The 28th International Conference on Automated Planning and Scheduling
June 24 – 29, 2018, Delft, The Netherlands

Proceedings of the ICAPS 2018
Doctoral Consortium
Mentoring Program

Dissertation Abstract: Deep Probabilistic Planning

Thiago P. Bueno
Department of Computer Science
Institute of Mathematics and Statistics
University of São Paulo, Brazil

Abstract

Deep Probabilistic Planning combines gradient-based optimization in deep neural nets with ideas from probabilistic planning in order to efficiently solve continuous sequential decision-making problems in discrete-time, dynamic domains. The basic approach is to explicitly encode a policy as a deep reactive net and to formulate a probabilistic planning problem as an optimization task defined over a suitable stochastic computation graph that shall be solved by gradient-based policy search methods, such as gradient ascent algorithms. In this PhD project, we propose: (i) to extend previous works on planning through back-propagation to the more general case of stochastic transitions; and (ii) to investigate the applicability of model-based ideas of probabilistic planning (e.g., heuristic search, re-planning, reachability analysis) to improve convergence properties. We shall focus on domains exhibiting nonlinear dynamics, which state-of-the-art planners struggle to solve due to computational issues or representational limitations.

Introduction

Planning is one of the oldest disciplines in Artificial Intelligence (AI). It is concerned with domain-independent methods for generating agents that exhibit intelligent decision-making behavior. The main feature that distinguishes the area of planning from other related fields is that the planner has direct access to a precise model of how actions induce state transitions and observations provide information about the current situation of a given environment. In other words, planning is the *model-based* approach to the problem of autonomous behavior in dynamic systems.

When the model allows actions to have uncertain outcomes, we are dealing with a *probabilistic planning* problem. Markov Decision Process (MDP) (Puterman 2014) is the de facto mathematical formalism for modeling such sequential decision-making problems under uncertainty. However, it is known that optimal algorithms for MDPs are intractable in the worst case (Littman, Dean, and Kaelbling 1995) when the model is represented in compact form. This complexity result is sometimes stated informally as the *curse of dimensionality*. To this end, there has been a growing interest in alternative approaches leveraging *Monte-Carlo sampling* (Keller and Eyerich 2012) or *re-planning* techniques (Yoon, Fern, and Givan 2007), as observed in recent ICAPS competitions.

Although these state-of-the-art planners are able to solve large complex MDPs, they are usually constrained to domains with discrete action spaces due to limitations on how to deal with infinite branching factors. Additionally, when the model's transition dynamics are high-dimensional or nonlinear, even specially-designed extensions for continuous actions (Weinstein and Littman 2012) have scalability issues. Hence, if one is concerned with large sequential decision-making under uncertainty in continuous discrete-time environments, different methods have to be considered.

Inspired by recent advances in Deep Learning (DL) (LeCun, Bengio, and Hinton 2015; Goodfellow et al. 2016), we propose to investigate the opportunities and limitations of formulating continuous probabilistic planning problems as *gradient-based optimization* tasks. DL methods has recently achieved remarkable success (Krizhevsky, Sutskever, and Hinton 2012; Sutskever, Vinyals, and Le 2014) in a variety of non-convex optimization tasks using *deep neural nets* (Schmidhuber 2015). Particularly, when it comes to decision-making, Deep Reinforcement Learning (Deep RL) (Silver et al. 2016) leverages these neural models as parameterized approximators of value functions and/or policies in order to derive locally-optimal behaviors for agents interacting with dynamic environments.

Policy-gradient methods (Buffet, Aberdeen, and others 2007; Silver et al. 2014), one of the most researched approaches in Deep RL, have been shown to successfully solve large, high-dimensional continuous control problems. The overall approach is grounded in the observation that it is possible to estimate the gradient of the policy's performance w.r.t. its parameters without explicit knowledge of the system's transition and to apply these estimates to improve the policy over time given sufficient amount of sampled data.

However, a natural question arises when formal knowledge of the system is available: *can policy-gradient methods be effectively applied in a model-based setting?* This PhD project aims at positively answering this question when policy-gradient is combined with ideas from the literature of probabilistic planning, especially those derived from heuristic search, re-planning and reachability analysis.

This dissertation abstract is organized as follows. First, we present background knowledge on continuous Markov Decision Processes and stochastic computation graphs. Afterwards, we develop the main ideas of our proposed approach on Deep Probabilistic Planning. Finally, we conclude with remarks on future work and opportunities.

Background

Continuous Markov Decision Processes

We consider sequential decision-making problems in which an agent is supposed to interact with a discrete-time, stochastic environment modeled by *continuous state-action Markov Decision Processes* (CSA-MDPs) defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \mathcal{H}, \mathbf{s}_0)$, in which $\mathcal{S} \subseteq \mathbb{R}^m$ is the state space, $\mathcal{A} \subseteq \mathbb{R}^n$ is the action space, $\mathcal{T}\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the Markovian transition function given by the conditional probability density $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ over next states $\mathbf{s}'$ given the current state $\mathbf{s}$ and action $\mathbf{a}$, $R\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function that specifies the immediate return received in the current state $\mathbf{s}$ given action $\mathbf{a}$, $\mathcal{H} = 0, 1, ..., H$ is the finite set of decision stages, and $\mathbf{s}_0$ is a start state. A deterministic policy is given by a function $\pi\colon \mathcal{S} \times \mathcal{H} \to \mathcal{A}$ that prescribes a single action to each state in a given decision stage. We define a state-value function V^π as the expected total reward received by following policy π, i.e., $V^\pi(\mathbf{s}) = \mathbb{E}\left[\sum_{t=1}^{H} R(\mathbf{s}_t, \pi(\mathbf{s}_t)) \,\middle|\, \mathbf{s}\right]$. The agent's objective is to find a policy π^* maximizing $V^{\pi^*}(\mathbf{s}_0)$.

We will consider as a running example throughout the text, a stochastic extension of the *Navigation* domain (Faulwasser and Findeisen 2009) in which an agent is supposed to get to a goal position from a start position as fast as possible while avoiding deceleration zones.

$$\lambda = \prod_j \frac{2}{1 + \exp(-\eta_j ||\mathbf{s}_t - \mathbf{c}_j||)} - 1 \tag{1}$$

$$\mathbf{p}_{t+1} \sim \mathcal{N}(\mu = \mathbf{s}_t + \lambda \mathbf{a}_t, \sigma = \frac{\sigma_{\max}}{\sqrt{2}} ||\mathbf{a}_t||) \tag{2}$$

$$\mathbf{s}_{t+1} = \max(\mathbf{l}, \min(\mathbf{u}, \mathbf{p}_{t+1})) \tag{3}$$

$$R = -||\mathbf{s}_t - \mathbf{g}|| \tag{4}$$

Figure 1: Navigation: en example of a continuous MDP

Figure 1 shows the domain specification. Each deceleration zone j is characterized by its center position $\mathbf{c}_j$ and decay constant η_j, and its effect depend upon the euclidean distance between the its center and agent's position. The joint deceleration factor λ is given by the multiplicative effect of each independent deceleration zone. Scalars $\mathbf{l}$ and $\mathbf{u}$ denote the lower and upper bounds of the grid. Figure 2 illustrates an example of nonlinear navigation domain.

