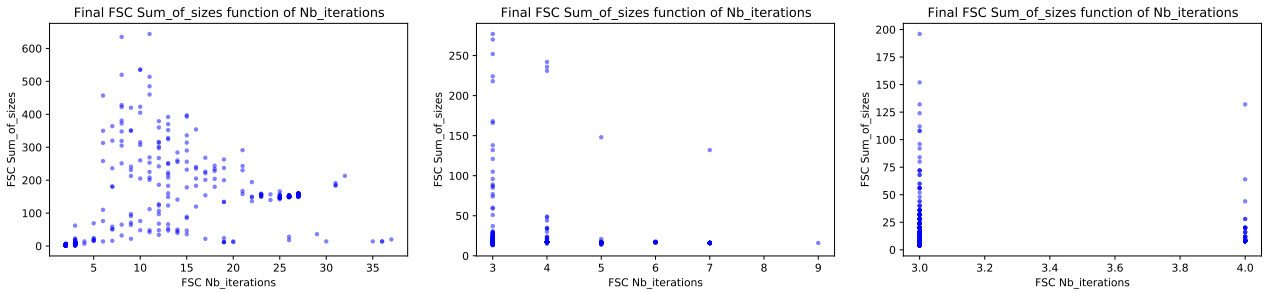


FIGURE 6 – Somme des tailles des FSC finaux obtenus avec inf-JESP(R-1_1) après convergence en fonction du nombre d'itérations requis pour les problèmes DecTiger, Grid et Recycling (de gauche à droite).

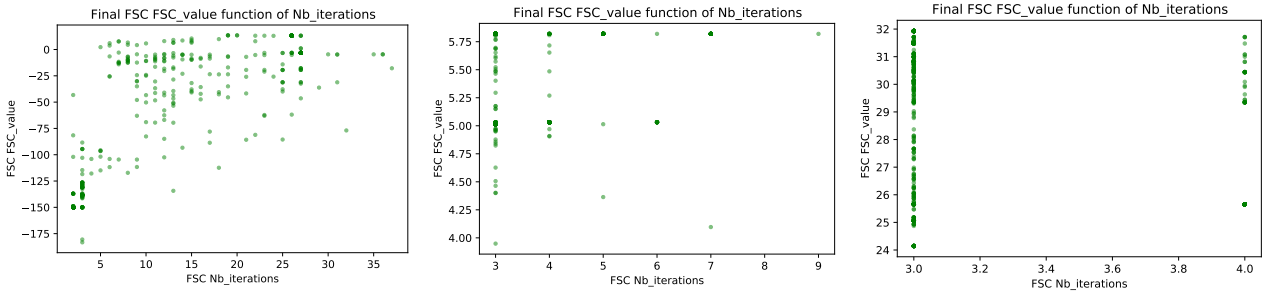


FIGURE 7 – Valeur des FSC finaux obtenus avec inf-JESP(R-1_1) après convergence en fonction du nombre requis d'itérations pour les problèmes DecTiger, Grid et Recycling (de gauche à droite).

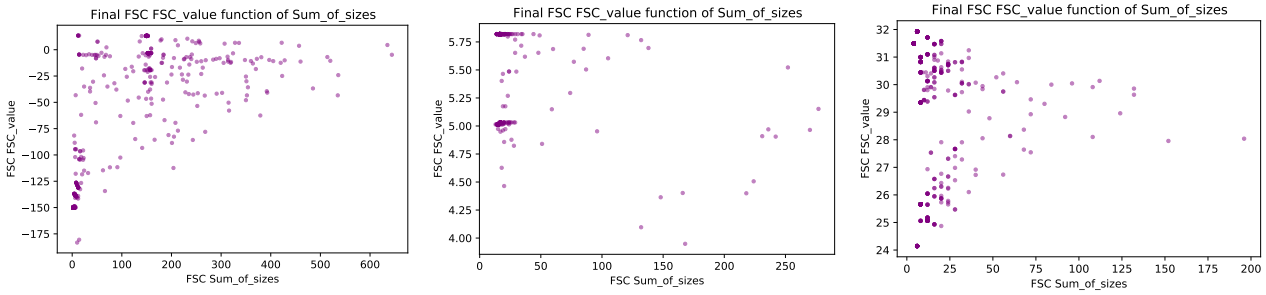


FIGURE 8 – Valeur des FSC finaux obtenus avec inf-JESP(R-1_1) après convergence en fonction de la somme des tailles des FSC finaux pour les problèmes DecTiger, Grid et Recycling (de gauche à droite).

Vérification symbolique de modèles pour la logique épistémique dynamique probabiliste

Sébastien Gamblin[1] Alexandre Niveau[1] Maroua Bouzid[1]

[1] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

prénom.nom@unicaen.fr

Résumé

La logique épistémique dynamique probabiliste (PDEL) permet de raisonner sur les connaissances probabilistes imbriquées des agents et sur les changements de connaissances induits par l'occurrence d'événements. Bien que PDEL a été étudiée de manière théorique, elle n'a été appliquée qu'à des exemples jouets : en effet, l'explosion combinatoire des structures de Kripke ne permet pas d'utiliser directement le formalisme pour des applications réelles, comme des jeux de cartes.

Le présent article constitue un premier pas vers l'utilisation de PDEL en pratique : dans la continuité des récents travaux sur la vérification de modèles de DEL (non probabiliste), nous proposons une représentation "symbolique" des structures de Kripke probabilistes avec des fonctions pseudo-booléennes, représentables par des structures de données de la famille des diagrammes de décision, en particulier les diagrammes de décisions algébriques (ADDs). Nous montrons que les ADDs passent mieux à l'échelle que les structures de Kripke explicites, et permettent une vérification symbolique de modèles efficace, même sur un exemple réaliste du jeu de carte Hanabi, ouvrant ainsi la voie à l'application pratique de techniques de planification épistémique.

Mots Clef

Vérification symbolique de modèles, Logique épistémique dynamique probabiliste, PDEL, Structure de Kripke, Planification, Hanabi

Abstract

Probabilistic Dynamic Epistemic Logic (PDEL) is a formalism for reasoning about the higher-order probabilistic knowledge of agents, and about how this knowledge changes when events occur. While PDEL has been studied for its theoretical appeal, it was only ever applied to toy examples: the combinatorial explosion of probabilistic Kripke structures makes the PDEL framework impractical for realistic applications, such as card games.

This paper is a first step towards the use of PDEL in more practical settings: in line with recent work applying ideas from symbolic model checking to (non-probabilistic) DEL, we propose a "symbolic" representation of probabilistic Kripke structures as pseudo-Boolean functions, which can be represented with several data structures of the decision diagram family, in particular Algebraic Decision Diagrams (ADDs). We show that ADDs scale much better than explicit Kripke structures, and that they allow for efficient symbolic model checking, even on the realistic example of the Hanabi card game, thus paving the way towards the practical application of epistemic planning techniques.

Keywords

Symbolic model checking, Probabilistic dynamic epistemic logic, PDEL, Kripke structures, Planification, Hanabi

1 Introduction

The card game Hanabi [Bauza, 2010] has recently drawn interest from the AI community [Bard *et al.*, 2020]. Hanabi is a multiplayer cooperative game with imperfect information in which *higher-order knowledge* plays a very important role, i.e., players need to make decisions depending on what they know about what other players know, and so on. With the ultimate goal of computing strategies for games like Hanabi, our focus is on approaches based on Dynamic Epistemic Logic (DEL) [Kooi, 2003], a formalism allowing one to reason about the higher-order knowledge of agents, and about how this knowledge changes when events occur. DEL constructs allow for an elegant approach to multi-agent epistemic planning [Bolander, 2017], and more recently, to the problems of controller and distributed strategy synthesis in adversarial games [Maubert *et al.*, 2019].

However, in many realistic games, in particular with imperfect or incomplete information, such as Hanabi, there is clearly no winning strategy for any player; what one usually wants in these cases is an *optimal* strategy, one that maximizes the expectation of a victory. A natural direction is to study an adaptation of epistemic planning to Probabilistic Dynamic Epistemic Logic (PDEL), a generalization of DEL which is interpreted on Kripke structures augmented with probabilistic information.

Yet, while there has been effort lately to make DEL useable

in practice, by applying ideas from symbolic model checking, thus avoiding the combinatorial explosion of explicit Kripke structures [Charrier et Schwarzentruber, 2017; van Benthem *et al.*, 2017; Gattinger, 2018; Charrier *et al.*, 2019], there has been no such work for PDEL; to the best of our knowledge, it remains an entirely theoretical framework, only ever applied on toy examples.

This paper is a first step towards the use of PDEL in more practical settings: we propose a "symbolic" representation of stochastic Kripke structures as pseudo-Boolean functions, which can be represented with several data structures of the decision diagram family. Our experiments using Algebraic Decision Diagrams (ADDs) on a modelization of Hanabi show that the size of these representations scale much better than explicit Kripke structures, while allowing for efficient symbolic model checking, even for near-realistic game sizes. The next step is to generalize epistemic planning to a probabilistic setting, so as to study optimal strategy synthesis for games with imperfect or incomplete information; our results indicate that such a generalization would not only be of theoretical interest, but could also be useful in practice.

After presenting some background, such as Hanabi or PDEL (2), we introduce our symbolic representation and model checking procedure (3), then report about experimental results (4).

2 Background

2.1 Hanabi

Hanabi [Bauza, 2010] is a cooperative card game where players must play their cards in order; the specificity is that players can never look at their cards, although they see those of other players and can give them information. There are 5 card colors (red, blue, green, yellow, white) and 5 card values (numbers from 1 to 5, with the following repartition: 3×1, $2 \times 2\,3\,4$, and 1×5) for each color. The goal is to play as many cards as possible on the table, but the cards of each color must be played in increasing order (an error costs one red token, out of three in total). At their turn, each player can either play a card on the table, give information to another player (which costs one blue token), or discard a card (which gives one blue token back). Giving information is limited to announcing all cards of a given color or value in a the hand of another player, by pointing to the cards. Figure 1 shows a game situation. See han [2021] for the precise rules of the game.

We will represent the "physical" state of the game (independant from the knowledge acquired by the players) using propositional variables, which basically describe where each card is located. For example, variable card^{ap}_{cvi} describes the fact that player a has the card of color $c \in \{W,R,B,Y,G\}$ and value $v \in \{1..5\}$ of id i (there are several cards with the same color and value), at position p in their hand. Variable card^{z}_{cvi} is the same, but with $z \in \{Draw, Discard, Table\}$ and with no notion of position. Still other propositional variables count the number of tokens (R_i or B_i) and indicate which player is next (turn_a). Clearly, this is not sufficient to represent the real state of a game: the knowledge of each player about their cards evolves during a game, and in order to reason about how best to play, it is necessary to have a model of this knowledge – we use *epistemic logic* for this.

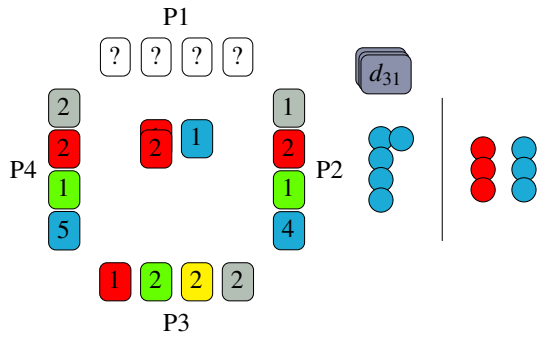


Figure 1: Presentation of a game situation as seen by player 1

2.2 Probabilistic epistemic logic

We fix a finite set A of *agents* and a countable set PS of propositional symbols. We denote as WS the countable subset of PS containing the symbols used to describe possible worlds; for simplicity we consider it fixed, even though it depends on the application (for example, in the context of Hanabi, we consider that WS contains all and only the necessary symbols of the form card^{ap}_{cvi}, card^{z}_{cvi}, R_i, B_i and turn_a).

First, we define probabilistic Kripke structures (also called probabilistic Kripke/epistemic models; in the following we simply call them "Kripke structures" for short), which allows one to model the epistemic and probabilistic knowledge of agents.

Definition 1 ((probabilistic) Kripke structure)**.** *A* (probabilistic) Kripke structure *is a tuple* $\mathscr{M} = \langle W, R, \mu, V \rangle$ *such that:*

- W *is a nonempty finite set of* worlds*,*
- R *associates to each agent* $a \in A$ *an* accessibility relation $R_a \subseteq W \times W$*,*
- μ *associates to each agent* $a \in A$ *and each world* $w \in W$ *a probability function* $\mu_a(w)$ *over all worlds, i.e., a function* $W \to [0,1]$ *such that* $\sum_{w' \in W} \mu_a(w)(w') = 1$*, and*
- V *is a* valuation function $V : W \to 2^{WS}$*, indicating the set of propositions that are true in each world.*

Note that there are more general definitions of probabilistic Kripke structures, using for example σ-algebras [Fagin et Halpern, 1994]; in this paper we use the simplest case of probability functions, but our approach would not be hard

to generalize to richer settings. Intuitively, the accessibility relation of agent a links worlds that a considers as *undistinguishable*, and $\mu_a(w)(w')$ indicates the probability a assigns to being in world w' when it is actually in world w.

Example 2. *Let s and r be two propositions standing for "sun" and "rain"; assume that at any time at least one of the two propositions must be true. Figure 2 (left) displays an example of a very simple Kripke structure, in which there are two agents, whose accessibility relations are consistent with the probability functions. The probabilities associated with the arcs show that the black agent deems that there is a greater chance of rain. World w_1 is marked with a double circle to indicate that it is the real world. Black arcs show probabilities for agent a only, and orange arcs for agents a and b.*

In Definition 1, we choose to follow the original definition of van Benthem *et al.* [2009], which was built upon that of Fagin et Halpern [1994], in which no relationship is enforced between accessibility relations and probability functions. Having both an accessibility relation and probability functions may seem redundant, but it has two advantages: (i) it allows one to model an unquantified uncertainty between possible worlds, which is not the same as assigning them a uniform probability, and (ii) it allows one to distinguish between having probability 0 to be true and being truly impossible. It is possible to impose restrictions on this relationship (see Fagin et Halpern [1994] for an in-depth discussion about these aspects), but different contexts could require different restrictions; being as general as possible, our framework is agnostic to these considerations.

Definition 3 (Probabilistic epistemic language $\mathscr{L}_{PEL}$)**.** *The language $\mathscr{L}_{PEL}$ of probabilistic epistemic logic is defined by the following Backus-Naur form :*

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathrm{K}_a\phi \mid \alpha_1 \Pr_a(\phi_1) + \cdots + \alpha_k \Pr_a(\phi_k) \geq \beta$$

where $p \in WS$, $a \in A$, $\alpha_1, \ldots, \alpha_k, \beta$ are rational numbers, and $k \geq 1$.

We also use parentheses in formulas for disambiguation, and we recall the usual abbreviations: $p \vee q$ (or) is $\neg(\neg p \wedge q)$, $p \rightarrow q$ (implication) is $\neg p \vee q$ and $p \leftrightarrow q$ (equivalence) is $(p \rightarrow q) \wedge (q \rightarrow p)$. This language allows higher-order formulas so that we can write : $\mathrm{K}_i(\Pr_j(\phi) \geq \beta)$ or $\Pr_j(\mathrm{K}_i\phi) \geq \beta$. For example, in the first formula, agent i knows that agent j estimates the probability that ϕ holds to be greater than or equal to β. Next, we explain how one decides whether a formula in $\mathscr{L}_{PEL}$ holds in a given state of epistemic and probabilistic knowledge:

Definition 4 (semantics of PEL)**.** *Let $\mathscr{M}$ be a Kripke structure and w a world of $\mathscr{M}$. We define the semantics of PEL inductively as follows, where $p \in WS$, $\phi, \psi, \phi_1, \ldots \phi_k \in \mathscr{L}_{PEL}$, and $a \in A$:*

$\mathscr{M}, w \models p$ *iff* $V(w)(p) = \top$, *with* $p \in WS$
$\mathscr{M}, w \models \neg\phi$ *iff* $\mathscr{M}, w \not\models \phi$
$\mathscr{M}, w \models \phi \wedge \psi$ *iff* $\mathscr{M}, w \models \phi$ *and* $\mathscr{M}, w \models \psi$
$\mathscr{M}, w \models \mathrm{K}_a\phi$ *iff* $\mathscr{M}, w' \models \phi$ *for all* w' *s.t.* $(w, w') \in R_a$
and $\mathscr{M}, w \models \alpha_1 \Pr_a(\phi_1) + \cdots + \alpha_k \Pr_a(\phi_k) \geq \beta$
iff $\sum_{i=1}^{k} \alpha_i \mu_a(\mathscr{M}, w)(\phi_i) \geq \beta$, *with* $\mu_a(\mathscr{M}, w)(\phi_i) := \sum_{w' :\, \mathscr{M}, w' \models \phi_i} \mu_a(w)(w')$.

In our weather example, formulas $Pr_b(r) \geq 0.5$ and $K_b(Pr_a(s \vee r) \geq 0.8)$ are true. Now, Kripke structures, probabilistic or not, only represent a *static* state of knowledge. In order to take changes into account, e.g. due to player actions, they have to be coupled with other structures called *update models*.

2.3 Updating knowledge

Note that we use $\mathscr{L}_{WS}$ to refer to the propositional language.