Stochastic Computation Graphs

Stochastic computation graphs (Schulman et al. 2015) are a formalism used to specify models that mix deterministic computations with random variables drawn from distributions whose parameters depend on the results of previous computations. Its purpose is to define all dependencies between the variables of an acyclic generative model in order to allow the development of efficient sampling and automatic differentiation algorithms.

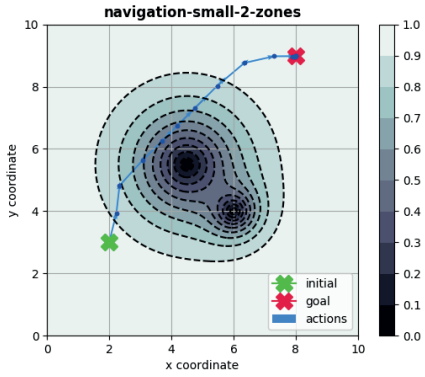


Figure 2: Navigation with nonlinear deceleration zones

Formally, a *stochastic computation graph* $\mathcal{G} = (V, E)$ is a directed, acyclic graph defined over three disjoint set of nodes: (i) *input nodes* I, directly observed; (ii) *deterministic nodes* D, corresponding to functions of its parent nodes; and (iii) *stochastic nodes* S, representing random variables conditionally distributed accordingly to a function whose parameters depend on its parent nodes. The set of nodes V is partitioned as $V = I \cup D \cup S$. An edge $(u, v) \in E$ if node v or its probability distribution depends on node u. Let v and w be nodes of a stochastic computation graph $\mathcal{G}$. Then, we denote by $v \prec w$ the property that it exists a dependency path from node v to node w in $\mathcal{G}$. Additionally, we denote by $v \prec^D w$ the property that it exists a dependency path from node v to node w traversing only deterministic nodes.

The *objective function* of a stochastic computation graph is given by $J = \mathbb{E}[\sum \hat{c}]$, where $\hat{c}$ are leaf nodes of interest known as *cost nodes*. Input nodes are root nodes typically used to define the parameters we would like to differentiate with respect to. Note that this objective function is given by an expectation, therefore we cannot directly use error backpropagation algorithms to optimize it in the general case.

The *gradient of the objective function* J of a stochastic computation graph w.r.t. input parameter θ (Schulman et al. 2015) is given by (provided all derivatives exist):

$$\nabla_\theta J = \mathbb{E}_{w \in S}\left[\sum_{\substack{w \in S, \\ \theta \prec^D w}} \nabla_\theta (\log p(w|\textsc{Pa}(w))) \hat{Q}_w + \nabla_\theta \sum_{\substack{\hat{c} \in C, \\ \theta \prec^D \hat{c}}} \hat{c}\right], \tag{5}$$

where $\hat{Q}_w = \sum_{\hat{c} \in C, w \prec \hat{c}} \hat{c}$ is the *cost-to-go* (also known as *downstream total cost*) from node w and $\textsc{Pa}(w)$ is the set of parents nodes of node w.

Deep Probabilistic Planning

We propose to formulate a probabilistic planning task as an optimization problem defined over a deep recurrent model in order to perform stochastic local search in policy space. In particular, we shall extend the recently proposed framework of planning through back-propagation (Wu, Say, and Sanner 2017) to the more general case of stochastic transitions. The core idea is to explicitly represent a deterministic policy π_θ by a parameterized approximation function whose parameters shall be optimized through *stochastic gradient ascent* over a suitable loss function $J(\theta)$:

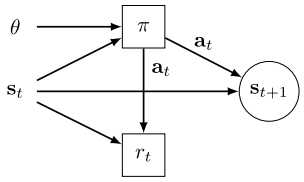
$$\theta \leftarrow \theta + \alpha \nabla_\theta J, \tag{6}$$

where hyper-parameter α is the *learning rate*.

For this matter, a deep neural model shall be used to encode a smooth, differentiable reactive policy that extracts relevant features of the state in order to prescribe nearly-optimal actions in a continuous space. We call this the *policy network*. Additionally, as a natural initial choice, we shall consider the value function of the start state as a loss function, i.e., $J(\theta) = V^{\pi_\theta}(\mathbf{s}_0)$.

Markov Recurrent Model

We propose a formal model inspired on Recurrent Neural Nets (Sutskever, Vinyals, and Le 2014), i.e., the *Markov Recurrent Models* (MRMs), in order to: (i) perform batch trajectory sampling; and (ii) to specify the symbolic dependencies between states, actions, and rewards which are necessary to leverage automatic differentiation.



$$\mathbf{a}_t = \pi_\theta(\mathbf{s}_t)$$
$$\mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)$$
$$r_t = R(\mathbf{s}_t, \mathbf{a}_t)$$

Figure 3: Markov Recurrent Model with parameterized policy: current state $\mathbf{s}_t$ and parameter θ are the cell inputs, next state $\mathbf{s}_{t+1}$ and reward r_t are the cell outputs, and policy π_θ is represented by a deep neural net function approximator sharing parameters across time-steps.

Planning via Gradient-Based Optimization

Following the theory of gradient estimation in stochastic computation graphs (Schulman et al. 2015), we derive the exact gradient of $J(\theta)$ in a similar way of model-free reinforcement learning policy-gradient algorithms (Silver et al. 2014). The *policy-gradient* of the Markov Recurrent Model is given by:

$$\nabla_\theta J = \mathbb{E}\left[\sum_{t=0}^{H} \nabla_\theta (\log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t))\, \hat{Q}_{t+1} + \nabla_\theta r_t\right] \tag{7}$$

where $\hat{Q}_{t+1} = \sum_{k=t+1}^{H} R(\mathbf{s}_k, \pi_\theta(\mathbf{s}_k))$ is the *reward-to-go* from state $\mathbf{s}_{t+1}$ to the end of the roll-out, provided the transition and reward functions are differentiable.

The overall approach of probabilistic planning via optimization is outlined in Algorithm 1. Trajectory sampling can be efficiently carried out by parallelized tensor-based implementations (i.e., TensorFlow (Abadi et al. 2016)). This is usually done by exploiting the *re-parameterization trick* which allows to sample a random variable $\mathbf{z} \sim p(\cdot|\mathbf{x})$ by transforming it into a deterministic function $\mathbf{z} = \phi(\mathbf{x}, \xi)$ and using an independent marginal density to sample an auxiliary random variable $\xi \sim p(\cdot)$. For the location-scale family of distributions the function $\phi(\mathbf{x}, \xi)$ is given by $\phi(\mathbf{x}, \xi) = \mu(\mathbf{x}) + \sigma(\mathbf{x})\xi$, where $\mu(\cdot)$ and $\sigma(\cdot)$ are functions defining the location and scale of probability density $p(\cdot|\mathbf{x})$.