Definition 5 ((probabilistic) update model)**.** *An* (probabilistic) update model *is a tuple $\mathscr{E} = \langle E, R^E, pre, post, \mu^E \rangle$, where*

- *E is nonempty set of* events*,*
- *R^E associates to each agent $a \in A$ an* accessibility relation *$R^E_a \subseteq E \times E$,*
- *pre is a* precondition *function $pre : E \rightarrow \mathscr{L}_{EL}$,*
- *post is a* postcondition *function $post : E \times WS \rightarrow \mathscr{L}_{WS}$ (with $\mathscr{L}_{WS}$ the language of propositional formulas),*
- *μ^E associates to each agent $a \in A$ and each event $e \in E$ a probability function $\mu_a(e)$ over all events.*

Example 6. *The update model in Figure 2 (middle) (in which we suppose that probability functions are the same for all agents) contains an event e_0 with precondition $s \wedge r$ which assigns true to the variable b ("rainbow"). If the precondition is true, there is a 60% chance that there is a rainbow. Nevertheless, this event is confused by agents with the event e_1, with precondition r. Would this second event occur, there would be an 80% chance that it be considered to occur.*

Note that using this definition, probabilistic update models are actually very close to the classical update models of DEL [van Benthem *et al.*, 2006b] – the only difference being the added probability functions. This makes them different from probabilistic update models as defined by van Benthem *et al.* [2009], for two reasons. First, they feature *postconditions*, which allow them to have *ontic effects*, i.e., to change the state of the world and not only the knowledge state of agents; this is indeed crucial if we want to use PDEL to reason about games such as Hanabi. By fixing $post(e, x) = x$ for all $e \in E$ et all $x \in WS$, we get a purely epistemic update model, with no ontic effect. The other difference is that in the update models of van Benthem *et al.* [2009], preconditions are not directly tied to

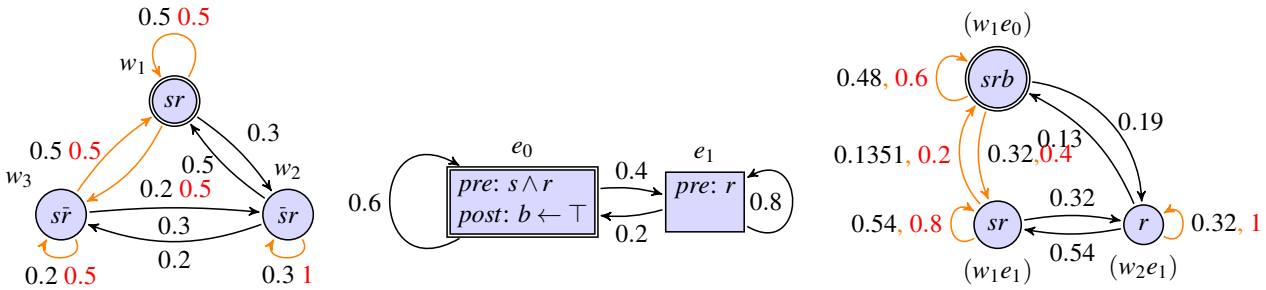


Figure 2: Examples of a probabilistic Kripke structure (left) and of an update model (middle), and their product update (right)

events. They are defined as a set Φ of pairwise inconsistent sentences, together with a probability distribution $pre(\phi, e)$ over E for each sentence $\phi \in \Phi$ and each event $e \in E$. As the authors argue, this allows one to independantly consider *observation probabilities*, given by μ^E – quantifying the distinguishability of events – and *occurrence probabilities*, given by Φ and *pre* – indicating the probability with which each event can occur in each situation. Even though these "full update models" are more flexible, they are actually not more expressive than our simpler version (without postconditions). As remarked by the authors in a prepublished extended version of their work [van Benthem *et al.*, 2006a], any full update model has an equivalent so-called "observation model", which corresponds to our definition (again, ignoring ontic effects). Moreover, our choice of a conceptually simpler definition is harmless in practice, because transforming a full update model into an observation model can be done in quadratic time. However, it is important to remark that the converse is not true – it can be shown that the following observation model has no polynomial representation as a full update model: a clique of $|WS|$ events, with uniform probabilities, each event having a distinct variable $x \in WS$ as precondition. The problem is that full update models require pairwise inconsistent preconditions, so we need one for each valuation in 2^{WS}. This is a strong motivation for our choice of limiting update models to "observation models" because, although update models of this form do not seem very useful semantically, they become essential as soon as we add postconditions to them – such combinatorial, probabilistically uniform updates are frequent in the context of card games.

Now that we have update models to represent events that can occur in the real world (either by a voluntary action of an agent, or not), and how these events are perceived by the agents, we will explain how we can use them. The mechanism used in PDEL to compute the new knowledge state of agents after an event took place is called *product update*; it is more or less a simple Cartesian product of the Kripke structure representing the previous knowledge state and of the update model representing the event.

Definition 7 (probabilistic product update)**.** *Let $\mathcal{M}$ be a Kripke structure and $\mathcal{E}$ be an update model.*

The product update *of $\mathcal{M}$ by $\mathcal{E}$ is the Kripke structure $\mathcal{M} \otimes \mathcal{E} = \langle W^\otimes, R^\otimes, \mu^\otimes, V^\otimes \rangle$ defined as follows:*

- $W^\otimes = \{(w,e) \in W \times E \mid \mathcal{M}, w \models pre(e)\}$
- $(w,e)R_a^\otimes(w',e')$ *iff* $(w,w') \in R_a$ *and* $(e,e') \in R_a^E$
- $\mu_a^\otimes((w,e))((w',e')) = \frac{\mu_a(w)(w') \cdot \mu_a^E(e)(e')}{\sum_{(w'',e'') \in W^\otimes} \mu_a(w)(w'') \cdot \mu_a^E(e)(e'')}$ *if the denominator is nonzero, and* 0 *otherwise*
- $V^\otimes((w,e)) = \{p \in WS \mid \mathcal{M}, w \models post(e,p)\}$

Example 8. *Fig. 2 (right) shows the product update of the Kripke structure and update model. We can see that after the application of the event, agent a (in black) knows that it is raining: $K_a r$, but also that considers that there may be a rainbow with probability more than 45% : $Pr_a b \geq 0.45$.*

Thanks to the product update, it is then possible to do model checking by using a new operator that means "after the application of such event, will this formula be true?" This is how the logic becomes "dynamic".

Definition 9 (PDEL)**.** *The* probabilistic dynamic epistemic language $\mathcal{L}_{PDEL}$ *is defined by the BNF of $\mathcal{L}_{PEL}$ (Def. 3) augmented with the following production rule:*

$$\phi ::= [\mathcal{E}, e]\phi$$

where $\mathcal{E}$ is a probabilistic update model and e an event. We denote by $\mathcal{L}^0_{PDEL}$ the language in which all update models have propositional preconditions. The semantics of this additional operator is defined as follows: $\mathcal{M}, w \models [\mathcal{E}, e]\phi$ iff

$$\mathcal{M}, w \models pre(e) \implies \mathcal{M} \otimes \mathcal{E}, (w,e) \models \phi$$

We are interested in the *model checking* problem, which consists in deciding, given a pointed Kripke structure $(\mathcal{M}, w)$ and a formula ϕ in $\mathcal{L}_{PEL}$ or $\mathcal{L}_{PDEL}$, whether $\mathcal{M}, w \models \phi$.

3 Symbolic representation

Because of the combinatorial nature of Kripke structures, the size of such representations quickly becomes huge in practical examples, even moderately realistic. To avoid

this problem, an idea is to use a "symbolic" representation of Kripke structures, in which redundancies are factored out, so that knowledge states remain scalable, while still allowing for reasonably efficient model checking. The idea of symbolic model checking [McMillan, 1993] has recently been applied to dynamic epistemic logic, the symbolic representation of Kripke structures being either *accessibility programs* [Charrier et Schwarzentruber, 2017; Charrier *et al.*, 2019] or Binary Decision Diagrams (BDDs) [van Benthem *et al.*, 2017; Gattinger, 2018], BDDs being a well-known efficient representation of Boolean formulas [Bryant, 1986].

We now show how these concepts apply to probabilistic dynamic epistemic logic; contrary to Shirazi et Amir [2008], who also represent probabilistic Kripke structures in a factored way (using Bayesian networks), we are not only interested in static structures. In order to represent probabilities, we have to go beyond Boolean formulas, and use pseudo-Boolean functions. Given $X = \{x_1, \dots, x_n\} \subseteq PS$ a finite set of propositional symbols, we call *pseudo-Boolean function (PBF) on variables X* a total function of the kind $f\colon 2^X \to \mathbb{R}$. There are several ways to represent pseudo-Boolean functions, notably generalizations of BDDs; let us mention Algebraic Decision Diagrams (ADDs) [Bahar *et al.*, 1997], Semiring-Labelled Decision Diagrams (SLDDs) [Wilson, 2005; Fargier *et al.*, 2013], Affine Algebraic Decision Diagrams (AADDs) [Sanner et McAllester, 2005], and Probabilistic Sentential Decision Diagrams (PSDDs) [Kisa *et al.*, 2014]. These languages are of varying *succinctness* (i.e., some are able to represent PBFs more compactly than others) and do not have the same efficiency for operations (such as summing PBFs or "forgetting" variables). There is a tradeoff to be found, depending on the application; systematically studying and quantifying such tradeoffs is the goal of the literature about the *knowledge compilation map* [Darwiche et Marquis, 2002; Fargier *et al.*, 2014]. In this paper, we remain as general as possible and only talk about PBFs; it should be implicitly understood that they are represented in some efficient language, such as ADD, the language we used in our experiments.

Before going on, we need to introduce some conventions and notations. We consider that Boolean functions, i.e., functions of the form $f\colon 2^X \to \mathbb{B}$, are particular pseudo-Boolean functions, and for simplicity we often identify propositional formulas ϕ over X with the Boolean function they represent; i.e., we see ϕ as the Boolean function $v \in 2^X \mapsto \begin{cases} 1 & \text{if } v \models \phi, \\ 0 & \textit{otherwise} \end{cases}$.

For a PBF f on $X \subseteq PS$ and a real number β, we denote $\mathrm{Cut}_{\geq\beta}(f)$ the Boolean function associating a valuation $v \in 2^X$ to 1 if $f(v) \geq \beta$, and to 0 otherwise.

We call support of f, written $\mathrm{supp}(f)$, the set $Y \subseteq 2^X$ where f is nonzero : $\mathrm{supp}(f) = \{x \in 2^X \mid f(x) \neq 0\}$. Note that if ϕ is a propositional formula over X, $\mathrm{supp}(\phi)$ is the model set of ϕ.

Finally, let $\odot\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be an associative and commutative operation with a neutral element (such as addition, multiplication, min, max, or the Boolean connectives $\vee$ and $\wedge$), let $X = (Y \uplus Z) \subseteq PS$, and let f be a PBF on X. The $\odot$-*marginalization* of Y in f, denoted $\mathrm{Marg}_{\odot,Y}(f)$, is the function $g\colon 2^Z \to \mathbb{R}$ defined by $g(v) = \bigodot_{v' \in 2^X\colon v = v'_Z} f(v')$. Note that if f is a Boolean function, $\vee$-marginalization (resp. $\wedge$-marginalization) of Y corresponds to *existentially (resp. universally) forgetting* the variables in Y. We write $\mathrm{Forget}^{\exists}_Y(f) = \mathrm{Marg}_{\vee,Y}(f)$ and $\mathrm{Forget}^{\forall}_Y(f) = \mathrm{Marg}_{\wedge,Y}(f)$. Also we use for $X \subseteq PS$, f a boolean function and $v \in 2^X$, $f_{|v} = \mathrm{Forget}^{\exists}_X(f \wedge v)$

In this article, we use PBF everywhere. In fact, for simplification, if PBF is Boolean Function we will write it as Boolean formula. Also, a valuation, list of term, or a list or valuation can be considered as a Boolean formula. We can notice that, for v a valuation and f a boolean formula, notations $v \in \mathrm{supp}(f)$, $v \models f$ and $f(v) = 1$ are equivalent.

3.1 Static structure

Let us now show how to represent a Kripke structure with a PBF. The basic idea is to identify worlds with their valuations, so that propositional formulas over *WS* directly represent sets of worlds. Obviously, it is not possible in the general case: two distinct worlds in a Kripke structure can have the same valuation. The symbolic representation only works for structures whose valuation function is injective:

Definition 10 (V-injectivity)**.** *A Kripke structure* $\mathscr{M} = \langle W, R, \mu, V\rangle$ *is called* V-injective *if* $\forall w, w' \in W\colon w \neq w' \to V(w) \neq V(w')$.

While V-injective Kripke structures are considerably restricted in terms of expressiveness (e.g., the satisfiable S5 formula $\mathrm{K}_a 1 p \wedge \neg \mathrm{K}_a 2(\mathrm{K}_a 1 p \vee \mathrm{K}_a 2 \neg p)$ has no V-injective S5 model), the setting still applies to a lot of games – notably those in which there are no private announcements or secret actions, so that all uncertainty is reducible to uncertainty about the "physical state" of the game. This is the case for Hanabi: for example, even though Alice does not know what the other players know about her cards, she can enumerate the possible game states (this boils down to enumerating her possible hands), and for each one, assuming it is the actual game state, she knows exactly what other players know. This would not be the case if there were private announcements (e.g., one player gives an information to another one without being heard by the others) or secret actions (e.g., two players switch one of their cards without others noticing). Moreover, for cases requiring more expressiveness, it is always possible to distinguish between worlds by labelling them with fresh symbols. Finally, note that this restriction is not unusual; it also holds for existing approaches to symbolic model checking for epistemic logic.

In order to represent relations over worlds, we will use the classical trick of *duplicating our vocabulary* [Gattinger, 2018]: we introduce a fresh set of symbols $WS' \subseteq PS \setminus WS$,

in one-to-one correspondence with those in WS (we fix some bijection $\mathrm{succ}\colon WS \to WS'$, denoting $\mathrm{pred} = \mathrm{succ}^{-1}$), and see each valuation $v \in 2^{WS \cup WS'}$ as a pair of valuations (v_1, v_2), where $v_1 = v \cap WS$ and $v_2 = \{\mathrm{pred}(x) \mid x \in v \cap WS'\}$. For $v \in 2^{WS}$, we write $\mathrm{succ}(v) := \{\mathrm{succ}(x) \mid x \in v\}$, and for f a PBF on WS, we write $\mathrm{succ}(f)$ for the PBF $v \in 2^{WS} \mapsto \mathrm{succ}(v)$; we introduce similar overloadings for pred.
We can define symbolic Kripke structure and the function to transform a explicit Kripke structure into symbolic one.

Definition 11 (Symbolic Kripke structure)**.** *A* symbolic Kripke structure *is a tuple* $F = \langle W^r, R^r, \mu^r \rangle$ *such that*

- W^r *is a Boolean function on WS, called the* law of worlds*;*
- R^r *associates to each agent* $a \in A$ *a Boolean function* R^r_a *on* $WS \cup WS'$*, called the* law of knowledge*;*
- μ^r *associates to each agent* $a \in A$ *a PBF* μ^r_a *on* $WS \cup WS'$*, called the* law of probabilities*.*

This representation allows any Kripke structure to be modelled, whether it is S5, KD45, or another combination of epistemic logic axioms.

Definition 12. *Each explicit Kripke structure* $\mathscr{M} = \langle W, R, \mu, V \rangle$ *corresponds to a symbolic Kripke structure* $\mathrm{symb}(\mathscr{M}) = \langle W^r, R^r_a, \mu^r_a \rangle$ *defined as follows:*

- $W^r := \bigvee_{w \in W} V(w)$*;*
- $R^r_a := \bigvee_{(w,w') \in R_a} V(w) \wedge \mathrm{succ}(V(w'))$ *;*
- $\mu^r_a = \sum_{(w,w') \in W^2} \mu_a(w)(w') \times V(w) \wedge \mathrm{succ}(V(w'))$.

Here, it is important to see that the "sums" are disjoint if $\mathscr{M}$ is V-inj. Indeed, whether it is for the or logical on R^r or for the sum on μ^r, if $\mathscr{M}$ is V-inj, the relation $V(w) \wedge \mathrm{succ}(V(w'))$ is unique. If it is not, there is a loss of information by collasping on the structure.

Definition 13 (Symbolic Kripke into explicit : $\mathrm{symb}^{-1}(F)$)**.** *Each symbolic Kripke structure* $F = \langle W^r, R^r, \mu^r \rangle$ *corresponds to the unique Kripke structure* $\mathrm{symb}^{-1}(F) = \langle W, R, \mu, V \rangle$ *defined as follows:*

- $W := \mathrm{supp}(W^r)$*;*
- $V := (v \in W \mapsto v)$*;*
- $R_a := \{(v, v') \in W^2 \mid v \cup \mathrm{succ}(v') \in \mathrm{supp}(R^r_a)\}$*;*
- $\mu_a := (v \in W \mapsto (v' \in W \mapsto \mu^r_a(v \cup \mathrm{succ}(v')) \times \mathrm{norm}_a(v)))$*, where* $\mathrm{norm}_a(v) = 1/\sum_{v' \in W} \mu^r_a(v \cup \mathrm{succ}(v'))$ *if the denominator is nonzero, and* 0 *otherwise.*

Lemma 14. *If* $\mathscr{M}$ *is V-injective,* $\mathscr{M} \simeq \mathrm{symb}^{-1}(\mathrm{symb}(\mathscr{M}))$.

Proof. Let's suppose $\mathscr{M}_l = \langle W_l, R_l, \mu_l, V_l \rangle$ V-inj and $\mathscr{M}_r = \mathrm{symb}^{-1}(\mathrm{symb}(\mathscr{M}_a)) = \langle W_r, R_r, \mu_r, V_r \rangle$. Let f be $W_l \to W_r$ such that $f(w_l) = V(w_l)$.
$W_r = \mathrm{supp}(\bigvee_{w \in W_l} V(w))$ and V_l is injective. So for all $w_l \in W_l$ there is one only $w_r \in W_r$ such that $f(w_l) = V(w_l) = w_r$.
Let $w_r, {w'}_r \in W_r^2$ and $w_l, {w'}_l \in W_l^2$ s.t. $w_r = f^{-1}(w_l) = V(w_l)$, ${w'}_r = f^{-1}({w'}_l) = V({w'}_l)$,

$(w_r, {w'}_r) \in R_r$
$\iff w_r \cup \mathrm{succ}({w'}_r) \models \mathrm{supp}(\bigvee_{(w,w') \in R_l} V(w) \wedge \mathrm{succ}(V(w')))$
$\iff V(w_l) \cup \mathrm{succ}(V({w'}_l)) \models \mathrm{supp}(\bigvee_{(w,w') \in R_l} V(w) \wedge \mathrm{succ}(V(w')))$
$\iff (w_l, {w'}_l) \in R_l$

$\mu_r(w_r, {w'}_r)$
- $(\sum_{w_l, {w'}_l \in W_l^2} \mu_l(w_l)({w'}_l) \times V(w_l) \times \mathrm{succ}(V({w'}_l)))(V(w_l) \cup \mathrm{succ}(V({w'}_l)) \times \mathit{norm}_a({w'}_l)$ with $\mathit{norm}_a({w'}_l) = 1/\sum_{w_l \in W_l} \sum_{w,w' \in W_l^2} \mu_l(w)(w') \times V(w) \times \mathrm{succ}(V(w'))$
- $(\sum_{w \in f^{-1}(w_r), w' \in f^{-1}({w'}_r)} \mu_l(w_l)({w'}_l) \times V(w_l) \times \mathrm{succ}(V({w'}_l)))(V(w_l) \cup \mathrm{succ}(V({w'}_l)) \times \mathit{norm}_a({w'}_l)$ with V_l injective
- $(\mu_l(w_l)({w'}_l) \times V(w_l) \times \mathrm{succ}(V({w'}_l)))(V(w_l) \cup \mathrm{succ}(V({w'}_l)) \times \mathit{norm}_a({w'}_l)$
- $\mu_l(w_l)({w'}_l) \times \mathit{norm}_a({w'}_l)$
  We have by definition of explicit Kripke structure : $\forall w \in W_l, \sum_{w'} \mu_l(w)(w') = 1$ and with distinct sums, we have $\mathit{norm}_a(w_r) = 1$.
- $\mu_l(w_l)({w'}_l)$

□

Though a symbolic Kripke structure represents a unique (V-injective) Kripke structure, the converse is not true. When building a symbolic representation of a V-injective Kripke structure, the law of worlds is fixed, but there are two degrees of freedom: (i) the law of probabilities does not have to be normalized, and (ii) the laws of knowledge and probabilities can assign values to any assignment in $2^{WS \cup WS'}$, even when it does not correspond to any pair of worlds.
Moreover, these "symbolic" Kripke structures are very generic, since, as we already mentioned, there is no constraint as to how the Boolean and pseudo-Boolean functions defining the three "laws" are represented. The choice of concrete representations can thus depend on the intended tradeoff between spatial and temporal efficiency for various applications. Anyway, it should be clear that symbolic Kripke structure can yield exponential space savings: a trivial example is $F = \langle \top, \top, 1 \rangle$ (where 1 here is the constant PBF), which represents a Kripke structure with $2^{|WS|}$

distinct worlds.