Algorithm 1: Planning via Stochastic Optimization

for epoch $= 1, \cdots, M$ **do**
  for $i = 1, \cdots, N$ **do** ▷ batch
    Initialize $\tau^{\mathbf{i}}$ as empty list ▷ trajectory
    for $t = 0, \cdots, T-1$ **do**
      $\mathbf{a}_t \leftarrow \pi_\theta(\mathbf{s}_t)$ ▷ policy network
      Sample $\mathbf{s}_{t+1}$ from $p(\cdot|\mathbf{s}_t, \mathbf{a}_t)$
      $r_t \leftarrow R(\mathbf{s}_t, \mathbf{a}_t)$
      Append $(t, \mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ to $\tau^{\mathbf{i}}$
    $\nabla_\theta J^i \leftarrow 0$ ▷ gradients
    foreach $(t, \mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ in $\tau^{\mathbf{i}}$ **do**
      $\hat{Q}_{t+1} \leftarrow \sum_{k=t+1}^{T-1} r_k$ ▷ reward-to-go
      $\nabla_\theta^{(t)} J \leftarrow \nabla_\theta [\log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\, \hat{Q}_{t+1} + r_t]$
      $\nabla_\theta J^i \leftarrow \nabla_\theta J^i + \nabla_\theta^{(t)} J$
  $\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta J^i$ ▷ gradient ascent
return π_θ

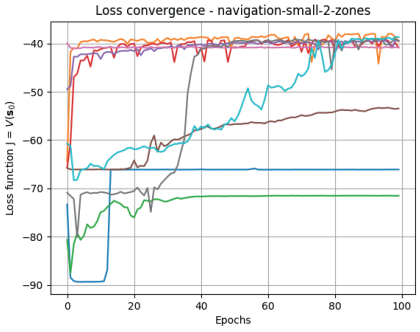


Figure 4: Loss convergence as a function of optimization epochs for the Navigation problem: each line corresponds to a separate optimization run

Variance Reduction through Informed Baselines

It has been shown that although unbiased, policy-gradient methods usually suffer from high variance (Greensmith, Bartlett, and Baxter 2004). Figure 4 illustrates this limitation for the Navigation problem of Figure 2. To alleviate

this problem, most methods use baselines (Weaver and Tao 2001) (also known as control variate in optimal control) that do not add bias, but considerably reduce the variance of the gradient estimator.

In general, baseline functions are purely statistical in the sense that they do not explicitly exploit any knowledge of the underlying model. However, we shall propose investigating the use of the solution of a relaxed MDP $\mathcal{M}_{\text{DET}}$ obtained through determinization techniques as a baseline. A possible research direction is to follow a similar approach of the FF-Replan (Yoon, Fern, and Givan 2007), which derives the MDP $\mathcal{M}_{\text{DET}}$ by only considering the effects of actions with highest probabilities (i.e, *most-likely determinization*).

Assuming that the probability density transition function of the original MDP is part of a location-scale family of distributions, we can use the re-parametrization trick to control the determinization. Therefore, by controlling the noise ξ we can induce different levels of determinization, and in the limit $\xi \to 0$ we obtain the most-likely determinization.

Once we got $\mathcal{M}_{\text{DET}}$ we can leverage a recently proposed approach based on planning through back-propagation (Wu, Say, and Sanner 2017) to obtain a locally-optimal plan cost $c_{\text{DET}}(s_0) = \sum_{t=0}^{H} \bar{r}_t$ for the initial state $\mathbf{s}_0$ and use it as a baseline $b(\mathbf{s}_0) = c_{\text{DET}}(s_0)$. Finally, we substitute the advantage function $\hat{A}_{i,t+1} = \hat{Q}_{i,t+1} - b_{t+1}(\mathbf{s}_0)$ in Equation 7 in place of the reward-to-go $\hat{Q}_{i,t+1}$ to hopefully reduce the variance of the gradient estimator.

Conclusion

This research proposal combines several ideas from a number of complementary areas in AI (e.g., probabilistic planning, statistical gradient-based optimization, and deep neural nets from machine learning). Although there has been some cross-fertilization between these research communities in recent years, few works have especially focused on bringing together advances of deep learning to model-based decision-making methods. Conversely, a wealth of techniques typically studied in the planning community (i.e., heuristic search, determinization, reachability analysis) has yet to bear fruits inside more data-driven approaches such as gradient-based optimization in terms of better data efficiency and state exploration. Therefore, we strongly believe that more work on Deep Probabilistic Planning can have a positively great impact on the planning community.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Buffet, O.; Aberdeen, D.; et al. 2007. FF+ FPG: Guiding a Policy-Gradient Planner. In *ICAPS*, 42–48.

Faulwasser, T., and Findeisen, R. 2009. Nonlinear model predictive path following control. *Nonlinear Model Predictive Control* 384:335–343.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Greensmith, E.; Bartlett, P. L.; and Baxter, J. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research* 5(Nov):1471–1530.

Keller, T., and Eyerich, P. 2012. PROST: Probabilistic Planning Based on UCT. In *ICAPS*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *nature* 521(7553):436.

Littman, M. L.; Dean, T. L.; and Kaelbling, L. P. 1995. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 394–402. Morgan Kaufmann Publishers Inc.

Puterman, M. L. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Schmidhuber, J. 2015. Deep learning in neural networks: An overview. *Neural networks* 61:85–117.

Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, 3528–3536.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 387–395.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529(7587):484–489.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Weaver, L., and Tao, N. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 538–545. Morgan Kaufmann Publishers Inc.

Weinstein, A., and Littman, M. L. 2012. Bandit-Based Planning and Learning in Continuous-Action Markov Decision Processes. In *ICAPS*.

Wu, G.; Say, B.; and Sanner, S. 2017. Scalable Planning with Tensorflow for Hybrid Nonlinear Domains. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 6276–6286.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In *ICAPS*, volume 7, 352–359.

Action Schema Splitting (ICAPS 2018 - Doctoral Consortium & Mentoring Program)

Bustos Facundo
UNC - FAMAF
Cordoba, Argentina
facundojosebustos@gmail.com

Abstract

As modeling details can have a large impact on planner performance, domain compilation has been a traditional subject of interest in the planning community not only between languages, but also within languages. In this dissertation, I will talk about intra-language compilation method that has as yet been applied only manually, and that has never been formally described: *Action Schema Splitting*, which transforms an action schema with a large interface (many parameters) into several schemas with shorter interfaces, exponentially reducing the number of ground actions. We spell out this method, characterizing exactly the choice of splits preserving equivalence to the original schema. Making that choice involves a trade-off between interface size and plan length, which we explore by designing automatic domain compilation method. Our experiments show that this technique may be very usefull mainly on those domains where planner preprocessing fails.

Introduction

Automatic domain compilation has been a topic of interest for a long time. The general idea behind of any domain compilation is transform a planning task into a simpler or more suitable model easier to deal with. For example, it has been considered to remove redundant actions (Haslum and Jonsson 2000) in order to reduce the branching factor or inversely add additional redundant macro-actions in order to reduce distance-to-goal (Botea et al. 2005; Newton et al. 2007), with the objective of improve the planner's performance. In this work we propose an intra-language compilation that has not, as yet, been systematically investigated, which consists basically in: given an action schema $a[X]$, i. e., a PDDL-like action with parameters (variables) X ranging over objects, create several sub-action schemas $a_1[X_1], \ldots, a_k[X_k]$ whose combination corresponds exactly to $a[X]$ in any valid plan. The key advantage of such split is the reduction in the number of ground actions. For example, if each $x \in X$ can be instantiated with 100 objects, $|X| = 3$, and $|X_i| = 1$, then we reduce that number from 1000000 to 300.