3.2 Model checking on static structures

In order to decide whether (the Kripke structure represented by) a given symbolic Kripke structure is a model of an $\mathscr{L}_{PEL}$ formula, we can build a Boolean function on *WS* representing the set of all worlds of the structure in which the formula holds. This can be done using dynamic programming thanks to the following inductive definition.

Definition 15 (Boolean translation of formula)**.** *Let $F = \langle W^r, R^r, \mu^r \rangle$ be a symbolic Kripke structure and ϕ be a formula in $\mathscr{L}_{PEL}$. The* Boolean translation of ϕ in *F, denoted $||\phi||_F$, is the Boolean function defined inductively as follows:*

$$||p||_F := p$$
$$||\neg\phi||_F := 1 - ||\phi||_{\text{symb}(\mathscr{M})}$$
$$||\phi \wedge \psi||_F := ||\phi||_{\text{symb}(\mathscr{M})} \times ||\psi||_{\text{symb}(\mathscr{M})}$$
$$||\mathrm{K}_a\phi||_F := \text{Forget}^{\forall}_{WS'}(\text{succ}(W^r) \wedge R^r_a \rightarrow \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}))$$
$$||\alpha_1 \Pr_a(\phi_1) + \cdots + \alpha_k \Pr_a(\phi_k) \geq \beta||_F := \text{Cut}_{\geq\beta}(\textstyle\sum_{i=1}^k \alpha_i \times ||\Pr_a(\phi_i)||_F) \text{ where :}$$
$$||\Pr_a(\phi)||_F = \text{Marg}_{\Sigma,WS'}(\text{succ}(W^r) \times \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}) \times norm(\mu^r_a))$$
$$norm(\mu^r_a) = \frac{\mu^r_a}{\text{Marg}_{\Sigma,WS'}(\mu^r_a)}$$

We said that the law of probabilities does not have to be normalized, but for modelcheck a $\Pr_a$ formula, we need to find normalized PBFs to be able to compare the PBF values with the β value of the formula. In fact, we just need before applying a model checking to normalize the F structure in this way: $norm(F) = \langle W^r, R^r, norm(\mu^r) \rangle$

Remark that using $||\Pr_a(\phi)||_F$ is a bit of an notational abuse, since $\Pr_a(\phi)$ is not a formula, and indeed $||\Pr_a(\phi)||_F$ is not a Boolean function. Now, it should be clear that the complexity of building the Boolean translation of a formula depends on the concrete representations used. Nonetheless, note that all operations need – sum, product, renamings, cut, marginalization/forgetting – can be considered as elementary operations on PBF representations; they are notably used by Fargier *et al.* [2014] to compare the efficiency of several languages of the decision diagram family.

The following result can be proved by induction :

Proposition 16 (symbolic model checking on PEL)**.** *Let $\mathscr{M} = \langle W, R, \mu, V \rangle$ be a V-inj Kripke structure. For any formula ϕ in $\mathscr{L}_{PEL}$ any $w \in W$, it holds that $\mathscr{M}, w \models \phi \iff ||\phi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$.*

Proof. The proof is by induction on ϕ. Cases of atomic propositions, negation and conjunction are immediate.

$\mathscr{M}, w \models \neg\phi$
$\iff \mathscr{M}, w \not\models \phi$
$\iff ||\phi||_{\text{symb}(\mathscr{M})}(V(w)) \neq 1$ (hyp)
$\iff 1 - ||\phi||_{\text{symb}(\mathscr{M})}(V(w)) \neq 0$
$\iff ||\neg\phi||_{\text{symb}(\mathscr{M})} = 1$ (Domain of $||\phi||_{\text{symb}(\mathscr{M})}(V(w))$ is boolean : [0,1])

$\mathscr{M}, w \models \phi \wedge \psi$
$\iff \mathscr{M}, w \models \phi \wedge \mathscr{M}, w \models \psi$
$\iff ||\phi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$ and $||\psi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$ (hyp)
$\iff ||\phi||_{\text{symb}(\mathscr{M})}(V(w)) \times ||\psi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$ (Domain of $||\phi||_{\text{symb}(\mathscr{M})}(V(w))$ and $||\psi||_{\text{symb}(\mathscr{M})}(V(w))$ are boolean : [0,1])
$\iff (||\phi||_{\text{symb}(\mathscr{M})} \times ||\psi||_{\text{symb}(\mathscr{M})})(V(w)) = 1$
$\iff ||\phi \wedge \psi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$

$\mathscr{M}, w \models \mathrm{K}_a\phi$
$\iff \forall w' \in W$ s.t. $(w, w') \in R_a : \mathscr{M}, w' \models \phi$
$\iff \forall w' \in W$ s.t. $V(w) \cup \text{succ}(V(w')) \in \text{supp}(R^r_a) : ||\phi||_{\text{symb}(\mathscr{M})}(V(w')) = 1$ (definition 12 + hyp)
$\iff \forall w' \in W : R^r_a(V(w) \cup \text{succ}(V(w')) = 1 \rightarrow ||\phi||_{\text{symb}(\mathscr{M})}(V(w')) = 1$
$\iff \forall v' \in 2^{WS'} : v' \models \text{succ}(W^r) \rightarrow (R^r_a(V(w) \cup v') = 1 \rightarrow \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}(v')) = 1))$
$\iff \forall v' \in 2^{WS'} : \neg v' \models \text{succ}(W^r) \vee \neg V(w) \cup v' \models R^r_a \vee \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}(v')) = 1$
$\iff \forall v' \in 2^{WS'} : (V(w) \cup v' \models \text{succ}(W^r) \wedge R^r_a) \rightarrow \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}(v')) = 1$
$\iff \text{Forget}^{\forall}_{WS'}(\text{succ}(W^r) \wedge R^r_a \rightarrow \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}))(V(w)) = 1$
$\iff ||\mathrm{K}_a\phi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$

We suppose that μ^r_a is normalized.
$\mathscr{M}, w \models \alpha_1 \Pr_a(\phi_1) + \cdots + \alpha_k \Pr_a(\phi_k) \geq \beta$
$\iff \sum_{i=1}^k \alpha_i \mu_a(\mathscr{M}, w)(\phi_i) \geq \beta$ with $\mu_a(\mathscr{M}, w)(\phi_i) := \sum_{w' :\, \mathscr{M}, w' \models \phi_i} \mu_a(w)(w')$.
$\iff \sum_{i=1}^k \alpha_i \sum_{w' :\, \mathscr{M}, w' \models \phi_i} \mu_a(w)(w') \geq \beta$ (rewrite)
$\iff \sum_{i=1}^k \alpha_i \sum_{w' :\, \mathscr{M}, w' \models \phi_i} \mu^r_a(V(w) \cup \text{succ}(V(w'))) \geq \beta$ (with definition 12)
$\iff \sum_{i=1}^k \alpha_i \sum_{v \in 2^{WS} :\, \text{succ}(W^r) \times \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}(v)) = 1} \mu^r_a(V(w) \cup \text{succ}(v)) \geq \beta$ (with the induction hypothesis that $\mathscr{M}, w \models \phi \iff ||\phi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$ with $v = V(w')$ and the addition of W^r in order to respect $w' \in W$ which become $v \models W^r$)
$\iff \sum_{i=1}^k \alpha_i \sum_{v \in 2^{WS}} (\mu^r_a(V(w) \cup \text{succ}(v)) \times \text{succ}(W^r) \times \text{succ}(||\phi||_{\text{symb}}(\mathscr{M})))(V(w) \cup \text{succ}(v)) \geq \beta$ ($\text{succ}(W^r) \times \text{succ}(||\,\text{symb}(\phi)||_(v))$ are purely boolean, and that's why we remove the $= 1$. It works as a filter under $\sum$ and also in multiplication of PBF.)
$\iff \sum_{i=1}^k \alpha_i \text{Marg}_{\Sigma,WS}((\mu^r_a(V(w) \cup \text{succ}(v)) \times$

$\text{succ}(W^r) \times \text{succ}(||\phi||_{\text{symb}(\mathscr{M})}))(V(w)) \geq \beta$ (definition of Marginalisation)

$\iff \sum_{i=1}^{k} \alpha_i ||\Pr_a(\phi_i)||_{\text{symb}(\mathscr{M})}(V(w)) \geq \beta$ (definition 15)

$\iff \text{Cut}_{\geq\beta}(\sum_{i=1}^{k} \alpha_i \times ||\Pr_a(\phi_i)||_{\text{symb}(\mathscr{M})})(V(w))$ (definition 15)

$\iff ||\alpha_1 \Pr_a(\phi_1) + \cdots + \alpha_k \Pr_a(\phi_k) \geq \beta||_{\text{symb}(\mathscr{M})}(V(w))$

□

3.3 Symbolic updates

Now that we have defined a symbolic Kripke structure and showed how PEL model checking can be done on it, we will show how these structures can be updated through symbolic update models, which finally yields a model checking algorithm for PDEL (restricted to propositional preconditions).

Symbolic representation of update models. We want to use the same principle as for Kripke structures to represent the accessibility relation and the probability functions. But an event is not labeled with a valuation, but with precondition and postcondition functions. If we consider only update models with propositional formulas as preconditions (which are a lot less expressive, but still sufficient for most real-life games, in which the applicability of actions only depends on the objective state of the game and not on what players know about it), it is possible to view precondition and postcondition functions as valuations, by representing them into propositional *action theories* from classical planning. An action theory is a propositional formula θ on $WS \cup WS+$ (where $WS+$ is, once again, a set of fresh propositional symbols in one-to-one correspondance with those in WS via some bijection after, of which we denote before the reciprocal, and that we overload in the same way as succ and pred) that describes the effects of an action (or rather, here, an event) in the following way: given two valuations $v, v' \in 2^{WS}$, world state v' is a possible outcome of applying the action in world state v if and only if $v \cup \text{after}(v') \models \theta$.

Definition 17 (propositional event theory)**.** *Let $\mathscr{E} = \langle E, R^E, \mu^E, pre, post\rangle$ be an update model with propositional preconditions. The* propositional event theory *of event $e \in E$ is the Boolean function θ_e defined by the propositional formula $pre(e) \wedge \theta_e^{post}$, where $\theta_e^{post} = \bigwedge_{x \in WS} (post(e)(x) \leftrightarrow \text{after}(x))$*

We can break down the θ_e definition of a e event with the following lemma, with $post_e(v) := \{x \in WS \mid v \models post(e)(x)\}$.

Lemma 18. *For $v \in 2^{WS}$, ${\theta_e^{post}}_{|v} = \text{Forget}_{WS}^{\exists}(\theta_e^{post} \wedge v) = \text{after}(post_e(v))$*

In other words, as θ_e permit to associate to each $v \in 2^{WS}$ a unique $post_e(v)$ and we can see that $v \wedge \theta_e^{post} = v \wedge \text{after}(post_e(v))$

We say that an event e in an update model is *atomic* if $|\text{supp}(pre(e))| = 1$, i.e., the event only applies to a single valuation in 2^{WS}: it only describes the transition from one complete valuation to another. We call *event-atomic* an update model of which all events are atomic. Note that the propositional event theory of atomic events can be represented by a complete term over $WS \cup WS+$, or equivalently, by a single valuation in $2^{WS \cup WS+}$. Thus, an event-atomic update model can simply be represented by a (static) Kripke structure on $WS \cup WS+$ (of which the valuation of each world can be interpreted as an atomic event theory) – and to any Kripke structure on $WS \cup WS+$ corresponds a unique event-atomic update model. Moreover, it can be shown (it is a consequence of our results) that any update model (with propositional preconditions) can be represented as an equivalent event-atomic update model, by separating each event into as many atomic events as needed, duplicating all accessibility arcs and probabilities (modulo a final normalization step).

We can now show how an update model can be translated into a symbolic representation, and finally show how to compute the product update on symbolic structures.

Definition 19 (symbolic update model)**.** *Let $\mathscr{E} = \langle E, R^E, pre, post, \mu^E\rangle$ be a update model with propositional preconditions; its symbolic representation is the symbolic Kripke structure* $\text{symb}(\mathscr{E}) = \langle W^r, R^r, \mu^r\rangle$ *on $WS \cup WS+$ defined as follows:*

- $W^r := \bigvee_{e \in E} \theta_e$;
- $R_a^r := \bigvee_{(e,e') \in R^E} \theta_e \wedge \text{succ}(\theta_{e'})$;
- $\mu_a^r := \sum_{(e,e') \in E^2} \theta_e \times \text{succ}(\theta_{e'}) \times \mu_a^E(e)(e')$.

Thanks to the symbolic representation definitions of the symbolic Kripke structures and the symbolic update models, we can now define the symbolic product update.

By the way, to keep model checking valid, no information must be lost by collapsing. So we need to keep in mind that $\mathscr{M} \otimes \mathscr{E}$ must be V-injective.

So, we need to describe some sufficient conditions for everything to go well. We define a new constraint on the event structure: the Transition-injectivity.

Definition 20 ($\mathscr{E}$ Transition-injective)**.** *$\mathscr{E}$ is said to be transition-injective (T-inj) wrt ϕ iff $\forall e, e' \in E^2$, $\forall v, v' \in 2^{WS}$ s.t. $v \models pre(e) \wedge \phi$ and $v' \models pre(e') \wedge \phi$, $post_e(v) = post_{e'}(v') \implies v = v', e = e'$.*

In other words we can describe the Transition-injectivity as follow : $\forall v \neq v' \in 2^{WS}$ or $e \neq e' \in E^2 \implies post_e(v) \neq post_{e'}(v')$.

It allows us, regardless of the starting worlds responding to a ϕ formula, to guarantee that there will not be two worlds of the same valuation : $post_e(v) = post_{e'}(v')$ only if $v = v'$ and $e = e'$, and v are all differents by the V-injectivity of $\mathscr{M}$, so $V((w,e))$ can't have duplicate (i.e.to

have a Valuation-injective structure). Lemma 23 and Definition 20 permit to get $\mathscr{M} \otimes \mathscr{E}$ V-inj avoiding a collapsing of worlds of the same valuation with the symb() function, so we can have a symbolic product update, and later a valid model checking.

Proposition 21. *Forall $\mathscr{M}$ V-inj s.t. $W \models \phi$ and forall $\mathscr{E}$ T-inj wrt ϕ, $\mathscr{M} \otimes \mathscr{E}$ is V-inj.*

Proof. Let $w, w' \in W$ with $V(w) \models \phi$ and $V(w') \models \phi$ and by the definition of $T-injectivity$, $\forall V(w), V(w')' \in 2^{WS}$, $\forall e, e' \in E$, $\forall V(w) \neq V(w')$ or $e \neq e' \implies post_e(V(w)) \neq post_{e'}(V(w'))$, so $V^{\otimes}((w,e)) \neq V^{\otimes}((w',e'))$. Consequently, $\mathscr{M} \otimes \mathscr{E}$ is V-inj. □

Definition 22 (symbolic product update)**.** *Let $F = \langle W_F^r, R_F^r, \mu_F^r \rangle$ be a symbolic Kripke structure on WS and $\chi = \langle W_\chi^r, R_\chi^r, \mu_\chi^r \rangle$ be a symbolic update model on $WS \cup WS+$ of a Transition-injective Event model (with propositional precondition, as required in definition of T-injective). The* symbolic product update *of F by χ is the symbolic Kripke structure on WS defined as*

- $W^r = \text{before}(\text{Forget}^{\exists}_{WS}(W_F^r \wedge W_\chi^r))$
- $R^r = \text{before}(\text{Forget}^{\exists}_{WS}(R_F^r \wedge R_\chi^r))$
- $\mu^r = \frac{\alpha}{\text{Marg}_{\Sigma, WS'}(\alpha)}$ *with* $\alpha = \text{before}(\text{Marg}_{\Sigma, WS}(\mu_F^r \times \mu_\chi^r))$

Note that this definition implies that *WS* is "quadruplicated", since we need fresh symbols to represent $\text{succ}(\text{after}(x))$ for $x \in WS$. We implicitly extend succ to range over $WS \cup WS+$ and after to range over $WS \cup WS'$, making sure that $\text{succ} \circ \text{after} = \text{after} \circ \text{succ}$. The resulting symbolic Kripke structure is on *WS* because the worlds in the product update are pairs (w,e), on $WS \cup WS'$, which is being forgotten and renamed.
With goods properties on $\mathscr{M}$ (V-injective) and on $\mathscr{E}$ (T-injective), we have :

Lemma 23. *For any V-inj $\mathscr{M}$ and any T-inj $\mathscr{E}$ with propositional preconditions,* $\text{symb}(\mathscr{M} \otimes \mathscr{E}) = \text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})$

The following result, of which we omit the straightforward but tedious proof, puts all pieces together; then we complete Def. 15 with the case of the update operator.
We note $\mathscr{L}_{PDEL}^{0\text{-}T\text{-}inj}$ the language in which update models have only propositional preconditions and are Transition-injective.

Definition 24. *Let $\mathscr{M} = \langle W, R, \mu^r \rangle$ be a V-inj Kripke structure, $\mathscr{E}$ be a Transition-injective update model, and ϕ be a formula in $\mathscr{L}_{PDEL}^{0\text{-}T\text{-}inj}$.*
The Boolean translation of ϕ in $\text{symb}(\mathscr{M})$*, denoted $||\phi||_{\text{symb}(\mathscr{M})}$, is the Boolean function defined inductively in the same way as in Def. 15, with the following additional case:*
$||[\mathscr{E}, e]\phi||_{\text{symb}(\mathscr{M})} \coloneqq \text{Forget}^{\exists}_{WS+}(pre(e) \rightarrow (\theta_e^{post} \wedge \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})})))$

We can see $\theta_e^{post} \wedge \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})})$ as $\{w, w^+ \mid w^+ = post_e(w),\ F \times \text{symb}(\mathscr{E}), \text{before}(w^+) \models \phi\}$. So w^+ are worlds after product update that satisfy ϕ, and w the worlds which led to w^+, and forgetting the *WS+* allows to keep the worlds that led to it. With this new definition on translation, we obtain a proposition to do symbolic model checking on PDEL.

Proposition 25 (symbolic model checking on PDEL)**.** *Let $\mathscr{M}$ be a V-inj Kripke structure. For any formula ϕ in $\mathscr{L}_{PDEL}^{0\text{-}T\text{-}inj}$ and any $w \in W$, it holds that $\mathscr{M}, w \models \phi \iff ||\phi||_{\text{symb}(\mathscr{M})}(V(w)) = 1$.*

Proof. The proof is by induction on ϕ. For case of $\phi \in \mathscr{L}_{PEL}$, see proposition 16.

$\mathscr{M}, w \models [\mathscr{E}, e]\phi$
$\iff \mathscr{M}, w \models pre(e) \implies \mathscr{M} \otimes \mathscr{E}, (w,e) \models \phi$. (Definition 9)
$\iff \mathscr{M}, w \models pre(e) \implies ||\phi||_{\text{symb}(\mathscr{M} \otimes \mathscr{E})}(V^{\otimes}((w,e))) = 1$ (Hyp and Proposition 21)
$\iff pre(e)(V(w)) = 1 \rightarrow ||\phi||_{\text{symb}(\mathscr{M} \otimes \mathscr{E})}(V^{\otimes}((w,e))) = 1$ (preconditons are propositional)
$\iff pre(e)(V(w)) = 1 \rightarrow ||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})}(V^{\otimes}((w,e))) = 1$ (Lemma 23)
$\iff pre(e)(V(w)) = 1 \rightarrow ||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})}(post_e(V(w))) = 1$ (Def of $post_e(v)$)
$\iff pre(e)(V(w)) = 1 \rightarrow \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})})(\text{after}(post_e(V(w)))) = 1$
$\iff pre(e)(V(w)) = 1 \rightarrow \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})})(\theta_e^{post}{}_{|V(w)}) = 1$ (Lemma 18)
$\iff pre(e)(V(w)) = 1 \rightarrow (\theta_e^{post} \wedge \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})}))(V(w) \wedge \theta_e^{post}{}_{|V(w)}) = 1$ (Lemma 18) (translation still only work on $\theta_e^{post}{}_{|V(w)}$ and $\theta_e^{post} \models V(w) \wedge \theta_e^{post}{}_{|V(w)}$)
$\iff \text{Forget}^{\exists}_{WS+}(pre(e))(V(w)) = 1 \rightarrow \text{Forget}^{\exists}_{WS+}((\theta_e^{post} \wedge \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})})))(\text{Forget}^{\exists}_{WS+}(V(w) \wedge \theta_e^{post}{}_{|V(w)})) = 1$ (forget is free in *pre* which is on *WS*, on forgetting on right is possible because of the Transition-injectivity of the event model : there is one valuation on $WS \cup WS+$)
$\iff \text{Forget}^{\exists}_{WS+}(pre(e))(V(w)) = 1 \rightarrow \text{Forget}^{\exists}_{WS+}((\theta_e^{post} \wedge \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})})))(V(w)) = 1$
$\iff \text{Forget}^{\exists}_{WS+}(pre(e) \rightarrow (\theta_e^{post} \wedge \text{after}(||\phi||_{\text{symb}(\mathscr{M}) \otimes \text{symb}(\mathscr{E})})))(V(w)) = 1$
$\iff ||[\mathscr{E}, e]\phi||_F(V(w)) = 1$

□

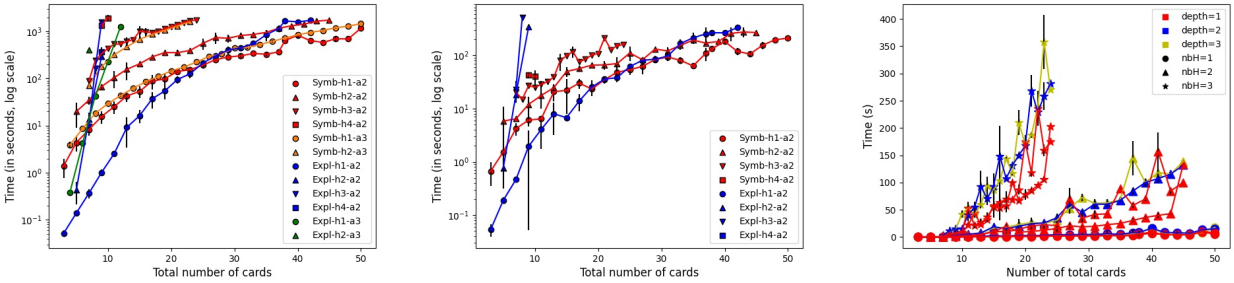


Figure 3: Computation time for the creation of Kripke structures (left) and for updating knowledge (middle) in their explicit and symbolic versions, and for model checking PEL formulas on symbolic structures. “hX-aY” means Y agents with X cards in hand. “depth” is the modal depth of the formula, “nbH” is the number of cards in each player’s hand.$\mathscr{M}$

4 Experiments on Hanabi

The primary interest to test model checking with PDEL with Hanabi as example is to see the card game aspect with inherent probabilities. It seems quite justified to ask "what is the probability that this card is 2 green", "is this probability greater than that this card is 2 blue"? In this game it is not always possible to make a safe move: when playing in an unsafe context, players will (generally) favor moves that they think have the highest probability of success. Our (longer-term) goal being to compute strategies for Hanabi, we need to take probabilities into account. Admittedly, the PDEL setting is a lot more general than necessary for Hanabi, but since our symbolic approach is powerful enough to handle it, it would have been strange to restrict its applicability to a less expressive framework.

4.1 Explicit vs symbolic

In this section, we report on experiments we ran on our Python implementation of the PDEL framework, with both on “explicit” and symbolic Kripke structures. The concrete language we used to represent PBFs is that of ADDs [Bahar *et al.*, 1997], which, while strictly less succinct than other decision diagram languages such as SLDDs, has efficient algorithms for almost all the operators we need [Fargier *et al.*, 2014]. We used our own Python implementation of an ADD package. The variable order in ADDs is of crucial importance to get small representations; we use a natural order that arise when compiling formulas, after having experimentally checked that varying the position of variable groups (e.g., putting variables from WS closer to WS' or to $WS+$) did not seem to yield much better results.

On the ADD structure, basic operations are polynomial in the size of the structure. For example applying an operator (or, and, sum, product) between 2 adds is quadratic. The problem is for Marginalisation or Forget: it’s quadratic in the worst case, for *a single variable*, so it can explode when you need to forget a lot. However (1) it doesn’t explode if the variables to be forgotten are at the end of the order, and (2) in practice it remains reasonable because there are a lot of symmetries in the structures, before and after marginalisations. The aim of the experiments is precisely to see that in practice, at scales sufficient to represent Hanabi, it works.

All experiments reported here have been performed on a server with 4 AMD Opteron 6282SE 2.6GHz processors, 64 cores and 512G RAM (but no experiment exploited any parallelism or multithreading. This power has not been exploited and the calculation times are roughly the same on a standard laptop). Each experiment has been run five times; the graphs show the average time, with a black line indicating standard deviation.

In Fig. 3 (left) , we can compare the creation times of static Kripke structures with event structures, in the explicit and symbolic cases, with varying total number of cards and number of cards in each player’s hand, with a timeout of 1800 seconds. For 2 agents, we can compare the red curves (symbolic) and blue curves (explicit). For nbH=1, the gain appears only from 25 cards. For nbH=2, the difference is more drastic: with the symbolic approach we can generate the game with 46 cards, versus only 11 for the explicit one. We can make similar observations for 3 agents. Let us note that the creation time of the symbolic structures is mainly used for creating the update models, which have twice as many variables as the static structures (because of the variables after(var)). For testing the application of the product update, we used the most complex action of the game: playing a card. It is programmed as an “action pipeline”: play a card, then shift the cards in hand to fill the “hole”, draw a new card, change turns. Results (Fig. 3, middle) are similar to those about creation time; the gain becomes flagrant for nbH=2 or 3.

For model checking on structures (only with 2 agents), we use a number of formulas, up to a depth of 3, using K and μ operators, and apply them on different Kripke structures with varying nbH. For the explicit structures that could be generated, model checking takes less than 5 seconds for almost all formulas. For symbolic structures (Fig. 3, right), propositional formulas are instantaneous, but the model checking of formulas with K or μ operators requires forgetting variables and time-consuming calculations. Yet, it is feasible in practice, and it compares very advantageously to explicit structures when those cannot even be generated.

4.2 Experiments with a KBP

Knowledge-based programs (KBPs), introduced by Fagin *et al.* [1997], are a way to represent plans. A KBP is a program where conditions and loops are epistemic logic formulas. We implemented a small KBP, presented in algorithm 1, that is intended as a rather naive but still reasonably realistic example of a policy for playing Hanabi. We ran this program to check whether our framework made it possible to execute a policy in such a form.

Algorithm 1 KBP

Require: $(\mathscr{M}, w)$ epistemic model, E list of events, a an agent, *nbH* number of cards in hands, A list of agents, *nbred* and *nbblue* number of current red and blue tokens
Ensure: An action of E
  for all $p \in \{1..nbH\}$ **do**
    if $\mathscr{M}, w \models \mathrm{K}_a$ a-can-play(p) **then**
      return a-plays-p
  for all *event* $\in E$ in announcement **do**
    for all $b \in A \backslash \{a\}$ **do**
      if $\mathscr{M}, w \models [!event] K_b$ b-can-play-something **then**
        return event
  $\gamma \leftarrow 0.66$ if *nbred*=0, 0.80 if *nbred*=1, 0.90 if *nbred*=2
  for all $p \in \{1..nbH\}$ **do**
    if $\mathscr{M}, w \models \mathrm{Pr}_a$ a-can-play-(p) $\geq \gamma$ **then**
      return a-plays-p
  if $nbblue = 0$ **then**
    return a-discards-nbH
  return random-annoucement

Let us first illustrate that this KBP makes sense. By varying the distribution of the cards at the beginning of the game, here is what we were able to obtain as a sequence of actions, for nbA=2, nbC=10, with pointed world as agent a has {W5, W2, W4} and b has {W2, W1, W4} in hand:

- b-has-v=1-at-1 because of [!event] K_b he-can-play-sth
- b-plays-p1, because K_b b-can-play-1
- b-has-v=2-at-1 because of [!event] K_b he-can-play-sth
- b-plays-p1, because K_b b-can-play-1
- random-announcement
- b-plays-p2, because of Pr_b b-can-play-2 ≥ 0.66

The relatively low number of cards allows here to have only yellow cards. Thus, players already know a large part of the information on their cards. All they have to do now is learn the value of their cards. Nevertheless, we can see from the sequence of actions generated by the KBP that the actions are well targeted and that the model checking with probabilities even allows agent b to take risks by playing a card that enters the game with a 66% chance.
Now, how many time does it take to execute such a KBP? We fixed nbA=2 and nbH=3, and we tried two game sizes: 10 cards or 20 cards in total. With the first size, the KBP require up to 46 model checkings, whereas it's 56 with the other size.

With nbC=10 Executing the KBP takes 18.88 (average on 3 runs), with a a standard deviation of 19.49, and the average time to apply a model checking is 2.99 seconds with standard deviation 3.56.

With nbC=10 Executing the KBP takes 21.08 (average on 2 runs), with a standard deviation of 22.42. The average time to apply a model checking is 19.53 seconds with a standard deviation 32.07.
All in all, executing a policy written as a KBP is possible in practice for small instances of Hanabi, and it does not take an unreasonably long time.

5 Conclusion

In this article we have shown that it is possible to do model checking on probabilistic Kripke structures using pseudo-boolean functions, implemented with structures such as ADDs. The approach is sufficiently scalable to be applicable to an almost-realistic instance of the card game Hanabi. Future work will rely on symbolic structures to apply planning techniques in order to compute strategies for Hanabi.

References

(2021). Hanabi rules. `https://www.ultraboardgames.com/hanabi/game-rules.php`. accessed 2021-01-16.

BAHAR, R. I., FROHM, E. A., GAONA, C. M., HACHTEL, G. D., MACII, E., PARDO, A. et SOMENZI, F. (1997). Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2/3):171–206.

BARD, N., FOERSTER, J. N., CHANDAR, S., BURCH, N., LANCTOT, M., SONG, H. F., PARISOTTO, E., DUMOULIN, V., MOITRA, S., HUGHES, E., DUNNING, I., MOURAD, S., LAROCHELLE, H., BELLEMARE, M. G. et BOWLING, M. (2020). The hanabi challenge: A new frontier for AI research. *Artif. Intell.*, 280:103216.

BAUZA, A. (2010). Hanabi. `http://www.antoinebauza.fr/?tag=hanabi`. accessed 2018-06-11.

BOLANDER, T. (2017). A gentle introduction to epistemic planning: The DEL approach. *In Proceedings of the Ninth Workshop on Methods for Modalities, M4M@ICLA 2017, Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017*, pages 1–22.

BRYANT, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Trans. Computers*, 35(8):677–691.

CHARRIER, T., PINCHINAT, S. et SCHWARZENTRUBER, F. (2019). Symbolic model checking of public announcement protocols. *J. Log. Comput.*, 29(8):1211–1249.

CHARRIER, T. et SCHWARZENTRUBER, F. (2017). A succinct language for dynamic epistemic logic. *In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*, pages 123–131.

DARWICHE, A. et MARQUIS, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, pages 229–264.

FAGIN, F., HALPERN, J. Y., MOSES, Y. et VARDI, M. Y. (1997). Knowledge-based programs. *Distrub Comput.*, 10:199–225.

FAGIN, R. et HALPERN, J. Y. (1994). Reasoning about knowledge and probability. *J. ACM*, 41(2):340–367.

FARGIER, H., MARQUIS, P., NIVEAU, A. et SCHMIDT, N. (2014). A knowledge compilation map for ordered real-valued decision diagrams. *In* BRODLEY, C. E. et STONE, P., éditeurs : *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1049–1055. AAAI Press.

FARGIER, H., MARQUIS, P. et SCHMIDT, N. (2013). Semiring labelled decision diagrams, revisited: Canonicity and spatial efficiency issues. *In* ROSSI, F., éditeur : *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pages 884–890. IJCAI/AAAI.

GATTINGER, M. (2018). *New directions in Model Checking Dynamic Epistemic Logic*. Thèse de doctorat, Universiteit van Amsterdam.

KISA, D., den BROECK, G. V., CHOI, A. et DARWICHE, A. (2014). Probabilistic sentential decision diagrams. *In* BARAL, C., GIACOMO, G. D. et EITER, T., éditeurs : *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

KOOI, B. P. (2003). Probabilistic dynamic epistemic logic. *Journal of Logic, Language and Information*, 12(4):381–408.

MAUBERT, B., PINCHINAT, S. et SCHWARZENTRUBER, F. (2019). Reachability games in dynamic epistemic logic. *In* KRAUS, S., éditeur : *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 499–505. ijcai.org.

MCMILLAN, K. L. (1993). *Symbolic model checking*. Kluwer.

SANNER, S. et MCALLESTER, D. A. (2005). Affine algebraic decision diagrams (aadds) and their application to structured probabilistic inference. *In* KAELBLING, L. P. et SAFFIOTTI, A., éditeurs : *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 1384–1390. Professional Book Center.

SHIRAZI, A. et AMIR, E. (2008). Factored models for probabilistic modal logic. *In* FOX, D. et GOMES, C. P., éditeurs : *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 541–547. AAAI Press.

van BENTHEM, J., GERBRANDY, J. et KOOI, B. P. (2006a). Dynamic update with probabilities. Prepublication/extended version PP-2006-21 (March 15th, 2006) [https://www.illc.uva.nl/Research/Publications/Reports/PP/#PP-2006-21], ILLC (Institute for Logic, Language and Computation), University of Amsterdam.

van BENTHEM, J., GERBRANDY, J. et KOOI, B. P. (2009). Dynamic update with probabilities. *Studia Logica*, 93(1):67–96.

van BENTHEM, J., van EIJCK, J., GATTINGER, M. et SU, K. (2017). Symbolic model checking for Dynamic Epistemic Logic — S5 and beyond. *Journal of Logic and Computation*, 28(2):367–402.

van BENTHEM, J., van EIJCK, J. et KOOI, B. P. (2006b). Logics of communication and change. *Inf. Comput.*, 204(11):1620–1662.

WILSON, N. (2005). Decision diagrams for the computation of semiring valuations. *In* KAELBLING, L. P. et SAFFIOTTI, A., éditeurs : *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 331–336. Professional Book Center.

Planning for Connected Agents in a Partially Known Environment

Arthur Queffelec[†], Ocan Sankur[‡], François Schwarzentruber[†]
[†] Univ Rennes, CNRS, IRISA
[‡] Univ Rennes, Inria, CNRS, IRISA

Abstract

The Connected Multi-Agent Path Finding (CMAPF) problem asks for a plan to move a group of agents in a graph while respecting a connectivity constraint. We study a generalization of CMAPF in which the graph is not entirely known in advance, but is discovered by the agents during their mission. We present a framework introducing this notion and study the problem of searching for a strategy to reach a configuration in this setting. We prove the problem to be PSPACE-complete when requiring all agents to be connected at all times, and NEXPTIME-complete in the decentralized case, regardless of whether we consider a bound on the length of the execution.

Keywords: multi-agent path finding, multi-agent planning, connectivity, imperfect knowledge, decentralized planning

1. Introduction

The coordination of mobile agents is at the heart of many real world problems such as traffic control [1], robotics [2, 3], aviation [4] and more [5, 6]. Some of these problems have multiple aspects which make them complex: (1) Some systems are *multi-agent*, that is, the behaviors of agents influence others' and these influences must be taken into consideration when computing missions; this can be due for instance, to collisions [7], sensor interferences [8, 9] etc.; (2) Some missions must ensure *connectivity*, that is, ensure periodic or constant connection to a station/agent to share acquired information [10]; (3) The environment may be only *partially known*, and the agents may discover it during the mission [11, 12]. Several works have considered problems containing these three aspects. For instance, several algorithmic approaches have been investigated to solve the coordination of multi-robot exploration [13–15]. Our objective in this paper is to present a framework to study the theoretical complexity of planning problems with respect to these three aspects.

The theoretical complexity of some related problems have been studied in the literature. Multi-Agent Path Finding (MAPF) is an important framework introduced to study collision-free navigation of agents in warehouses (see [7, 16]). This problem was intensively studied and gave rise to a popular algorithm known as Conflict-Based Search (CBS) [17]. An extension of MAPF with connectivity constraints, called *Connected MAPF* (CMAPF), was studied as well [18]. The complexity of CMAPF and algorithmic solutions were studied in [19, 20]. However, CMAPF only addresses the multi-agent and connectivity aspects, and not the partial knowledge of the environment. The latter aspect is considered in the Canadian Traveler Problem (CTP), which is a well-known problem to study the navigation of an agent in a partially known graph [21]. While the initial framework was for a single agent, CTP has been extended to multiple agents in the settings of packet routing [22], multi-robot exploration [23] and more [24]. While a notion of communication was considered in [25, 26], it is limited to settings where all agents can receive information at all times or only designated agents can send information. In contrast, we are interested in studying the setting where agents' ability to communicate depends on their positions in the graph, and in establishing theoretical complexity results of resulting problems.