We spell out formally what a valid action schema split is, devising a general compilation method. Specifically, we show that, given a schema $a[X]$, one can choose any split $a_1[X_1], \ldots, a_k[X_k]$ that preserves the intended order among potentially identical preconditions/adds/deletes in the original schema (e. g., preconditions need to be checked before corresponding deletes are applied, or else the split schema may not be applicable even though the original schema is). Choosing $a_1[X_1], \ldots, a_k[X_k]$ constitutes a trade-off between minimizing interface size $\max_i |X_i|$ and thus the number of ground actions, vs. minimizing *split size* k and, therewith, plan length. We design automatic domain compilation techniques addressing that trade-off.

Valid Splits

One of the most important properties of our technique is that, the compilation was designed such that the plans in a compiled planning task are in one-to-one correspondence with those in the original task. To ensure this, we need to guarantee: 1) *Correctness:* every plan of the compiled task is a split of a plan of the orignal task, (i. e., we don't add plans). 2) *Completeness:* for all plan of the original task, its split is a plan of the compiled task, (i. e.we don't delete plans). When a split satisfies both properties, we say that it is valid. To achive valid splitting, the following issues must be tackled:

1. Nothing ensures that the two sub-schemas are *instantiated consistently*, i. e., assign the same object to the shared parameters.
2. Nothing ensures that the sub-schemas of the same action are executed *en-block*, i. e., all together and without any other subactions from other action inserted in between.
3. Nothing ensures that preconditions be checked before corresponding identical deletes or add effects are applied. Also nothing ensures that deletes effects be applied before corresponding identical add effects is applied.

Issues (1) and (2) are easy to fix, for arbitrary splits, by decorating the sub-schemas with new atoms ensuring consistent instantiation and en-block execution. Issue (3) is more subtle, and is the only one restricting the set of splits we can choose from.

Example 1 *As an illustrating running example, we will consider the action schema moving block x from block y to block z. We can write this in STRIPS notation as:*

Move(x, y, z)
pre : $\{on(x, y), clear(x), clear(z)\}$
add : $\{on(x, z), clear(y)\}$
del : $\{on(x, y), clear(z)\}$

A tentative split into sub-schemas could be:

$\textbf{\textit{Move}}_1(x,y)$	$\textbf{\textit{Move}}_2(x,z)$
pre : $\{on(x,y), clear(x)\}$	pre : $\{clear(z)\}$
add : $\{clear(y)\}$	add : $\{on(x,z)\}$
del : $\{on(x,y)\}$	del : $\{clear(z)\}$

The correspondence of this split schema to the original one appears obvious, and one may be tempted to conclude that action schema splitting is trivial. However, note that the split shown is not actually valid due: the shared parameter x may be instantiated to different objects in both sub-actions. Also, add $clear(y)$ of $Move_1(x,y)$ is applied before precondition $clear(z)$ of $Move_2(x,z)$, then if y and z are instantiated with the same object, then, in any reachable state s, the original schema will not be applicable because we cannot have $on(x,y)$ and $clear(y)$ at the same time, while in the split schema, however, the add of $Move_1(x,y)$ will establish that atom, rendering $Move_2(x,z)$ (and therewith the overall split schema) applicable.

Characterizing Valid Splits

We have now identified exactly which splits can be chosen (namely, the valid ones). Remains the question, how to actually find such splits? Does there even always exist a non-trivial split, with more than a single sub-schema? Both questions are easily answered; we start with the latter one:

Definition 1 *Let $a[X]$ be an action schema. Then its Trivial Split, denoted $TrivialSplit(a[X])$, is the one that assigns every atom to the same single sub-schema. Its Atom Split denoted $AtomSplit(a[X])$, is the one that assigns every atom to a separate sub-schema.*

Obviously, $TrivialSplit(a[X])$ and $AtomSplit(a[X])$ are both valid for any action schema $a[X]$. Towards answering the question how to find more general valid splits, note first that splits naturally form a hierarchy: the unique coarsest split is the Trivial Split, and the unique *finest* split (i. e., the least coarse one) is the Atom Split. We can travel between these two extremes by iteratively merging sub-schemas. Iterating such merging steps, starting from the Atom Split, underlies our search methods for valid splits. This is suitable because:

Theorem 1 *Let $a[X]$ be an action schema. Then any split constructed by starting with $AtomSplit(a[X])$, and iteratively merging mergeable sub-schemas, is valid. Vice versa, any valid split can be constructed in this way.*

Computing Valid Splits

We still need to design automatic methods for applying that machinery automatically: *How to find good splits automatically? And what are "good splits" anyhow?* As we move up and down in the hierarchy for an action schema $a[X]$, coarser splits have less sub-schemas and therefore tend to result in shorter plans using the split domain; and finer splits have smaller interfaces and therefore tend to result in less ground actions. We capture this in terms of the split's *size*, $SplitSize(a_1[X_1], \ldots, a_k[X_k]) = k$, and *interface size*, $IntSize(a_1[X_1], \ldots, a_k[X_k])) = \max_i |X_i|$. Plan length increases linearly in $SplitSize$ (if the underlying action indeed participates in the plan), and the number of ground actions decreases exponentially in $|X| - IntSize$. The Trivial Split is optimal in split size, the Atom Split is optimal in interface size. In practice, we need to find a good trade-off between these two extremes. Unsurprisingly, doing so optimally is hard:

Theorem 2 *Let Split Optimization be the problem of deciding, given an action schema $a[X]$ as well as natural numbers K and N, whether there exists a valid split $a_1[X_1], \ldots, a_k[X_k]$ such that $SplitSize(a_1[X_1], \ldots, a_k[X_k]) \leq K$ and $IntSize(a_1[X_1], \ldots, a_k[X_k]) \leq N$. Then split optimization is **NP**-complete.*

We approximate that optimization problem through of *hill-climbing* in the split hierarchy, starting at the finest split and moving to coarser ones. In detail, we instantiate hill-climbing as follows:

- **Start node:** $s = AtomSplit(a[X])$.
- **Successor function:** $SuccFn(a_1[X_1], \ldots, a_k[X_k]) = \{a'_1[X'_1] \ldots, a'_{k-1}[X'_{k-1}] \mid a'_1[X'_1] \ldots, a'_{k-1}[X'_{k-1}]$ is a merged valid split of $a_1[X_1], \ldots, a_k[X_k]\}$.
- **Evaluation function:** $f(a_1[X_1], \ldots, a_k[X_k]) = TradeOff(a_1[X_1], \ldots, a_k[X_k])$.
- **Termination condition:** $s = TrivialSplit(a[X])$.
- **Returned Value:** the expanded node with the smallest $TradeOff$ function.