In this paper, we study the theoretical complexity of generating plans for a group of agents to reach a given target configuration. More precisely, we analyze the impact of enforcing or

ignoring: (A) connectivity; (B) collision; (C) a bound on the length of the execution. For (A), we consider either *fully-connected* strategies, requiring that the agents remain connected at all times during the mission, or a *decentralized* strategy, allowing agents to disconnect and reconnect. (B) In some applications, collisions can be handled by a local collision avoidance system [18], and one can thus abstract away and ignore collisions in graph-based planning algorithms. (C) By providing a bound on the execution length, we can study the complexity of the decision problem associated to the optimization problem. Our results are summarized in Table 1. Interestingly, The PSPACE algorithm for the connected problem with a bounded execution is subtle and relies on a variant of the Savitch's theorem [27] we present here. Additionally, the PSPACE-completeness holds even in the case in which agents can always communicate, thus the hardness of the problem already comes from the incomplete knowledge of the movement graph. For the decentralized case, we prove the NEXPTIME-hardness in the bounded and unbounded cases by two separate reductions from the True Dependent Quantified Boolean Formula problem (TDQBF) [28], thus showing that the problem becomes significantly harder in this case.

Let us compare our results with known complexity results. In the fully known environment, the CMAPF problem is PSPACE-complete in the connected and unbounded case [19], while it is NP-complete in the bounded case (with the bound given in unary) [18]. Thus, the partial knowledge of the environment does not render the problem harder in terms of complexity. In contrast, recall that in MAPF (without connectivity), one can check the existence of a solution in polynomial time [29], while the bounded problem is NP-hard [30], so the hardness is due to connectivity constraints. Some algorithms were presented for CMAPF in [19] that can scale up to about ten agents. Since both problems are similar and belong to PSPACE, one can hope that these approaches can be extended to the partial knowledge case. On the other hand, our results show that the complexity of the decentralized case is significantly higher. While some tools and algorithms are available for Decentralized POMDPs, which are in the same complexity class, the scalability is limited and the development of efficient algorithms for this case seems more challenging (see [31] for a survey).

Outline. In Section 2, we recall the CMAPF and TDQBF (True Dependency QBF) problems. In Section 3, we formalize our imperfect information setting. Our results for the connected and decentralized cases are given in Sections 4 and 5 respectively. Section 6 contains additional results that can be obtained from our work, discussion on related work, and perspectives.

	Decentralized	Connected
bounded	NEXP-complete (Th. 7)	PSPACE-complete (Th. 4, 5)
unbounded	NEXP-complete (Th. 6)	PSPACE-complete (Th. 3)

Table 1. Complexity Results.

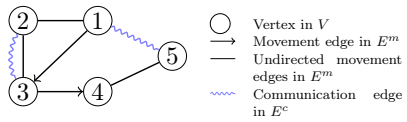


Figure 1. Example of a topological graph.

2. Preliminaries

Connected MAPF. We consider a setting where agents move on a graph, with the *vertices/nodes* being their possible locations and the *movement edges* defining how they can move. In addition, *communication edges* define how the agents can communicate. The graphs we consider thus have two types of edges and are called *topological graphs*.

Definition 1. *A* topological graph *is a tuple* $G = \langle V, E^m, E^c \rangle$*, with* V *a finite non-empty set of vertices,* $E^m \subseteq V \times V$ *a set of movement edges and* $E^c \subseteq V \times V$ *a set of undirected communication edges.*

A movement edge (u, v) is said to be undirected if $(u, v), (v, u) \in E^m$. Figure 1 gives an example of a topological graph. For instance, an agent can go from 1 to 3 in one step. Two agents at vertices 1 and 5 can communicate, but two agents at 1 and 4 cannot.

We suppose that each vertex has a self-loop movement edge and a self-loop communication edge. This respectively represents the ability of an agent to stay at their location and to communicate with a nearby agent (i.e. at the same location/vertex).

Definition 2. *A* configuration *is a tuple $c = \langle c_1, \ldots, c_n \rangle$ where c_i is the vertex of agent i.*

We write $c \to c'$ when $(c_i, c'_i) \in E^m$ for all $1 \leq i \leq n$. This means that each agent i can move from their vertex c_i in c to their vertex c'_i in c' in one step.

Definition 3 (Execution)**.** *An execution π is a sequence of configurations $\langle \pi[0], \pi[1], \ldots, \pi[\ell] \rangle$ such that $\pi[t] \to \pi[t+1]$ for all $t \in \{0, \ldots, \ell - 1\}$.*

We denote the length of π by $|\pi|$, and write $\pi[0..t-1]$ to denote the sub-execution $\langle \pi[0], \pi[1], \ldots, \pi[t-1] \rangle$ of π. Moreover, $\mathsf{last}(\pi)$ denotes the last configuration of π.

We say that a configuration c is *connected* iff the subgraph of the vertices occupied by the agents form a connected graph for relation E^c, i.e. the graph $\langle V_a, E^c \cap (V_a \times V_a) \rangle$ is connected with $V_a = \{c_1, \ldots, c_n\}$. An execution is said to be *connected* iff all its configurations are connected. The Connected Multi-Agent Path Finding problem consists in finding a connected execution for the agents from a given initial configuration to a target configuration. We summarize below the known complexity results for the CMAPF problem.

Theorem 1 ([18])**.** *Deciding whether there is a* connected *execution from c^s to c^t of length at most k in a given instance (G, c^s, c^t, k), where k is written in unary is NP-complete.*

Theorem 2 ([19])**.** *The problem of deciding whether for a given instance (G, c^s, c^t, ∞), there exists a connected execution from c^s to c^t is PSPACE-complete.*

We do not consider collision constraints which would forbid agents from sharing the same vertex. For CMAPF with perfect knowledge, executions are not harder to compute with or without collision constraints [19, 20]. The same holds in our case, and we discuss, in the Section 6, how to incorporate these constraints in our setting.

Dependency Quantified Boolean Formula. A Dependency QBF (DQBF) is a formula in which dependencies of existential variables over universal variables are explicitly specified. A DQBF is of the form $\forall y_1, \ldots, y_n \exists x_1(O_{x_1}) \ldots \exists x_n(O_{x_n})\ \psi$, where each O_{x_i} is, the *dependency set*, a subset of universally quantified variables, and ψ is a Boolean formula in CNF over $x_1, \ldots, x_n, y_1, \ldots, y_n$. It is worth noting that a QBF can be seen as a DQBF with $O_{x_1} \subseteq O_{x_2} \subseteq \ldots \subseteq O_{x_n}$.

The *True DQBF* (TDQBF) is the problem of deciding whether a given DQBF holds. Formally, a DQBF φ holds iff there exists a collection of Skolem functions $A = (A_{x_i} : \{0,1\}^{O_{x_i}} \to \{0,1\})_{i=1..n}$ such that replacing each existential variable x_i by a Boolean formula representing A_{x_i}, turns ψ into a tautology. TDQBF is NEXPTIME-complete [28], and will be used to prove NEXPTIME lower bounds in Section 5.

3. Our framework

Modeling Imperfect Information. To formalize CMAPF in the imperfect information setting, let us show how to represent the initial knowledge of the agents, and how the information they have evolves during the execution. Agents initially know the exact set of vertices, but only have a lower and an upper approximation of the actual graph: they know that some (communication or movement) edges are *certain* (they must be present), while some are *uncertain* (they may be absent).

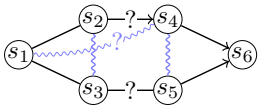


Figure 2. An initial knowledge (G_1, G_2).

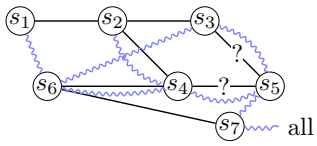


Figure 3. An initial knowledge of a topological graph. Vertex s_7 has communication edges with all other vertices.

Certain movement edge
Certain undirected movement edge
Uncertain movement edge
Uncertain undirected movement edge
Certain communication edge
Uncertain communication edge

Definition 4 (Initial Knowledge)**.** *The initial knowledge is modeled by a pair of topological graphs* (G_1, G_2)*, with the graph* $G_1 = \langle V, E_1^m, E_1^c \rangle$ *a lower bound, and* $G_2 = \langle V, E_2^m, E_2^c \rangle$ *an upper bound on the knowledge about the actual graph with* $E_1^m \subseteq E_2^m$ *and* $E_1^c \subseteq E_2^c$*.*

The agents initially know G_1 and G_2 while the actual graph $G = \langle V, E^m, E^c \rangle$ is initially unknown to them. They only know that $E_1^m \subseteq E^m \subseteq E_2^m$ and $E_1^c \subseteq E^c \subseteq E_2^c$, written as $G_1 \subseteq G \subseteq G_2$. The perfect information case is captured by $G_1 = G_2 (= G)$.

A movement (resp. communication) edge is said to be *certain* (i.e. sure to be present) if it is in E_1^m (resp. E_1^c); it is said *uncertain* (i.e. can be absent) if it is in $E_2^m \setminus E_1^m$ (resp. $E_2^c \setminus E_1^c$). We assume that the communication edges of the actual graph are undirected, so for all $(u, v) \in E_2^c \setminus E_1^c$, either $(u, v), (v, u) \in E^c$, or $(u, v), (v, u) \notin E^c$.

We say that an edge (u, v) is an *uncertain undirected movement edge* when $(u, v), (v, u) \in E_2^m \setminus E_1^m$. The environment can leave both edges $(u, v), (v, u)$, remove both edges $(u, v), (v, u)$, but also remove (u, v) and leave (v, u), or remove (v, u) and leave (u, v). That is, the movements edges of the actual graph are not necessarily undirected.

Example 1. *Figure 2 depicts an initial knowledge* (G_1, G_2)*. The area is divided in two zones connected by two bridges, represented by the edges* (s_2, s_4) *and* (s_3, s_5)*, with an uncertainty on which bridge is open and on the communication between* s_1 *and* s_4*.*

A strategy σ_i for agent i tells where to go next after a given execution π. Formally:

Definition 5. *A* strategy σ_i *for agent* i *maps any execution* π *to a vertex such that* $(v, \sigma_i(\pi)) \in E^m$ *where* v *is the vertex at which agent* i *is in the last configuration of* π*.*

A *joint strategy* σ is a tuple $\langle \sigma_1, \ldots, \sigma_n \rangle$ where σ_i is a strategy for agent i. The *outcome* of a joint strategy σ starting from configuration c^s is the execution π defined by induction as follows: $\pi[0]$ is c^s, and for $t \geq 1$, $\pi[t]$ is the configuration in which agent i is at vertex $\sigma_i(\pi[0..t-1])$.

In the context of imperfect information, the behaviors of the agents only depend on their observations, as in imperfect information games as in [32]. The strategies, as defined above, do not necessarily take observations of the agents into account. We will now formalize observations and *uniform* strategies, that is, those respecting the observations of the agents.

In our setting, at any time, an agent observes all movement edges adjacent (both in- and out-coming) to the vertex v it occupies. Moreover, they observe the presence or absence of a communication edge between v and v' if v' is occupied by another agent with which there is a direct or indirect communication (via other agents). Intuitively, during an execution, at each step, each agent updates their knowledge about the graph with these observations they receive. Moreover, they share all their knowledge with all agents with which they are connected at each step.

The observation of adjacent movement edges has been a recurrent practice in theoretical works [**Bar-Noy:91:CTP**, 21] as well as robotics [**Dudek:91:REGC**, **Koenig:01:GMT**], and our formalism is inspired from these works.

The knowledge of an agent at any time corresponds intuitively to a pair of graphs as in Definition 4. For agent i and execution π, let us denote by $k_i(\pi)^G$ the knowledge of agent i about the graph after observing the execution π in actual graph G. Given such knowledge $K = k_i(\pi)^G$, the agent can deduce an under-approximation and an over-approximation of the actual graph; let us denote these by $\underline{G}^K$ and $\overline{G}^K$ respectively. In particular, if K is the knowledge the agents have initially, then $\underline{G}^K = G_1$ and $\overline{G}^K = G_2$. We present a representation of the knowledge using predicates in the appendix where a detailed formalization of $k_i(\pi)^G$ can be found.

In the rest of the paper, we assume all considered strategies to be *uniform*, that is, they comply with the knowledge of the agents: the strategies prescribe the same move to all executions that are indistinguishable with the agent's observations. Formally, if $|\pi| = |\pi'|$ and $k_i(\pi)^G = k_i(\pi')^G$, then $\sigma_i(\pi) = \sigma_i(\pi')$.

Decision Problems. We consider the decision problem of reaching a configuration c^t from a configuration c^s in less than k steps, using uniform strategies. For two configurations c^s, c^t, let us call a topological graph G (c^s, c^t)-*admissible* if there is an execution from c^s to c^t, which is not necessarily connected.

Definition 6. *We say that an instance (G_1, G_2, c^s, c^t, k) is positive if there exists a joint strategy σ such that in all (c^s, c^t)-admissible graphs G satisfying $G_1 \subseteq G \subseteq G_2$, the outcome of σ starting in c^s ends in c^t in less than k steps.*

Observe that the above problem requires that a strategy ensures the reachability of the target configuration only for graphs that are (c^s, c^t)-admissible and compatible with the initial knowledge. In fact, intuitively, we would like the strategy to work under all possible graphs G with $G_1 \subseteq G \subseteq G_2$. However, requiring a strategy to ensure reachability in a non-admissible graph does not make sense, since even a strategy with full information would fail. We thus require the strategies to make their best efforts, that is, to ensure the objective unless it is physically impossible.

Example 2. *Consider the example of Figure 2. If both bridges (i.e. movement edges (s_2, s_4) and (s_3, s_5)) are absent in the actual graph, the graph is not (s_1, s_6)-admissible and there cannot be a strategy ensuring reachability. The admissible graphs contain either (s_2, s_4), or (s_3, s_5), or possibly both. Note that, this instance is negative for $k < 6$. Indeed, consider a strategy that moves the agent, for instance, to s_2. In the graph where (s_2, s_4) is absent, the agent would need to come back to s_1 and take the alternative path, which requires an execution of total length 6; and the situation is symmetric if the first move is towards s_3. The instance is nonetheless positive for $k \geq 6$ with the described strategy. However, if the edges (s_2, s_1) and (s_3, s_1) were not present, then the instance would be negative. In fact, once the agent moves to s_2 or s_3, they get stuck if the graph only contains the other bridge.*

We now define the *connected* version of Definition 6. For two configurations c^s, c^t, we say that a topological graph G is (c^s, c^t)-*c-admissible* if there is a connected execution from c^s to c^t. We will often omit the pair of configurations which will be clear from the context, and write simply admissible or c-admissible.

Definition 7. *We say that an instance (G_1, G_2, c^s, c^t, k) is c-positive if there exists a joint strategy σ such that in all (c^s, c^t)-c-admissible graphs G satisfying $G_1 \subseteq G \subseteq G_2$, the outcome of σ starting in c^s is connected and ends in c^t in less than k steps.*

In the connected case, agents cannot visit a disconnected configuration. Hence, the considered strategies only visit configurations that are certainly connected. Observe that agents can make observations about the presence or absence of communication edges while being connected and use this information later.

Example 3. *Let us illustrate the above property on the example of Figure 3. Assume there are two agents, the starting and goal configurations are* $c^s = \langle s_1, s_6 \rangle$ *and* $c^t = \langle s_5, s_7 \rangle$*, and the only uncertainty is about the movement edges* (s_3, s_5) *and* (s_4, s_5)*. Here, Agent* 2 *could immediately move to her target* s_7*; however, she could also cooperate with Agent* 1 *and lower the total completion time. Indeed, from their start configuration* $\langle s_1, s_6 \rangle$*, the agents first move to* $\langle s_2, s_4 \rangle$ *where Agent* 2 *observes whether* (s_4, s_5) *is present. Assume the edge is present. Then, they follow the sequence* $\langle s_4, s_6 \rangle \cdot \langle s_5, s_7 \rangle$*; and otherwise* $\langle s_3, s_6 \rangle \cdot \langle s_5, s_7 \rangle$*. Thus, in order to minimize the length of the execution, the agents do not always take their shortest paths but might help other agents by obtaining information about the graph.*

Consider now the same example in which the communication edge (s_3, s_6) *is uncertain. If this edge is absent then Agent* 2 *cannot help Agent* 1 *achieve the target faster since if the former moves to* s_4 *and* (s_4, s_5) *is absent, then, in order to maintain connectivity, the next configurations should be* $\langle s_6, s_4 \rangle \cdot \langle s_2, s_7 \rangle \cdot \langle s_3, s_7 \rangle \cdot \langle s_5, s_7 \rangle$*. An execution of the same size is obtained when Agent* 2 *moves to* s_7 *in the first step.*

For both Definitions 6 and 7 above, let us call a joint strategy a *witness* if it witnesses the fact that the given instance is positive, and respectively, c-positive.

We instantiate the Connected MAPF problem in four different settings. The four following decision problems are defined depending on whether we consider the connectivity requirement and whether the bound is finite. Note that the bounded problems are the decision problems associated to the optimization problems.

Bounded Decentralized Reachability. Is given (G_1, G_2, c^s, c^t, k), with $k < \infty$, positive?	*Unbounded Decentralized Reachability.* Is given $(G_1, G_2, c^s, c^t, \infty)$ positive?
Bounded Connected Reachability. Is given (G_1, G_2, c^s, c^t, k), with $k < \infty$, c-positive?	*Unbounded Connected Reachability.* Given $(G_1, G_2, c^s, c^t, \infty)$ c-positive?

As we will see, the encoding of the integer k does not change the overall complexity. Lower bounds are all obtained directly for the unary encoding; the lower bounds for the binary encoding follows. Concerning the upper bounds, we explain for each case how to design an algorithm with k encoded in binary.

4. Connected Reachability

We first address the case where agents must be connected at each step of the execution. In this case, agents share their knowledge at all times and thus the group of agents can be considered as a single agent playing against the environment.

Unbounded Case. We first focus on the existence of an unbounded connected strategy. Interestingly, we show that verifying the existence of a connected strategy in a partially known environment is not harder than in a perfectly known environment.

Theorem 3. *The unbounded connected reachability problem is PSPACE-complete.*

Proof Sketch. The PSPACE lower bound follows from Theorem 2 with $G_1 = G_2$.

For the upper bound, let us explain the intuition of an algorithm. Fix instance $I = (G_1, G_2, c^s, c^t, \infty)$. If I is c-positive, then G_2 must admit a connected execution π from c^s to c^t such that along this execution, whenever the absence of a set of edges is revealed, there should still exist a connected execution from the current configuration to c^t, unless the graph reveals to be not c-admissible. This is a recursive property that is necessary since it is satisfied by all witness strategies. We prove that it is also sufficient for c-positive instances. In the full proof, we show how to check this property in polynomial space. □

Bounded Case. We now study the existence of a bounded connected strategy. We show that this problem is PSPACE-complete even when the communication graph is complete. Please find in supplementary material detailed proofs of the Lemmas 1, 2 and Theorem 5.

Theorem 4. *The bounded connected reachability problem is in PSPACE when the bound is given in binary.*

Proof. Let us first prove the upper bound when k is given in unary. As APTIME = PSPACE [33], we give an alternating algorithm that runs in polynomial time, as follows. At each step, the existential player chooses the next connected configuration to move the agents; and the universal player chooses the information about the newly discovered edges. After k steps the algorithm accepts if the target configuration is reached, or the revealed edges mean that the graph is not c-admissible. The number of steps is bounded by k, which is polynomial, thus the algorithm runs in polynomial time.

There is one subtlety to prove the correctness. The alternating algorithm actually corresponds to a slight variant of our setting which can be seen as a game. In our setting, the environment chooses a graph G with $G_1 \subseteq G \subseteq G_2$ at the beginning, and the agents discover the graph G as they move. In contrast, in the alternating algorithm, the universal player reveals the graph step by step; therefore the environment might adapt the graph to the moves of the existential player.

Lemma 1. *The alternating algorithm is correct.*

When k is binary, the previous algorithm does not run in polynomial time. However, observe that the number of alternations can be bounded by a polynomial because there is only a polynomial number of steps in which the universal players reveal *new* information to the coalition of agents. In fact, the universal player is only useful when some agent is at a vertex that has not been seen before, and this can only happen a linear number of times. Furthermore, the previous algorithm runs in polynomial space.

When k is binary, our problem is in $\mathsf{STA}(\mathsf{poly}(n), *, \mathsf{poly}(n))$ where $\mathsf{STA}(s(n), t(n), a(n))$ is the set of problems decided in space $O(s(n))$, time $O(t(n))$ with $O(a(n)))$ alternations. Our problem is in PSPACE thanks to the generalization of Savitch's theorem we prove:

Lemma 2. $\mathsf{STA}(\mathsf{poly}(n), *, \mathsf{poly}(n)) \subseteq \mathsf{PSPACE}$.

Intuitively, this lemma is proved by guessing the computations between each universal choice by a PSPACE oracle, which yields an overall PSPACE algorithm. □

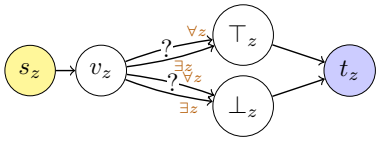
Theorem 5. *The bounded connected reachability problem with complete connectivity is PSPACE-hard.*

The proof is by reduction from the truth of a quantified binary formula. The uncertain edges are used to model the universally quantified variables, so that the choice of the actual graph corresponds to a valuation for these. The existence of a connected execution corresponds to the choice of existential variables, while connectivity constraints are used to ensure the satisfaction of all clauses.

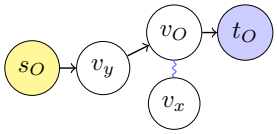
Our reduction actually builds an undirected movement graph. Thus, PSPACE-hardness holds already for undirected movement graphs. Note that in our current setting, pairs of uncertain edges of the form (u, v) and (v, u) are treated separately, but the lower bound proof still holds when they are seen as one.

5. Decentralized Reachability

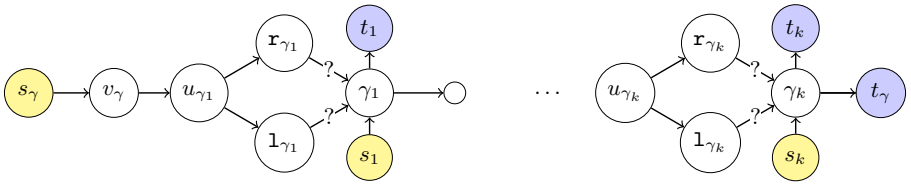
We now tackle the reachability problem when agents are allowed to be disconnected. At each configuration, agents share their knowledge with all agents to which they are connected.



(a) Gadget for variable z. Both edges $(v_z, \top_z)$ and $(v_z, \bot_z)$ are certain (resp. uncertain) if z is existential (resp. universal).



(b) Gadget for the observation O of universal variable y from existential variable x.



(c) Clause gadget.

Certain movement edge
Certain communication edge
Uncertain movement edge
Certain (resp. uncertain) movement edge if z is existential (resp. universal)

$(\top_z, \gamma_i) \in E_1^c$ when z appears in γ_i
$(\bot_z, \gamma_i) \in E_1^c$ when $\neg z$ appears in γ_i
$(\top_z, v_\gamma) \in E_1^c, (\bot_z, v_\gamma) \in E_1^c$

Figure 4. Gadgets in the reduction from DQBF to unbounded decentralized reachability.

The decentralized case is intuitively harder because the agents no longer follow a centralized strategy, nor do they act independently; but they must cooperate to exchange information at the right moment in order to reach their targets. See supplementary material for additional details.

Unbounded Case. We, first, show that the unbounded decentralized reachability problem is NEXPTIME-complete.

Theorem 6. *The unbounded decentralized reachability problem is NEXPTIME-complete.*

Proof Sketch. For the upper bound, an NEXPTIME algorithm consists in guessing uniform strategies for all agents and checking whether the joint strategy is a witness. Such a strategy has exponential size since it is a function of the sets of knowledge of the agents and the current vertex. One can enumerate all graphs G between G_1 and G_2, and execute the joint strategy on G to check that it ensures the reachability of the target. Moreover, the executions to be checked have at most exponential length. In fact, executions can be seen as paths in a meta-graph where vertices are configurations augmented with the sets of knowledge of the agents. This meta-graph is of exponential size, so it is sufficient to consider executions of exponential length. The overall non-deterministic algorithm is thus in exponential time.

The lower bound is shown by reduction from TDQBF. Given a DQBF $\varphi = \forall y_1, \ldots, y_n \exists x_1(O_{x_1}) \ldots \exists x_n(O_{x_n})\ \psi$, we build an instance (G_1, G_2, c^s, c^t, k) of unbounded decentralized reachability. We denote by $\gamma_1, \ldots, \gamma_m$ the clauses in ψ.

The graph G_1 and G_2 as follows. For each variable z, we create the gadget depicted in Figure 4a. We create the observation gadget for all existential variables x and for all (universal) variables $y \in O_x$, depicted in Figure 4b. For convenience, we write O for the pair (x, y) corresponding to observation of y by x.

Finally, we create the clause gadget, depicted in Figure 4c. A vertex γ_i certainly communicates with $\top_z$ iff $z \in \gamma_i$, and with $\bot_z$ iff $\neg z \in \gamma_i$. Moreover, the vertex v_γ communicates with all $\top_z$ and $\bot_z$ for all variables z.

We define the initial and target configurations as $c^s = \langle s_\gamma, s_{\gamma_1}, \ldots, s_{\gamma_m}, s_{x_1}, \ldots, s_{x_n}, s_{y_1}, \ldots, s_{y_n}, s_{O_1}, \ldots, s_{O_k} \rangle$, and $c^t = \langle t_\gamma, t_{\gamma_1}, \ldots, t_{\gamma_m}, t_{x_1}, \ldots, t_{x_n}, t_{y_1}, \ldots, t_{y_n}, t_{O_1}, \ldots, t_{O_k} \rangle$.

We give some intuitions about the reduction; the full proof is in the supplementary material. Assuming the DQBF φ holds, one can build a witness strategy as follows. The agents starting at a universal s_z are forced to follow a path not deleted by the environment, which determines the value of z. The agents starting at s_O can observe the valuation of their respective universal variables (by visiting v_y), and then inform their respective existential variables thanks to the communication edge between v_O and v_x. Thus, agents starting at an existential s_z are aware of the values of all variables in O_z and choose an appropriate value so as to satisfy φ. The agents starting at s_i idle at γ_i at step 3. At this point, the agent that starts s_γ also idles at v_γ, and all agents that start s_z are at $\top_z$ or $\bot_z$. Thanks to the communication edges, and because φ is true, each γ_i communicates with at least some $\top_z$ or $\bot_z$, thus also with the agent at v_γ. Therefore, the latter agent knows about all available edges in the clause gadget and can successfully go to t_γ. □

Bounded Case. In the bounded case, the problem is NEXPTIME-complete independently of the encoding of the bound. Moreover, the hardness holds even for undirected graphs.

Theorem 7. *The bounded decentralized reachability problem is NEXPTIME-complete. NEXPTIME-hardness holds for undirected graphs.*

The NEXPTIME algorithm is similar to that of Theorem 6. We just add a counter in the algorithm to count the number of steps when the joint strategy is checked.

The lower bound is by reduction from TDQBF and follows the ideas of the reduction in Theorem 6. Let us mention one difference. In Theorem 6, directed edges were used in the clause gadget so that the clause agent gets stuck if they attempt to move without the necessary knowledge about the available edges. Here, we use a similar construction with undirected edges. In this case, the clause agent does not get stuck but if they make a wrong guess, they can no longer respect the time bound. Thus, the only way to reach the target within the time bound is to communicate with other agents at the right moment.

6. Discussion

Additional Results. We present results obtained by a simple observation/modification.

Unbounded Reachability and Undirected Graphs. Both the unbounded connected and unbounded decentralized reachability become *trivial* on undirected graphs. This is because we only require reachability for (c-)admissible graphs. In the decentralized case, each agent can run a DFS independently, and eventually reach their targets in at most $2|V|$ steps, and a similar search can be done by the set of agents in the connected case.

Base Station. Several works consider a designated *base* vertex to which all agents must stay connected during the execution [19, 20, 34]. This concept is only relevant in the connected case. Our results also hold with this additional constraint. In fact, the lower bound of Theorem 3 follows from [19], which proves the bound also with a base. In Theorem 5, we can add the base vertex as an isolated vertex so that the reduction is still valid.

Collisions. We did not require the paths to be collision-free in the results presented in this paper. However, this property is already ensured by our proofs or can be obtained by simple modifications. The lower bound proof of Theorem 3 relies on Theorem 2 from [19] which holds with collision constraints as well, so this is also true for our case. The proof of Theorem 5 does not generate collision-free paths as the groups of occurrence agents start and finish at the same location and follow almost the same path. This proof can be adapted to prevent collisions by delaying each occurrence agent by 3 steps behind one another. This can be achieved easily by extending the movement paths and shifting the starting

location and target location of an agent up by 3 vertices behind the previous agent. The proof of lower bound of Theorem 6 features a construction ensuring a collision-free strategy. Indeed, the observations agents only need to take turns to visit the universal variables. Thus, the result holds with collision constraints as well. The algorithms of Theorem 3, 5, and 7 can be adapted by restricting all considered configurations to collision-free ones; while c-admissibility of a graph with collision constraints can be checked using Theorem 2.

Graph Classes. The MAPF and CMAPF problems have been studied for different classes of graphs (planar, grid, ...). The proof of lower bound in Theorem 3 relies on the proof of unbounded reachability done in [20], thus the result of PSPACE-hardness on planar graphs also carries over to our problem. Planar QBF is known to be PSPACE-complete [35], and the construction of Theorem 5 is such that when applied to a planar QBF, the resulting graph is planar. Our PSPACE-completeness result thus holds on planar graphs.

Related Work. Different definitions of robust plans [36–38] have been studied. A *k-robust plan* guarantees the reachability of the target in the events of at most k delays. A *p-robust plan* executes without a conflict with probability at least p. Our framework does not consider delayed agents but focus on synchronous executions with imperfect knowledge of the area.

The problem of MAPF with a dynamic environment has multiple formulations. The Adversarial Cooperative Path-Finding [39] considers that the obstacles are agents which reason to prevent the cooperation to reach its goal. [40] considered the problem where the dynamics of the environment is predictable. Additionally, when obstacles have unknown dynamics, one can estimate their movements and plan to minimize the probability of a collision [41], or predict their movements [42] and plan online the movement of the agents [43]. In our setting, the environment is static, thus, all observations are fixed.

MAPF with Uncertainty (MAPFU) asks for a plan which guarantees that mishaps, localization and sensing errors do not impact the proper execution of the plan. This problem can be solved by temporal logic [44], POMDPs [45], replanning [46–48], interaction regions [1, 49], and belief space planning [50–52]. Nebel et al. [53] studied the MAPF problem with an uncertainty on the destination of the agents and lack of communication. The asynchronous movement of the agents, studied in those papers, cannot be expressed in our framework as we require the agent to follow some universal clock to execute their plan.

Perspectives. We proposed a setting for CMAPF in the imperfect case and studied the theoretical complexity of the reachability problem. The first natural question is to find classes of graphs (e.g. grid graphs) on which the reachability problem is easier to solve, as it was done for MAPF in [54, 55], and CMAPF in [20]. Another possible direction is to study the coverage of all vertices [20]. An alternative way to handle non-admissible graphs is to require that agents return to their starting configuration if the graph is discovered not to be admissible. We believe that such variants should be as hard as reachability. Furthermore, there are several possible generalizations that could be considered by introducing dynamic environments (instead of static), faulty sensing of agents, robustness, uncertainty, etc.

References

[1] K. Dresner and P. Stone. "A Multiagent Approach to Autonomous Intersection Management". In: *JAIR* 31.1 (2008), 591–656. DOI: 10.1613/jair.2502.

[2] E. Erdem et al. "A General Formal Framework for Pathfinding Problems with Multiple Agents". In: *Proc. of AAAI*. 2013, 290–296.

[3] J. Yu and S. LaValle. "Planning Optimal Paths for Multiple Robots on Graphs". In: 2012, pp. 3612–3617. DOI: 10.1109/ICRA.2013.6631084.

[4] L. Pallottino et al. "Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems". In: *IEEE Trans, on Rob.* 23.6 (2007), 1170–1183. DOI: 10.1109/TRO.2007.909810.

[5] D. Silver. "Cooperative Pathfinding". In: *Proc. of AIIDE*. 2005, 117–122.

[6] J. Yu and S. M. LaValle. "Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics". In: *IEEE Trans. on Rob.* 32.5 (2016), 1163–1177. DOI: 10.1109/TRO.2016.2593448.

[7] H. Ma and S. Koenig. "AI Buzzwords Explained: Multi-Agent Path Finding (MAPF)". In: *AI Matters* 3 (2017). DOI: 10.1145/3137574.3137579.

[8] D. Goldberg and M. J. Matarić. "Interference as a Tool for Designing and Evaluating Multi-Robot Controllers". In: *Proc. of AAAI*. 1997, 637–642.

[9] M. Schneider-Fontán and M. Mataric. "Territorial multi-robot task division". In: *IEEE Trans. on Rob. and Autom.* 14 (1998), pp. 815–822.

[10] F. Amigoni, J. Banfi, and N. Basilico. "Multirobot Exploration of Communication-Restricted Environments: A Survey". In: *IEEE Intelli. Sys.* 32.6 (2017), pp. 48–57. DOI: 10.1109/MIS.2017.4531226.

[11] N. Rao et al. "Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms". In: 1993.

[12] S. Thrun. "Robotic Mapping: A Survey". In: *Exploring Artificial Intelligence in the New Millennium*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, 1–35.

[13] W. Burgard et al. "Coordinated multi-robot exploration". In: *IEEE Transactions on Robotics* 21.3 (2005), pp. 376–386. DOI: 10.1109/TRO.2004.839232.

[14] K. M. Wurm, C. Stachniss, and W. Burgard. "Coordinated multi-robot exploration using a segmentation of the environment". In: *IROS*. IEEE. 2008, pp. 1160–1165.

[15] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib. "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes". In: *Proc. of AAAI*. Vol. 26. 1. 2012.

[16] A. Felner et al. "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges". In: *SoCS*. 2017, pp. 28–37.

[17] G. Sharon et al. "Conflict-based search for optimal multi-agent pathfinding". In: *Artificial Intelligence* 219 (Feb. 2015), pp. 40–66. DOI: 10.1016/j.artint.2014.11.006.

[18] G. A. Hollinger and S. Singh. "Multirobot Coordination With Periodic Connectivity: Theory and Experiments". In: *IEEE Trans. on Rob.* 28.4 (2012), pp. 967–973. DOI: 10.1109/TRO.2012.2190178.

[19] D. Tateo et al. "Multiagent Connected Path Planning: PSPACE-Completeness and How to Deal With It". In: *Thirty-Second Conference on Artificial Intelligence,* 2018.

[20] T. Charrier et al. "Complexity of planning for connected agents". In: *JAAMAS* 34.2 (2020), p. 44. DOI: 10.1007/s10458-020-09468-5.

[21] C. H. Papadimitriou and M. Yannakakis. "Shortest paths without a map". In: *Theoretical Computer Science* 84.1 (1991), pp. 127–150. DOI: 10.1016/0304-3975(91)90263-2.

[22] A. Itai and H. Shachnai. "Adaptive source routing in high-speed networks". In: *ISTCS*. 1993, pp. 212–221. DOI: 10.1109/ISTCS.1993.253468.

[23] W. Burgard et al. "Collaborative multi-robot exploration". In: *Proc. of ICRA*. Vol. 1. 2000, pp. 476–481. DOI: 10.1109/ROBOT.2000.844100.

[24] L. V. Lita, J. Schulte, and S. Thrun. "A System for Multi-Agent Coordination in Uncertain Environments". In: *Proc. of AGENTS*. 2001, pp. 21–22. DOI: 10.1145/375735.375806.

[25] H. Zhang and Y. Xu. "The k-Canadian Travelers Problem with Communication". In: *FAW-AAIM*. 2011, pp. 17–28. DOI: 10.1007/s10878-012-9503-x.

[26] D. Shiri and F. S. Salman. "On the Online Multi-Agent O—D k-Canadian Traveler Problem". In: *J. of Comb. Opt.* 34.2 (2017), 453–461. DOI: 10.1007/s10878-016-0079-8.

[27] W. J. Savitch. "Relationships Between Nondeterministic and Deterministic Tape Complexities". In: *J. Comput. Syst. Sci.* (1970). DOI: 10.1016/S0022-0000(70)80006-X.

[28] G. Peterson, J. Reif, and S. Azhar. "Lower Bounds for Multiplayer Noncooperative Games of Incomplete Information". In: *Comput & Math. Appl.* 41.7-8 (2001), pp. 957–992. DOI: 10.1016/S0898-1221(00)00333-3.

[29] J. Yu and S. M. LaValle. "Multi-agent Path Planning and Network Flow". In: *Algorithmic Foundations of Robotics X*. 2013, pp. 157–173.

[30] P. Surynek. "An Optimization Variant of Multi-Robot Path Planning is Intractable". In: *Proc. of AAAI*. 2010, 1261–1263.

[31] C. Amato et al. "Decentralized Control of Partially Observable Markov Decision Processes". In: *CDC*. 2013, pp. 2398–2405. DOI: 10.1109/CDC.2013.6760239.

[32] R. Berthon et al. "Strategy logic with imperfect information". In: *LICS*. 2017, pp. 1–12. DOI: 10.1109/LICS.2017.8005136. eprint: 1805.12592.

[33] A. K. Chandra, D. Kozen, and L. J. Stockmeyer. "Alternation". In: *J. of ACM* 28.1 (1981), pp. 114–133. DOI: 10.1145/322234.322243.

[34] T. Charrier et al. "Reachability and Coverage Planning for Connected Agents". In: *Proc. of IJCAI*. 2019, pp. 144–150. DOI: 10.24963/ijcai.2019/21.

[35] D. Lichtenstein. "Planar Formulae and Their Uses". In: *SIAM Journal on Computing (SICOMP)* 11 (1982), pp. 329–343. DOI: 10.1137/0211025.