Hill-climbing thus iteratively generates all splits obtained by merging a mergeable pair of sub-schemas, selecting one with the best trade-off. On the other hand, we capture the trade-off in terms of a weighted sum, normalizing each criterion to the interval $[0, 1]$ to enhance comparability, thus we define $TradeOff(a_1[X_1], \ldots, a_k[X_k])$ as

$$\gamma \frac{k}{N} + (1 - \gamma) \frac{\max_i |X_i|)}{|X|}$$

where $N = SplitSize(AtomSplit(a[X]))$ and the parameter $\gamma \in [0, 1]$ controls the trade-off.

A few words are in order regarding the extreme cases $\gamma = 1$ (all weight on split size) and $\gamma = 0$ (all weight on interface size). With $\gamma = 1$, the search will end up returning $TrivialSplit(a[X])$ (implying that it makes no sense to run them with $\gamma = 1$). With $\gamma = 0$, in contrast, the searches become very conservative, exploring only nodes with optimal interface size equalling that of $AtomSplit(a[X])$, these searches thus attempt to find smaller splits with optimal interface size.

Decorating Valid Splits

We have now clarified how to tackle issue (3), but we have not yet done anything about issue (1), ensuring consistent parameter instantiation across the split schema, nor about issue (2), ensuring en-block execution across all split schemas in the domain. As advertized, both issues are easy to address by introducing artificial (new) atoms implementing a token system between sub-actions.

Example 2 *Consider the action schema* $\boldsymbol{Move}(x, y, z)$ *again. Using the following valid split and* $id(\boldsymbol{Move}(x, y, z)) = 1$ *as well as* $id(x) = 1$ *and* $id(y) = 2$*, we obtain the following decorated valid split:*

$\boldsymbol{Move}_1(x, y)$
  pre : $\{on(x, y), clear(x), \boldsymbol{none}\}$
  add : $\{\boldsymbol{do}_2^1, \boldsymbol{arg}^1(x), \boldsymbol{arg}^2(y)\}$
  del : $\{on(x, y), \boldsymbol{none}\}$
$\boldsymbol{Move}_2(x, z)$
  pre : $\{clear(z), \boldsymbol{do}_2^1, \boldsymbol{arg}^1(x)\}$
  add : $\{on(x, z), \boldsymbol{do}_3^1\}$
  del : $\{clear(z), \boldsymbol{do}_2^1, \boldsymbol{arg}^1(x)\}$
$\boldsymbol{Move}_3(y)$
  pre : $\{\boldsymbol{do}_3^1, \boldsymbol{arg}^2(y)\}$
  add : $\{clear(y), \boldsymbol{none}\}$
  del : $\{\boldsymbol{do}_3^1, \boldsymbol{arg}^2(y)\}$

Tokens **none** and **do** together ensure that the sub-schemas of an original schema $a[X]$ can only be executed en block, i. e., grouped together. Thanks to **none**, no other block can be active when we start with $a_1[X_1]$, and we only release the block when we end with $a_k[X_k]$. As the **do** token is ID'ed, no sub-actions from any other action schema can be executed in between. Token **arg** forces the planner to instantiate the sub-schemas consistently, by fixing the instantiation of every shared parameter x in the first sub-schema using x. The token is ID'ed with the variable in question, as otherwise the roles of two shared variables could be exchanged (if x and y are instantiated to o_1 respectively o_2 up front, then the roles of non-ID'ed instantiated tokens $\textbf{arg}(o_1)$ and $\textbf{arg}(o_2)$ could be changed later on, e. g. using o_2 for x and o_1 for y). Finally, all three issues are solved.

Testing Conclusions

Our technique was implemented as a stand-alone tool in C, starting from existing PDDL parser of FF planner system (Hoffmann and Nebel 2001). The source code is available at the GitHub repository `https://github.com/facubus/ActionSchemaSplitting.git`. A major question in the testing is *which domains to run*. We, of course, did run the IPC domains. However, it is a well known fact that these domains are engineered to challenge search, not pre-processes. Almost all action schemas in IPC domains have small interfaces, and there is not much to gain by Action Schema Splitting. More generally, most benchmarks were created having in mind to test search capabilities, not to test pre-processing capabilities. Our focus domains thus are ones whose action schemas have large interfaces relatively and in which we can create tasks (through generators) with important number of objects per task. Below, we list the most important observations raised from our experiments:

- The technique is suitable on those domains where the preprocessing runs out of memory capacity of the planner system due to the number of objects scales considerably, in contrast, the technique is not suitable for IPC domains.
- Finer splits (less sub-actions) are preferable over coarser splits one (more sub-actions), because the distance to the goal is preponderant at the time of the heuristic search. Then, we recommend using γ values closer to one than zero.
- Also, we recommend dividing as few action schemas as possible as long as you get into planner memory. To help to decide: which ones? we define a notion called *Splitability* which tries to characterize those schemas that have the greatest need being optimized over other ones. Thus, the schemas with highest splitability coefficient have priority to be splitted over the remain.

In general, we have systematized and automated prior works on action schema splitting, as a pre-process to standard planners that ground out the actions. The method shows promise on domains with large interfaces on tasks with preprocessing troubles indicating that it could be a useful tool for, especially, applications without planning expertise who wish to apply planning technology but are not intimately familiar with it.

References

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research* 24:581–621.

Haslum, P., and Jonsson, P. 2000. Planning with reduced operator sets. In Chien, S.; Kambhampati, R.; and Knoblock, C., eds., *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS-00)*, 150–158. Breckenridge, CO: AAAI Press, Menlo Park.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Newton, M. A. H.; Levine, J.; Fox, M.; and Long, D. 2007. Learning macro-actions for arbitrary planners and domains. In Boddy, M.; Fox, M.; and Thiebaux, S., eds., *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, 256–263. Providence, Rhode Island, USA: Morgan Kaufmann.

Relaxed Decision Diagrams for Discrete Optimization Problems

Margarita P. Castro
Department of Mechanical & Industrial Engineering
University of Toronto, Toronto, Ontario, Canada mpcastro@mie.utoronto.ca

1 Introduction

Discrete optimization is one of the main research fields in Operational Research and Computer Science due to its challenging problems and many applications. Several techniques have been used to solve these problems, such as Integer Programming, Constraints Programming and heuristic search. Relaxed Decision Diagrams (DDs) are one of the newest approaches to tackle discrete optimization problems, which use a graphical structure to represent the set of feasible solutions and compute bounds.

A decision diagram is a pervasive data structure for representing Boolean functions (Bryant, 1992). A *relaxed* DD, first introduced by Andersen et al. (2007), is a diagram of limited size that approximates the set of solutions to a discrete problem. It has been largely applied to mathematical programming and discrete optimization, in particular for obtaining optimization bounds (Hoda, Van Hoeve, and Hooker, 2010; Bergman et al., 2016b). A survey on the use of relaxed DDs for optimization is presented by Bergman et al. (2016a).

Recent work has shown how relaxed DD can be used to solve general discrete optimization problem (Bergman et al., 2016b) by using the graphical encoding of the problem as a bounding mechanism in a branch-and-bound tree search procedure. In particular, the technique has shown promising results in scheduling and vehicle routing problems (Cire and van Hoeve, 2013; Kinable, Cire, and van Hoeve, 2017; Castro, Cire, and Beck, 2018).