[36] H. Ma, T. K. S. Kumar, and S. Koenig. "Multi-Agent Path Finding with Delay Probabilities". In: *Proc. of AAAI*. 2017, 3605–3612.

[37] D. Atzmon et al. "k-Robust Multi-Agent Path Finding". In: *International Symposium on Combinatorial Search (SoCS)*. 2017, pp. 157–158.

[38] D. Atzmon et al. "Probabilistic Robust Multi-Agent Path Finding". In: *Proc. of ICAPS*. 2020, pp. 29–37.

[39] M. Ivanová and P. Surynek. "Adversarial Cooperative Path-Finding: Complexity and Algorithms". In: *ICTAI*. 2014, pp. 75–82. DOI: 10.1109/ICTAI.2014.22.

[40] A. Murano, G. Perelli, and S. Rubin. "Multi-agent Path Planning in Known Dynamic Environments". In: *PRIMA*. 2015. DOI: 10.1007/978-3-319-25524-8_14.

[41] J. Miura and Y. Shirai. "Probabilistic Uncertainty Modeling of Obstacle Motion for Robot Motion Planning". In: *Journal of Robotics and Mechatronics* 14 (2002), pp. 349–356.

[42] N. C. Griswold and J. Eem. "Control for mobile robots in the presence of moving objects". In: *IEEE Trans. on Rob. and Autom.* 6.2 (1990), pp. 263–268. DOI: 10.1109/70.54744.

[43] Yun Seok Nam, Bum Hee Lee, and Nak Yong Ko. "A View-Time Based Potential Field Method for Moving Obstacle Avoidance". In: *Proc. of SICE*. 1995, pp. 1463–1468. DOI: 10.1109/SICE.1995.526730.

[44] A. Ulusoy et al. "Robust multi-robot optimal path planning with temporal logic constraints". In: *Proc. of ICRA*. 2012, pp. 4693–4698. DOI: 10.1109/ICRA.2012.6224792.

[45] S. A. Miller, Z. A. Harris, and E. K. P. Chong. "Coordinated Guidance of Autonomous UAVs via Nominal Belief-State Optimization". In: *ACC*. 2009, pp. 2811–2818. DOI: 10.1109/ACC.2009.5159963.

[46] A. Stentz. "Optimal and Efficient Path Planning for Unknown and Dynamic Environments". In: *IJRA* 10 (Feb. 1993).

[47] D. Ferguson, N. Kalra, and A. Stentz. "Replanning with RRTs". In: *Proc. of ICRA*. 2006, pp. 1243–1248. DOI: 10.1109/ROBOT.2006.1641879.

[48] M. Likhachev et al. "Anytime Dynamic A*: An Anytime, Replanning Algorithm". In: *Proc. of ICAPS*. 2005, 262–271.

[49] C. Ferrari et al. "Multirobot motion coordination in space and time". In: *Robotics and Autonomous Systems* 25.3-4 (1998), pp. 219–229. DOI: 10.1016/S0921-8890(98)00051-7.

[50] A. Bry and N. Roy. "Rapidly-exploring Random Belief Trees for motion planning under uncertainty". In: *Proc. of ICRA* (2011), pp. 723–730.

[51] J. P. Gonzalez and A. Stentz. "Planning with uncertainty in position an optimal and efficient planner". In: *IROS*. 2005, pp. 2435–2442. DOI: 10.1109/IROS.2005.1545048.

[52] S. Prentice and N. Roy. "The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance". In: *IJRR* 28 (Oct. 2009), pp. 1448–1465. DOI: 10.1177/0278364909341659.

[53] B. Nebel et al. "Implicitly Coordinated Multi-Agent Path Finding under Destination Uncertainty: Success Guarantees and Computational Complexity". In: *JAIR* 64.1 (2019), 497–527. ISSN: 1076-9757. DOI: 10.1613/jair.1.11376.

[54] K.-H. C. Wang and A. Botea. "Tractable Multi-Agent Path Planning on Grid Maps". In: *Proc. of IJCAI*. 2009, pp. 1870–1875.

[55] J. Banfi, N. Basilico, and F. Amigoni. "Intractability of Time-Optimal Multirobot Path Planning on 2D Grid Graphs with Holes". In: *RA-L* 2.4 (2017), pp. 1941–1947. DOI: 10.1109/LRA.2017.2715406.

Appendix A. Complements on the Imperfect Information Setting

We present here a formal modeling of the imperfect information in our setting.

In our setting, at any time, an agent observes all movement edges adjacent (in- and outcoming edges) to the vertex v it occupies. Moreover, they observe the presence or absence of a communication edge between (v, v') if v' is occupied by another agent with which there is a direct or indirect communication (via other agents). Intuitively, during an execution, at each step, each agent updates their knowledge about the graph with these observations they receive. Moreover, they share all their knowledge with all agents with which they are connected at each step.

Given graph $G = \langle V, E^m, E^c \rangle$, let us define the direct observation $obs_i(c)$ of agent i at a configuration c to be the set:

$$\{o^m_{c_i,v'} \mid (c_i, v') \in E^m\} \cup \{o^{\neg m}_{c_i,v'} \mid (c_i, v') \notin E^m\} \tag{A.1}$$

$$\cup \{o^m_{v',c_i} \mid (v', c_i) \in E^m\} \cup \{o^{\neg m}_{v',c_i} \mid (v', c_i) \notin E^m\} \tag{A.2}$$

$$\cup \{o^c_{c_i,c_j} \mid j \text{ is an agent and } (c_i, c_j) \in E^c\} \tag{A.3}$$

$$\cup \{o^{\neg c}_{c_i,c_j} \mid j \text{ is an agent connected to } i \text{ in } c, (c_i, c_j) \notin E^c\} \tag{A.4}$$

where $o^m_{u,v}$, $o^{\neg m}_{u,v}$, $o^c_{u,v}$ and $o^{\neg c}_{u,v}$ are abstract terms that represent the *observations*. In the definition of $obs_i(c)$, points (1) and (2) mean that agent i directly observes the set of movement edges adjacent to her current position. Point (3) means that agent i observes a communication edge when she can communicate with another agent j. Point (4) means that agent i observes the absence of a communication edge when she sees that she can not communicate directly with another agent j but can communicate with j via multi-hop. That is, we say that j is an agent connected to i in c when there is a communication path $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ with $i_1 = i$ and $i_k = j$.

Let O denote the set of all observations. We define the knowledge of an agent as a subset of O. We define the *initial knowledge for the pair* (G_1, G_2) as follows:

$$\begin{aligned} K^0(G_1, G_2) = \quad & \{o^m_{u,v} \mid (u, v) \in E^m_1\} \cup \{o^c_{u,v} \mid (u, v) \in E^c_1\} \cup \\ & \{o^{\neg m}_{u,v} \mid (u, v) \notin E^m_2\} \cup \{o^{\neg c}_{u,v} \mid (u, v) \notin E^c_2\} \\ & \text{where } G_i = \langle V, E^m_i, E^c_i \rangle. \end{aligned}$$

This corresponds to the *a priori* knowledge on the graph all agents have before making any observation.

During the execution, agents update their knowledge at each step, upon visiting a new configuration. Formally, the knowledge $k_i((K_j)_{1 \le j \le n}, \pi)^G$ of agent i after observing execution π in actual graph G, where each agent j starts with initial knowledge K_j, is defined by induction on π:

- $k_i((K_j)_{1 \le j \le n}, \epsilon)^G = K_i$
- $k_i((K_j)_{1 \le j \le n}, \pi c)^G$ is the union of:

$$k_i((K_j)_{1 \le j \le n}, \pi)^G; \tag{a}$$

$$obs_i(c); \tag{b}$$

$$\bigcup_{j \text{ connected to } i \text{ in } c} (k_j((K_j)_{1 \le j \le n}, \pi)^G \cup obs_j(c). \tag{c}$$

Intuitively, the knowledge of an agent is composed of (a) her knowledge collected until now, (b) her current observation, and (c) the knowledge of the agents she is connected to and their current observation.

When the agents start with an initial knowledge (G_1, G_2) that is clear from the context, we will omit the tuple $(K_j)_{1 \le j \le n}$ and simply write $k_i(\pi)^G$.

Note that during a connected execution, all agents have an identical knowledge at all times. We thus omit the subscript i, and replace the tuple of initial knowledge sets by a single set K and write $k(K,\pi)^G$ rather than $k_i((K_j)_{1\leq j\leq n},\pi)^G$.

Appendix B. Proofs of Section 4

Unbounded Case. We formalize the intuition presented in the proof sketch of Theorem 3. We define the following recursive property: $P(\underline{G},\overline{G},c,c^t)$ holds for graphs $\underline{G},\overline{G}$, and configurations c,c^t if either $\overline{G}$ is not (c^s,c^t)-c-admissible, or there exists a connected execution π from c to c^t in $\overline{G}$ such that for all G with $\underline{G}\subseteq G\subseteq\overline{G}$, writing K_0 for the initial knowledge for the pair $(\underline{G},\overline{G})$, there exists $0\leq i_0\leq|\pi|$ with $k(K_0,c)^G=k(K_0,\pi[0..i_0-1])^G$, and either $\pi[i_0]=c^t$ or $(k(K_0,c)^G\subsetneq K=k(K_0,\pi[0..i_0])^G$ and $P(\underline{G}^K,\overline{G}^K,\pi[i_0],c^t))$.

The following two lemmas prove Theorem 3.

Lemma 3. *An instance $I=(G_1,G_2,c^s,c^t,\infty)$ is c-positive if, and only if $P(G_1,G_2,c^s,c^t)$.*

Proof. We prove the following more general property by induction on the number of edges present in G_2 and absent in G_1, that is, $|E_2^m|-|E_1^m|+|E_2^c|-|E_1^c|$: For all graphs $G_1\subseteq\overline{G}\subseteq G_2$, and configurations c, if the instance $(\underline{G},\overline{G},c,c^t,\infty)$ is c-positive then $P(\underline{G},\overline{G},c,c^t)$.

If $\underline{G}=\overline{G}$ then either the instance is not c-admissible and, then, $P(\underline{G},\overline{G},c,c^t)$ holds, or there is a connected execution in $\overline{G}$ from c to c^t in which case the property holds as well.

Assume $\underline{G}\subsetneq\overline{G}$, and consider σ a witness strategy for the instance $(\underline{G},\overline{G},c,c^t,\infty)$. Consider a graph $\underline{G}\subseteq G\subseteq\overline{G}$. If the execution π induced by σ does not reveal any new observation in G, then it must end in c^t and $P(\underline{G},\overline{G},c,c^t)$ holds. Otherwise, let i_0 be the first step where a new observation is made. Since σ is a witness strategy, it is a witness for the instance $(\underline{G}^K,\overline{G}^K,\pi[i_0],c^t,\infty)$ as well where $K=k(\pi[0..i_0])^G$. Since $\overline{G}^K$ and $\underline{G}^K$ have a smaller number of differences than $\overline{G}$ and $\underline{G}$, we conclude by induction that $P(\underline{G},\overline{G},\pi[i_0],c^t)$. Thus, $P(\underline{G},\overline{G},c,c^t)$ holds.

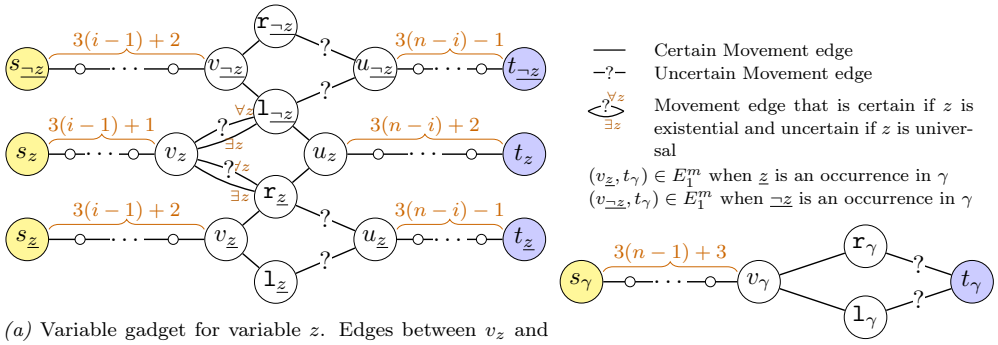
Let us now show that all instances that satisfy the property are c-positive. Assume $P(G_1,G_2,c^s,c^t)$ holds. We define the joint strategy σ on all graphs $\overline{G}^K$ and executions π in $\overline{G}^K$, such that $P(\underline{G}^K,\overline{G}^K,\mathsf{last}(\pi),c^t)$, by induction on the length of π.

Assume σ is constructed for an execution π and knowledge K. If $\overline{G}^K$ is not c-admissible, then any strategy is a witness strategy so σ can be defined arbitrarily. Otherwise, consider the connected execution π' given by $P(\underline{G}^K,\overline{G}^K,\mathsf{last}(\pi),c^t)$. We define σ so that agents follow π' until index i_0, in which case either $\pi'[i_0]=c^t$ or $P(\underline{G}^{K'},\overline{G}^{K'},\pi\pi'[1..i_0],c^t)$. □

Lemma 4. *$P(G_1,G_2,c^s,c^t)$ can be checked in polynomial space.*

Proof. The existence of a connected execution can be checked in polynomial space by Theorem 2. However, the size of such an execution can be exponential, and checking $P(G_1,G_2,c^s,c^t)$ requires iterating over the step of the execution. We thus need to combine the enumeration of the connected execution as we check P recursively.

The procedure to check $P(\underline{G},\overline{G}^K,c,c^t)$ works as follows. We non-deterministically guess a connected execution step by step, from c to c^t using the PSPACE algorithm of Theorem 2. We thus only keep the last configuration in memory, a binary integer counter to bound the length of the execution (bounded by the number of configurations, thus an exponential), and the current graph $\overline{G}^K$. If there is no such execution, we accept. Otherwise, at each step, say, after having visited execution π and generated next configuration c' we enumerate all possible sets $K'=k(K_0,\pi c')^G$, where K_0 is the initial knowledge for the pair (G_1,G_2). This can be done by enumerating all subsets of movement edges adjacent to c', present in $\overline{G}^K$ but not in $\underline{G}^K$, and similarly communication edges revealed by c'. Note that $k(K_0,\pi c')^G$



(a) Variable gadget for variable z. Edges between v_z and $\mathtt{l}_{\underline{\neg z}}$, and between v_z and $\mathtt{r}_{\underline{z}}$ are both certain if z is existential and both uncertain if z is universal.

(b) Clause gadget γ.

Figure 5. Gadgets for the reduction from QBF to the bounded reachability problem in the complete connectivity case.

only depends on $k(K_0, \pi)^G$ and c' which the algorithm already has. There is an exponential number of possibilities, and these can be enumerated in polynomial space. For each case, we check recursively whether $P(\underline{G}^{K'}, \overline{G}^{K'}, c', c^t)$. Since the knowledge can increase only a polynomial number of times (since the knowledge can only increase when an edge is added or removed), the depth of the recursive calls is polynomial. Thus, overall, the procedure uses polynomial space. □

Bounded Case.

Proof of Lemma 1. First, observe that if the existential player has a strategy σ in the alternating algorithm, then the instance is c-positive. In fact, for any graph G with $G_1 \subseteq G \subseteq G_2$, consider strategy τ of the universal player which makes choices according to G. Since this σ wins against τ, either the graph is not admissible or the agents successfully arrive to the target configuration. Conversely, assume that the instance is c-positive, that is, for all choices of an admissible graph G, the agents arrive at a target configuration under some joint strategy σ. We apply σ in the alternating algorithm. Consider any strategy τ of the universal player, and observe the execution induced by (σ, τ). If the graph induced by τ is revealed not to be admissible, then the existential player wins. Otherwise, consider any admissible graph G with $G_1 \subseteq G \subseteq G_2$ compatible with the edges revealed during the execution of (σ, τ). Since σ is winning in the original game when the underlying graph is G, the existential player also wins. □

Proof of Lemma 2. To build a PSPACE algorithm, we perform a DFS of the computation tree T in a succinct manner. Consider the tree T', built from T, where we only keep vertices in which the universal player makes a decision, while paths along which only the existential player moves are shortcut into single edges. The depth of this tree is polynomial by definition.

The idea is to run a DFS on T' to check whether the machine accepts. This can be done in polynomial space provided that the children of all vertices can be computed in polynomial space. Successors of an existential configuration c in T' are computed as follows: we generate on-the-fly all possible configurations c' and test whether c' is reachable from c in the original alternating machine by using a PSPACE oracle. The DFS that runs in PSPACE augmented with this PSPACE oracle gives a polynomial space procedure. □

Proof of Theorem 5. The lower bound is proven by reduction from TQBF.

Consider a QBF φ of the form $\forall z_1 \exists z_2 \dots Q_n z_n \psi$ where ψ is a Boolean formula in conjunctive normal form with n variables and m clauses.

In the reduction, we call a *movement path*, from vertex v to vertex u, a chain of vertices linking v to u by movement edges. In addition, we denote an occurrence of a positive (resp. negative) literal of a variable z, by $\underline{z}$ (resp. $\underline{\neg z}$)

Let us describe the construction of the graph G_1 depicted in Figure 5. For each variable z, we create a gadget depicted in Figure 5a. Formally, for all variables z, the i-th quantified variable, we create the following vertices: s_z, $s_{\underline{z}}$, $s_{\underline{\neg z}}$, v_z, $v_{\underline{z}}$, $v_{\underline{\neg z}}$, u_z, $u_{\underline{z}}$, $u_{\underline{\neg z}}$, t_z, $t_{\underline{z}}$, $t_{\underline{\neg z}}$, $\mathtt{l}_{\underline{z}}$, $\mathtt{l}_{\underline{\neg z}}$, $\mathtt{r}_{\underline{z}}$ and $\mathtt{r}_{\underline{\neg z}}$. We create the following movement paths: between s_z and v_z of length $3(i-1)+1$, between $s_{\underline{z}}$ (resp. $s_{\underline{\neg z}}$) and $v_{\underline{z}}$ (resp. $v_{\underline{\neg z}}$) of length $3(i-1)+2$, between u_z and t_z of length $3(n-i)+2$, and between $u_{\underline{z}}$ (resp. $u_{\underline{\neg z}}$) and $t_{\underline{z}}$ (resp. $t_{\underline{\neg z}}$) of length $3(n-i)$-1. Then, we add the following movement edges: from $u_{\underline{z}}$ (resp. $u_{\underline{\neg z}}$) to $\mathtt{l}_{\underline{z}}$ (resp. $\mathtt{l}_{\underline{\neg z}}$) and $\mathtt{r}_{\underline{z}}$ (resp. $\mathtt{r}_{\underline{\neg z}}$), from u_z to $\mathtt{r}_{\underline{z}}$ and $\mathtt{l}_{\underline{\neg z}}$. Finally, if the variable z is an existential variable then we add movement edges from v_z to $\mathtt{r}_{\underline{z}}$ and $\mathtt{l}_{\underline{\neg z}}$.

We create clause gadgets as depicted in Figure 5b. Formally, for all clauses γ, we create the following vertices: s_γ, v_γ, $\mathtt{l}_\gamma$, $\mathtt{r}_\gamma$ and t_γ. We create a movement path of length $3(n-1)+3$ from s_γ to v_γ. Finally, we create movement edges from v_γ to $\mathtt{l}_\gamma$ and $\mathtt{r}_\gamma$.

Note that G_2 contains G_1 and for all universal variables, it contains additionally the following movement edges: from v_z to $\mathtt{r}_{\underline{z}}$ and $\mathtt{l}_{\underline{\neg z}}$, from $u_{\underline{z}}$ to $\mathtt{r}_{\underline{z}}$ and $\mathtt{l}_{\underline{z}}$, from $u_{\underline{\neg z}}$ to $\mathtt{r}_{\underline{\neg z}}$ and $\mathtt{l}_{\underline{\neg z}}$, and from t_γ to $\mathtt{l}_\gamma$ and $\mathtt{r}_\gamma$.

We define the initial configuration and target configuration as follows:

$$c^s = \langle s_{\gamma_1}, \dots, s_{\gamma_m}, s_{z_1}, \dots, s_{z_n}, s_{\underline{z_1}}, \dots, s_{\underline{z_1}}, \dots, s_{\underline{\neg z_n}} \rangle;$$
$$c^t = \langle t_{\gamma_1}, \dots, t_{\gamma_m}, t_{z_1}, \dots, t_{z_n}, t_{\underline{z_1}}, \dots, t_{\underline{z_1}}, \dots, t_{\underline{\neg z_n}} \rangle.$$

We show that φ is true iff (G_1, G_2, c^s, c^t, k) is c-positive with $k = 3(n-1)+5$.

For simplicity, we classify the agents as follows. (1) A *Variable agent* is an agent starting at a vertex s_z; (2) a *Clause agent* is an agent starting at a vertex s_γ; (3) a *Positive (resp. negative) occurrence agent* is an agent starting at $s_{\underline{z}}$ (resp. $s_{\underline{\neg z}}$).

($\Rightarrow$) Assume that the QBF φ is true. There exists a collection of Skolem functions A such that for each existential variable z_i (where i is even), and an assignment ν to universally quantified variables in $z_1, z_3, \dots, z_{i-1}$, $A_{z_i}(\nu) \in \{\top, \bot\}$ is the value assigned to z_i such that φ is true under assignment ν augmented with the values of A. We construct the following strategy σ, which guarantees that c^t is reached in k steps from c^s.

Intuitively, the lengths of the initial movement paths are designed so that agents starting at s_{z_i} arrive at v_{z_i} in the order of their indices. For an existential variable agent, the choice of the successor from v_{z_i} determines the value of z_i; while for a universal variable agent, the choice is made by the environment. More precisely, if the agent moves to $\mathtt{r}_{\underline{z_i}}$, then z_i is set to true, if she moves to $\mathtt{l}_{\underline{\neg z_i}}$ it is set to false.

Formally, all agents start by moving to their respective v vertices (e.g. A clause agent at s_γ moves to v_γ). They arrive at these vertices at different moments due to the sizes of their movement paths. An existential variable agent arrives at vertex v_{z_i} at time $3(i-1)+1$. At this point, all universal variable agents among $z_1, z_3, \dots, z_{i-1}$ have arrived to their respective vertices v_{z_j}, thus revealing the values of these variables. Let ν be this assignment. If $A_{z_i}(\nu) = \top$, the agent moves to $\mathtt{r}_{\underline{z_i}}$, and otherwise, to $\mathtt{l}_{\underline{\neg z_i}}$.

When a universal variable agent arrives to v_{z_i} at time $3(i-1)+1$, she follows the only edge dictated by the environment, either to $\mathtt{r}_{\underline{z_i}}$ or to $\mathtt{l}_{\underline{\neg z_i}}$. This assigns $\top$ to the variable z_i in the former case, and $\bot$ in the latter case. Observe that if the graph is admissible, then there must exist a path from source to target for each agent; this means that one of these edges must be present.

Consider a positive (resp. negative) occurrence agent associated to a clause γ. This agent arrives to $v_{\underline{z_i}}$ (resp. $v_{\underline{\neg z_i}}$) at time $3(i-1)+2$. Observe that the variable agent has already determined the value of z_i in the previous step.

• if z_i is assigned $\top$ (resp. $\bot$) then the agent moves to t_γ, observe which edges are available at t_γ, and immediately comes back to $v_{\underline{z_i}}$ (resp. $v_{\underline{\neg z_i}}$). Now, the edge between $\mathtt{r}_{\underline{z_i}}$ (resp. $\mathtt{l}_{\underline{\neg z_i}}$) and $u_{\underline{z_i}}$ (resp. $u_{\underline{\neg z_i}}$) has been observed by agent z_i; so if this edge is present, she moves to $\mathtt{r}_{\underline{z_i}}$ and $u_{\underline{z_i}}$; and if not, then the edge from $\mathtt{l}_{\underline{z_i}}$ to $u_{\underline{z_i}}$ must be available, and she arrives to $t_{\underline{z_i}}$ (resp. $t_{\underline{\neg z_i}}$) at time k.

• if z_i is assigned $\bot$ (resp. $\top$) then she does not visit t_γ, but moves to $\mathtt{l}_{\underline{z_i}}$ (resp. $\mathtt{l}_{\underline{\neg z_i}}$). If the edge between $\mathtt{l}_{\underline{z_i}}$ (resp. $\mathtt{l}_{\underline{\neg z_i}}$) and $u_{\underline{z_i}}$ (resp. $u_{\underline{\neg z_i}}$) is available, she moves to $t_{\underline{z_i}}$ (resp. $t_{\underline{\neg z_i}}$), otherwise she moves back and reach $t_{\underline{z_i}}$ (resp. $t_{\underline{\neg z_i}}$) at time k, through $\mathtt{r}_{\underline{z_i}}$ (resp. $\mathtt{r}_{\underline{\neg z_i}}$).

It remains to argue that clause agents can reach their target vertices within k steps. Since φ is true, by the definition of A, whatever the choice for the universal variables, some literal ℓ of each clause γ is assigned to true. Therefore, the positive or negative occurrence agent corresponding to this literal visits t_γ, thus revealing the edges available from t_γ to $\mathtt{r}_\gamma$ and $\mathtt{l}_\gamma$. Note that at least one of these edges must be available for the graph to be admissible. Thus, a clause agent arriving to v_γ at time $3(n-1)+3$ can follow the available path to reach t_γ exactly at time k.

($\Leftarrow$) Let σ be a witness joint strategy. Following σ, each clause agent c must know the available edges in the rest of their paths at time $3(n-1)+3$ since otherwise they cannot ensure reaching t_γ at time k. Thus, for each clause γ, the vertex t_γ is visited by some occurrence agent under strategy σ. Furthermore, an occurrence agent $\underline{z}$ (resp. $\underline{\neg z}$) can visit vertex t_γ and still make it to $t_{\underline{z}}$ (resp. $t_{\underline{\neg z}}$) in time iff the associated variable agent has observed the presence of the edge between $u_{\underline{z}}$ (resp. $u_{\underline{\neg z}}$) and $\mathtt{r}_{\underline{z}}$ (resp. $\mathtt{l}_{\underline{\neg z}}$) beforehand. In fact, otherwise, if the occurrence agent makes a wrong guess between $\mathtt{l}_{\underline{z}}$ and $\mathtt{r}_{\underline{z}}$, they will not arrive to $t_{\underline{z}}$ (resp. $t_{\underline{\neg z}}$) at time k. Hence, the joint strategy of the variable agents determines an assignment function which satisfies φ.

□

Appendix C. Proofs of Section 5

Unbounded Case. Given φ, let $(G_1, G_2, c^s, c^t, \infty)$ denote the graph built in the proof sketch of Theorem 6.

Lemma 5. *DQBF φ holds if and only if $(G_1, G_2, c^s, c^t, \infty)$ is positive.*

Proof. We denote the agents as (1) an agent that starts at s_z as the *existential agent* z if z is an existential variable; the agent is called *universal* if z is universal; (2) the *verification agent* is the one starting at s_γ; (3) the *clause agents* γ_i start at s_{γ_i}; (4) the *observation agents* O start at s_O.

($\Rightarrow$) Suppose the DQBF φ holds, and let A be the collection of Skolem functions. We build the following joint strategy. The environment chooses the truth values of universal variables z by deleting some edges v_z to $\top_z$ or v_z to $\bot_z$. If the environment deletes the edge v_z to $\top_z$, the agent is forced to pass in $\bot_z$, thus the variable z is considered to be false. If the environment deletes the edge v_z to $\bot_z$, the agent is forced to pass in $\top_z$, thus variable z is considered to be true. If the environment deletes neither edge, then we define the strategy for agent a_y to choose to pass in y, making y true by default.

The rest of the strategy is defined as follows. At the first step, each variable agent for variable z moves to v_z, and each observation agent for the pair (x, y) moves to v_y, and thus observes the value of universal variable y. At the second step, existential agents remain in place, while observation agents move to v_O, thus sharing their observations with the

corresponding existential agents. Thus, at this point, each existential agent corresponding to variable x knows the values of all universal variables $y \in O_x$. Then, agent z moves to $\top_z$ if $A_z(\nu) = 1$ and to $\bot_z$ otherwise, where ν is the valuation of the variables in O_z.

All clause agents move from s_i to γ_i and remain at γ_i for two steps so that all existential and universal variable agents z are at $\top_z$ or $\bot_z$. The verification moves to v_γ and also waits for two steps. Since each clause is satisfied by the currently read valuation, each clause agent γ_i communicates at least with one existential or universal agent. Thus, the verification agent communicates with *all* clause agents via these variable agents. Since the clause agents communicate with the verification agent at this moment, the latter can see which edges are present in the clause gadget, and can continue go to t_γ without getting stuck.

($\Leftarrow$) Conversely, suppose there is a witness joint strategy, in particular ensuring that the verification agent goes to t_z. This means that the agent must have received all the information about the topology around vertices $\gamma_1, \ldots, \gamma_k$. But this is only possible if the agents have occupied a configuration in which the verification agent is at v_γ, all clause agents are at γ_i such that for each clause γ_i, there is at least one variable agent z at $\top_z$ if $z \in \gamma_i$ and at $\bot_z$ if $\neg z \in \gamma_i$. Thus, φ is a positive instance of TDQBF. □

Bounded Case.

Proof of Theorem 7. The upper bound when the bound k is given in unary is obtained by the following non-deterministic algorithm:

(1) Guess a strategy σ_i for each agent i, up to executions of length $\leq k$. Such a strategy can be represented as a tree of depth k, and thus has size exponential in k.

(2) Check that σ_i is uniform for agent i.

(3) For all admissible graphs G such that $G_1 \subseteq G \subseteq G_2$, execute the joint strategy σ and check that the outcome execution from the initial configuration leads to the target configuration.

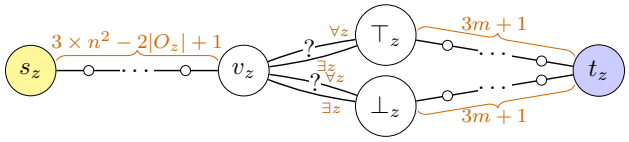
The obtained algorithm is non-deterministic and runs in exponential time. Note that the encoding of k is not relevant since for $k \geq 2|V|$ there is always a solution following the unbounded case.

We now prove the NEXPTIME-hardness result by reduction from TDQBF. Given an instance of TDQBF $\forall y_1, \ldots, y_n \exists x_1(O_{x_1}) \ldots \exists x_n(O_{x_n})\ \psi$, we build an instance of bounded decentralized reachability (G_1, G_2, c^s, c^t, k). We denote the number of clauses by m.

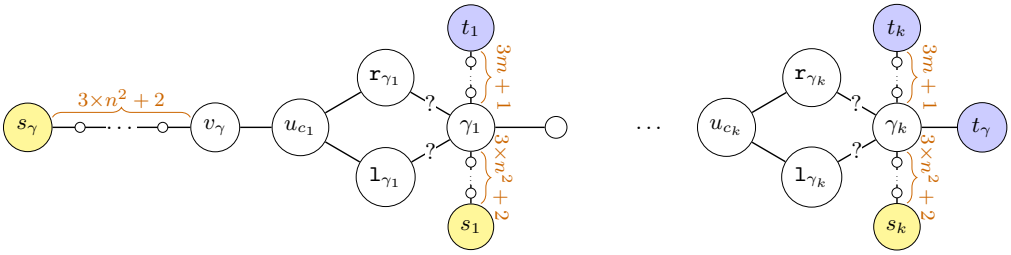
We construct the graph G_1 as follows:

For each variable z, we create a gadget, as depicted in Figure 6a. We create the vertices s_z, v_z, $\top_z$, $\bot_z$ and t_z, and we create a movement path of length $3m + 1$ from t_z to $\top_z$ and $\bot_z$. Then, if the variable z is universal, we create a movement path of length $3 \times n^2 + 1$ from s_z to v_z. If the variable z is x_i, that is the i-th existential variable, we build a movement path of length $3 \times n^2 - 2|O_z| + 1$ from s_z to v_z as follows. We first build a movement path of length $3 \times (in - |O_z|)$. We then extend this movement by a vertex ρ_y for each $y \in O_z$. We extend our path from the last such vertex ρ_y to v_z by a movement path of length $3 \times (n^2 - in)$. Furthermore, we add a bidirectional edge between ρ_y and v_y.

We create the clause gadget, depicted in Figure 6b, composed of the vertices s_γ, v_γ, t_γ and for all clauses γ_i, the vertices u_{γ_i}, $\mathtt{l}_{\gamma_i}$, $\mathtt{r}_{\gamma_i}$, s_i, c_i and t_i. We create a movement path of length $3 \times n^2 + 2$ between s_γ and v_γ and between all s_i and γ_i. For all γ_i, we create movement edges from u_{γ_i} to $\mathtt{l}_{\gamma_i}$ and to $\mathtt{r}_{\gamma_i}$, from the vertex γ_i to $u_{\gamma_{i+1}}$, or t_γ if $i = m$. In addition, we add a movement edge between v_γ and u_{γ_1}. For all γ_i, we add a movement path of length $3m + 1$ between vertices γ_i and t_i. Vertex γ_i communicates with $\top_z$ iff $z \in \gamma_i$, and with $\bot_z$ iff $\neg z \in \gamma_i$. Moreover, vertex v_γ communicates with all $\top_z$ and $\bot_z$ for all variables z.



(a) Variable gadget for variable z. Edges between v_z and $\top_z$, and between v_z and $\bot_z$ are both certain if z is existential and both uncertain if z is universal.



(b) Clauses gadget.

Certain Undirected Movement edge

Uncertain Undirected Movement edge

Undirected Movement edge that is certain if z is existential and uncertain if z is universal

$(\top_z, \gamma_i) \in E_1^c$ when z appears in γ_i
$(\bot_z, \gamma_i) \in E_1^c$ when $\neg z$ appears in γ_i
$(\top_z, v_\gamma) \in E_1^c, (\bot_z, v_\gamma) \in E_1^c$

Figure 6. Gadgets for the reduction from DQBF to the bounded decentralized reachability problem.

Graph G_2 contains G_1 and for all universal variables z, we add movement edges: from v_z to $\top_z$ and to $\bot_z$, and for all clauses γ_i, we add movement edges from the vertex γ_i to $\mathtt{l}_{\gamma_i}$ and to $\mathtt{r}_{\gamma_i}$.

We define the initial and target configurations as $c^s = \langle s_\gamma, s_{\gamma_1}, \ldots, s_{\gamma_m}, s_{z_1}, \ldots, s_{z_n} \rangle$ and $c^t = \langle t_\gamma, t_{\gamma_1}, \ldots, t_{\gamma_m}, t_{z_1}, \ldots, t_{z_n} \rangle$.

We show that φ is a positive instance of TDQBF iff (G_1, G_2, c^s, c^t, k) is positive with $k = 3 \times n^2 + 2 + 3m + 1$. We refer to an agent that starts at s_z as the *existential agent* z if z is an existential variable; the agent is called *universal* if z is universal. The *verification agent* is the agent that starts at s_γ and the *clause agents* γ_i start at s_{γ_i}.

($\Rightarrow$) Suppose the DQBF holds, and let A be an assignment function. We build a joint strategy. The environment chooses the truth values of variables z by deleting some edges v_z to $\top_z$ or v_z to $\bot_z$. If the environment deletes the edge v_z to $\top_z$, it enforces the agent to pass in $\bot_z$, thus the variable z is considered to be false. If the environment deletes the edge v_z to $\bot_z$, it enforces the agent to pass in $\top_z$, thus variable z is considered to be true. If the environment deletes neither edge, then we define the strategy for agent a_y to choose to pass in y, making y true by default.

The strategy is defined as follows. Each existential agent z follows the movement path of length $3 \times n^2 - 2|O_z| + 1$ to v_z, but whenever they have vertex v_y as a neighbor, they visit v_y, come back, and continue their paths. This happens exactly O_z times, so at time $3 \times n^2 + 1$, the existential agent is at u_z. Note that along this path the agent has visited all vertices $v_{z'}$ with $z' \in O_z$, thus knows which edge among $(v_{z'}, \top_{z'})$ and $(v_{z'}, \bot_{z'})$ is present. Thus, upon arriving to v_z, the agent has the knowledge of the valuation for all variables in O_z. Observe also that by construction of the movement paths between s_z and v_z, the

agents never meet in this phase of the execution. Agent z moves to $\top_z$ if $A_z(\nu) = 1$ and to $\bot_z$ otherwise, where ν is the valuation of the variables in O_z.

All clause agents reach γ_i at time $3 \times n^2 + 2$. Moreover, the verification agent is at v_γ at this point, and all existential and universal variable agents z are at $\top_z$ or $\bot_z$. Recall that all these vertices communicate. Since each clause is satisfied by the currently read valuation, each clause agent γ_i communicates at least with one existential or universal agent. Thus, the verification agent communicates with *all* clause agents via these variable agents. Since the clause agents can see which edges are present in the clause gadget, the verification agent has now full information about this gadget, and can continue their path until t_γ without backtracking, thus in total time $3 \times n^2 + 2 + 3m + 1$.

($\Leftarrow$) Conversely, suppose there is a joint strategy enforcing that the verification agent goes to t_γ in k steps. Thus, it means that she must have received all the information about the surroundings of the vertices $\gamma_1, \ldots, \gamma_k$, as she has no time to backtrack from a wrong choice. This information can only be sent to the verification agent from the clause agents through the variable agents after $3 \times n^2 + 2$ steps. The variable agents representing the assignments are connecting (i.e. satisfying) all clauses. Indeed, for all γ_i, there is one variable agent z at $\top_z$ if $z \in \gamma_i$ and at $\bot_z$ if $\neg z \in \gamma_i$. Thus, the DQBF is a positive instance of TDQBF. □