This thesis aims to extend and explore the usage of relaxed DDs to solve discrete optimization problems. Specifically, we want to uncover the advantages of the approach, its relationship to existing techniques, and its uses in conjunction with other approaches, such as Integer Programming. To this end, the thesis is divided into two main projects that show the power and flexibility of relaxed DDs and their relationship to existing methods.

The first project (Castro, Cire, and Beck, 2018) explores the use of relaxed DDs to solve a challenging vehicle routing problem. We propose an exact approach that uses a discrete relaxation based on *multivalued decision diagrams* (MDDs) to better represent the combinatorial structure of the problem. We enhance our relaxation by using the MDDs as a subproblem to a Lagrangian relaxation technique, leading to significant improvements both in bound quality and run time performance.

The second project focuses on solving cost-optimal classical planning via relaxed DDs. We explore the uses of a relaxed DD as an admissible heuristic embedded on an A^* search procedure. Moreover, we study the relationship with other techniques in the literature, such as relaxed planning graph heuristics (Haslum and Geffner, 2000).

2 Relaxed DDs for Vehicle Routing

This work investigates a new exact approach for the *one-to-one multi-commodity pickup and delivery traveling salesman problem* (m-PDTSP), a variant of the classical traveling salesman problem that incorporates the delivery of a fixed set of commodities by a capacitated vehicle. Specifically, the problem is defined over a directed graph $\mathcal{G}$, where nodes represent locations and arcs are associated with non-negative travel costs. We are also given a set of commodities, each having a weight, a pickup location, and a delivery location. A solution to the m-PDTSP is a minimum-cost Hamiltonian tour on $\mathcal{G}$ where each commodity's pickup location must be visited prior to its corresponding delivery location, and the total weight carried by the vehicle never exceeds its capacity.

The m-PDTSP was introduced by Hernández-Pérez and Salazar-González (2009) and can be viewed as an important subproblem in vehicle routing applications. The authors propose a multi-commodity flow and a path-based formulation for the m-PDTSP, both solved using a Benders decomposition approach. The state-of-the-art solution methods for the m-PDTSP are branch-and-cut algorithms proposed by Gouveia and Ruthmair (2015). Despite the significant solution time improvements, these branch-and-cut algorithms report instances with 21 locations and 10 commodities left unsolved within a reasonable amount of time.

We propose a novel exact approach for the m-PDTSP that applies Lagrangian duality to combine a linear and a discrete relaxation of the problem. In particular, the discrete relaxation is encoded as a relaxed MDD, a graphical structure that compactly represents a set of solutions to a problem. In this work, we leverage the underlying network representation of the problem to better exploit the combinatorial structure of the m-PDTSP while also incorporating dual informa-

tion from a linear programming relaxation of the problem.

This work presents structural results and strategies for constructing relaxed MDDs that take into account both tour constraints and vehicle capacities, extending previous work on MDDs for sequencing problems (Cire and van Hoeve, 2013). Specifically, our capacity-based construction guarantees the satisfaction of the vehicle capacity constraint for all solutions represented in the relaxed MDD. Our second key contribution is a Lagrangian technique that significantly strengthens the existing bounds for the m-PDTSP based on the concepts introduced by Bergman, Cire, and van Hoeve (2015). Namely, we incorporate Lagrange multipliers that penalize solutions encoded by the relaxed MDD which violate constraints of the problem.

The resulting approach provides a flexible relaxation that yields bounds on the optimal solution value of the m-PDTSP and can be embedded, e.g., in any branching search. Computational experiments using a constraint programming solver indicate that the resulting MDD relaxation can enhance solution time by orders of magnitude in a number of instances, in particular when capacities are small. We also find provably optimal solutions to 33 instances in the literature for the first time, 27 of those with our best relaxed MDD variant.

3 Relaxed DDs for Classical Planning problems

Cost-optimal classical planning problems are defined over a set of propositions describing the world, an initial state, a set of goals and a set of actions. Each action is uniquely defined by its set of preconditions, effects, and its associated cost. The problem asks for a minimal cost plan (i.e, a sequence of actions) that, starting from the initial state, reaches a state where all the goals are achieved.

This well-studied problem in the AI literature is commonly solved using heuristic search based planners that use admissible heuristics in an A^* search procedure, e.g., fast-downward (Helmert, 2006). Notably, several of these admissible heuristics rely on a graphical structure that encodes a relaxed version of the problem, e.g., abstractions (Edelkamp, 2001; Helmert et al., 2007), red-black planning trees (Katz, Hoffmann, and Domshlak, 2013; Katz and Hoffmann, 2013) and the LM-cut heuristics (Helmert and Domshlak, 2009).

Our work (Castro et al., 2018) proposes a new admissible heuristic for cost-optimal classical planning based on a relaxed representation of the state transition graph. Specifically, our approach approximates the state transition tree using relaxed DDs. States are encoded as nodes and applicable actions as edge labels. Due to the exponential size of the state-space, our approach merges nodes representing different states, i.e., it over-approximates the reachable space.

We propose a top-down algorithm to construct the relaxed DD. The resulting structure is a layered acyclic graph in which the root node represents the initial state and the leaf nodes correspond to goal states. Our approach uses propositional landmarks (Karpas and Domshlak, 2009) for our node merging heuristic. In addition, we develop several filtering rules to identify edges associated with non-optimal plans.

Once constructed, the relaxed DD is used to compute an admissible heuristic based on the shortest path from the initial state to a node representing a goal state. We use a labeling algorithm in which we compute the shortest path to reach each proposition represented in a node.

Furthermore, this work explores the relationship of our relaxed DD heuristic with several well-known heuristics in the literature. As a first step, we theoretically compare our approach to the h^{max} heuristics (Haslum and Geffner, 2000) and to abstractions (Helmert et al., 2007).

With regard to empirical performance, we have preliminary results over a subset of IPC domains with strictly positive action cost. The results show encouraging performance in terms of heuristic quality and number of nodes expanded. However, run time performance is still far from current state-of-the-art heuristics, such as the operator counting heuristic (Pommerening et al., 2014).

Future works include studying the relationship with specific abstraction heuristics (e.g., pattern databases (Edelkamp, 2001)), and heuristics based on red-black planning trees (Katz, Hoffmann, and Domshlak, 2013). Moreover, we want to improve our current heuristic implementation to reach a competitive level.

4 Conclusion

This thesis explores the use of relaxed DDs as a relaxation mechanism to solve discrete optimization problems. We focus our attention on challenging problems from the operation research and artificial intelligence literature. Specifically, we tackle a capacitated pickup-and-delivery vehicle routing problem and address the cost-optimal classical planning problems.

References

Andersen, H. R.; Hadzic, T.; Hooker, J. N.; and Tiedemann, P. 2007. A constraint store based on multivalued decision diagrams. In *Principles and Practice of Constraint Programming–CP 2007*. Springer. 118–132.

Bergman, D.; Cire, A. A.; van Hoeve, W.-J.; and Hooker, J. 2016a. *Decision Diagrams for Optimization*. Springer International Publishing, 1 edition.

Bergman, D.; Cire, A. A.; van Hoeve, W.-J.; and Hooker, J. N. 2016b. Discrete optimization with decision diagrams. *INFORMS Journal on Computing* 28(1):47–66.

Bergman, D.; Cire, A. A.; and van Hoeve, W.-J. 2015. Lagrangian bounds from decision diagrams. *Constraints* 20(3):346–361.

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

Castro, M. P.; Piacentini, C.; Cire, A. A.; and Beck, J. C. 2018. Relaxed decision diagrams for cost-optimal classical planning. In *HSDIP*, In Press.

Castro, M. P.; Cire, A. A.; and Beck, J. C. 2018. An mdd-based lagrangian approach to the multi-commodity pickup-and-delivery tsp. *INFORMS Journal of Computing (Under Review)*.

Cire, A. A., and van Hoeve, W.-J. 2013. Multivalued decision diagrams for sequencing problems. *Operations Research* 61(6):1411–1428.

Edelkamp, S. 2001. Planning with pattern databases. In *Sixth European Conference on Planning*.

Gouveia, L., and Ruthmair, M. 2015. Load-dependent and precedence-based models for pickup and delivery problems. *Computers & Operations Research* 63:56–71.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *Proceedings of the 5th Internat. Conf. of AI Planning Systems (AIPS 2000)*, 140–149.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Nineteenth International Conference on Automated Planning and Scheduling*.

Helmert, M.; Haslum, P.; Hoffmann, J.; et al. 2007. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, 176–183.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Intell. Res.(JAIR)* 26:191–246.

Hernández-Pérez, H., and Salazar-González, J.-J. 2009. The multi-commodity one-to-one pickup-and-delivery traveling salesman problem. *European Journal of Operational Research* 196(3):987–995.

Hoda, S.; Van Hoeve, W.-J.; and Hooker, J. N. 2010. A systematic approach to MDD-based constraint programming. In *Principles and Practice of Constraint Programming–CP 2010*. Springer. 266–280.

Karpas, E., and Domshlak, C. 2009. Cost-optimal planning with landmarks. In *IJCAI*, 1728–1733.

Katz, M., and Hoffmann, J. 2013. Red-black relaxed plan heuristics reloaded. In *Sixth Annual Symposium on Combinatorial Search*.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013. Red-black relaxed plan heuristics. In *AAAI*.

Kinable, J.; Cire, A. A.; and van Hoeve, W.-J. 2017. Hybrid optimization methods for time-dependent sequencing problems. *European Journal of Operational Research* 259(3):887 – 897.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based heuristics for cost-optimal planning. In *ICAPS*.

Robot Planning for Activity Recognition

Jean Massardi
Supervised by **Éric Beaudry**
Department of Computer Science
Université du Québec à Montréal

Abstract

In order to interact with humans during their Activities for Daily Living, assistant robots need to perform activity and intent recognition. Actual sensors are limited, therefore assistant robots need to find the best observation spot possible. This dissertation presents my thesis proposal to address the problem of ADL recognition with a robot in an open space.

Introduction

In most western countries, aging of the population has become an important healthcare concern. There is an increasing need to develop new solutions to help elderly people to stay at home with more autonomy to improve their quality of life and reduce pressure on caregivers (Pollack 2005). These issues have been addressed in two domains of artificial intelligence : smart homes environment (Pollack 2005), (Rashidi et al. 2011) and assistant robots (Cesta and Pecora 2005), (Graf et al. 2009).

In both cases, the development of quick and reliable activity recognition techniques is mandatory to understand the observed human actions. As of today, usual activity recognition techniques based on supervised learning are unable to address Activities for Daily Living (ADL) recognition as they are countless, making it impossible to have training sets for all of them (Pirsiavash and Ramanan 2012). Unsupervised learning techniques based on object affordance have shown good results for ADL. These techniques aim to identify activities by looking at the interaction of a subject and an object rather than focusing directly on observing a mouvement (Wu et al. 2007). These techniques are now mostly used in smart homes systems because it is possible to place sensors in strategic places to detect an interaction between the subject and an object (Pirsiavash and Ramanan 2012).

Object affordance methods are not as effective when integrated on a robot assistant without any external sensors, due to natural blindspots in an open space. In order to propose this kind of method, we need to address several issues : (1) develop an object interaction detection method, (2) give the robot a plan to maximize the quality of the activity recognition while assisting the person, (3) develop an intent recognition algorithm robust to noisy observation.

In most of the work on activity recognition, the experiments are done in ideal conditions. Usually the system is in front of the observed person, within an optimal distance for the sensors. This kind of conditions don't naturaly occur in an open world, where a robot perform. The question is, given a spatio-temporal context and key habits of a subject, how to strategically place the robot to increase the probability of observing a certain activity and improve the quality of intent recognition. There is an other constraint : the robot must still be able to assist the subject while performing the observations.

Problem description

Each given sensor has an optimal observation spot, which can usually be defined by the distance and the angle of observation of a scene. If the scene is static, it is easy to define the optimal observation spot. In most cases that have no constraints, the optimal observation spot is trivial to find, and if the scene has constraints, some well-know techniques can be used, for example spatial reasoning (Renz 2002).

If the scene is dynamic, especially if the subject is moving, it is more complicated to find these optimal spots. In the case of smart homes, the problem can be addressed by placing multiple sensors at strategic spots. If all the sensors are on the same moving platform, like an assistant robot, the system will have to move to observe the subject. Since the robot has to move according to a specific objective, the problem can be seen as a planning problem.

Two new constraints appear in the case of ADL recognition :

1. There is a potential infinite number of possible activities. However, some activities have a certain probability of occurrence depending on the contexts, both spatio-temporal and previous activities.
2. The exact duration of an activity is unknown. An ap-

proximate duration can be found using probabilistic models, but with a high uncertainty.

This corresponds to a planning problem with temporal uncertainty in an open world.

This first constraint corresponds to a plan recognition task. Plan recognition is the opposite task of planning. Based on observation, plan recognition aim to recognize the goal, the intent or the next action of a given agent. Furthermore, as there is a plan recognition part in the problem described above, the global problem of optimal positioning in ADL recognition for a mobile robot can be seen as a loop interaction problem between plan recognition and planning (Freedman and Fukunaga 2015).

On a formal way the problem can be descibed as the following :

A plan P can be described as a set of couples (S, D), S an identified spatial location and D an uncertain duration. There are two sets of locations, S_A the set of activity most probable locations and S_R the set of optimal observation spots. There is a function $F : S_A * S_R \rightarrow [0; 1]$ which given two locations indicates the observability of an activity. The objective is to find a plan P_R which maximize the observability $O = \sum f$, given a set of probabilistic partially ordered plans P_A while minimizing $card(P_R)$, the number of robot actions. f is the result of the fonction $F(S_A i, S_R j)$ when the activity occurs at $S_A i$ and the robot is at $S_R j$ at the same time.

Before solving this planning problem, three elements have to be found : (1) the set S_A, (2) the set S_R and (3) the set of plans P_A.

Related Work

At the best of our knowledge, there is no integrated solution for daily living activities recognition for assistant robot that has no external sensors. Currently, one of the most advanced systems is the assistant robot ACCOMPANY (Jenkins and Draper 2015), however this system is pretty limited as it needs external sensors and has no intelligent placement algorithm. Another project for robot assistant, Giraff, uses only external sensor to perform activity recognition and activity monitoring (Palumbo et al. 2014). Finding an integrated solution would require to solve four intermediate problems, which are:

1. recognize any given activity in an unsupervised manner;
2. create a behaviour model for an observed subject;
3. plan the path of the robot with uncertainty constraints ;
4. close the interaction loop between activity recognition, plan recognition and planning;

all of these intermediate problems have already been addressed separately.

As previously mentioned, ADL recognition techniques are based on object affordance. One efficient way to do that is to identify the hands of the observed subject and recognize the objects that they interact with. Pirsiavash used this technique with a first person camera and obtained interesting results (Pirsiavash and Ramanan 2012).

In smart homes context, data mining is now widely used to extract behaviour models. Several data mining algorithms have already been tested, like Prefixspan or BIDE. More recently, frequent episode mining techniques like SPEED, obtained precise behaviour models (Rashidi et al. 2011). Stochastical Relational Learning (SRL) techniques are also an interesting option that have not been tested in this particular context yet, two recent studies realized in other contexts are especially interesting. BLP and MLN are already used in plan recognition (Raghavan and Mooney 2011) and problog is already used to create behaviour models (De Raedt, Kimmig, and Toivonen 2007).

The interaction loop between Activity recognition, plan recognition and planning has been identified as an important problem for the domain. There is some interesting perspective using the work of Ramirez,as he proposed to solve plan recognition problems by using classical planners (Ramırez and Geffner 2009) and PO-MDPs (Ramırez and Geffner 2011). In 2017, Freedman proposed an algorithm using this approach in order to close this intercation loop (Freedman and Zilberstein 2017) using only classical planners.

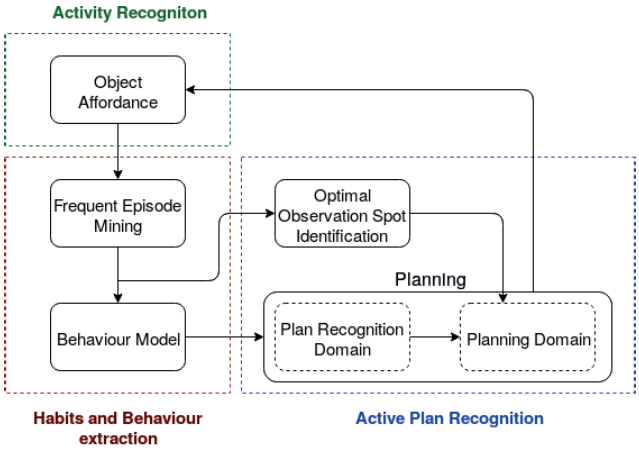
The general problem of planning under uncertainty of task duration is considered by the community as a challenging yet interesting problem. Several approaches have been developped to deal with this uncertainty, such as using MDP related algorithms(Weld and others 2005). Other approaches exist for planning with uncertainty, like using forward chaining search combined with bayesian network (Beaudry, Kabanza, and Michaud 2010). In case of partial observability, continuous PO-MDP are currently the main approach, even considering their low performance (Bai et al. 2010).

Thesis proposal

In order to address the problem of activity recognition for ADL with a robot in an open space, and the problem of planning of activity recognition knowing some behaviours and context, we propose the following architecture.

Activity recognition based on object affordance

Activity recognition is based on an object interaction detection module, which detects the beginning and the end of an interaction between the observed subject and an identified object, keeping their time and space coordinates in memory. The interactions are saved in memory as a transaction $tr_i =< o_i, s/e, t_i, c_i >$ with o_i the identified object, s/e if it is the interaction starting or ending time, t_i the date of the transaction and c_i the coordinates of the transaction.



In order to observe human-object and object-object interactions, we have a approach similar to Pirsiavash (Pirsiavash and Ramanan 2012) and Koppula (Koppula and Saxena 2016), using RGB-D sensors on the robot to track the hands of the observed subject and identify objects in direct contact with them. In our approach, we will detect the hands using the capabilities of the RGB-D sensor, then we will extract images of the hands and their close environment in order to identify any object in interaction with them using modern object recognition techniques.(Krizhevsky, Sutskever, and Hinton 2012) This will permit modeling both direct and indirect human-object interaction for a activity.

Habits and behaviour extraction

We want to extract behaviours and plans from the transactions produced by the activity recognition module. To do so, we propose to use the UP-Span algorithm (Wu et al. 2013). This algorithm gives good results on complex transaction mining by using an utility function. Another advantage of UP-Span is that this algorithm is from the Span family, therefore it can be extended to extract frequent episodes from data streams.

To model the observed subject behaviour, we propose to use a PO-MDP. PO-MDPs have shown good results in both plan recognition and behaviour modelling (Ramırez and Geffner 2011).

Active activity and plan recognition

We decided to solve the plan recognition problem by describing it as a PO-MDP domain and express the solution as a policy. In this domain, both the current state of the observee and his goal are hidden states, while the estimated goal of the observer prediction is a known state. The current state of the observee evolves independently of the observer and can be observed through the observer actions. If the estimated goal differs from the hidden goal, the observer loses rewards.

This modeling has several advantages: (1) it allows the problem to use classical PO-MDP planners to solve it. In our integration, it is currently resolved with SARSOP (Kurniawati, Hsu, and Lee 2008);(2) since it is expressed in the form of a domain, it can be integrated with other domains in order to close the interaction loop between plan recognition and planning;(3) the usage of a PO-MDP allow to favor highly informative observations over less informative ones,thus enabling to actively plan the robot observations.

The next steps are to integrate this in a continous and performant PO-MDP solver, which might require the development of a new solver, as the actual ones underperform in comparison to online discrete-states ones, in order to enhance the plan recognition with the robot planning domain.

In order to enhance performance in continous space, it will be necessary to use spatial reasoning to define best observation spots for the robot before the planning.

Methodology

To assess the performance of the system, for both speed and accuracy, each of the three modules described in this dissertation will be tested separately before being implemented on a real robot all together. The first module, the object interaction detection, will be tested by comparing its result with other activity recognition techniques proposed for robots. The second module, will be tested by using artificialy generated transactions using an MDP. The third module, concerning the planning of the activity recognition, will be tested in simulation by generating scenarii.

On a second step, the system will be integrated and tested on the VIGIL robot, a mobile robot for telepresence and ADL (Activities for Daily Living) assistant, being developed under AGE-WELL NCE (Aging Gracefully across Environments using Technology to Support Wellness, Engagement and Long Life NCE). We have selected three scenarii to test the system : (1) making a cup of tea, (2) going outside, (3) doing the laundry. All of these scenarii have several subtasks and require a repositioning of the robot in order to have a correct observability and avoid obstructing the path of the assisted elder.

Conclusion

This dissertation propose the roadmap for my thesis and, as we know, present a problem that has not been addressed yet. This problem concerns the positions of a robot in an open world to observe an activity with the following constraints : (1) the robot must not be in the way of the observed subject and (2) the robot aims to maximize the odds of observing an activity.

This research should provide some significant advances in three ways. First, it should provide a reliable method to perform ADL recognition in an open world, like at a senior's home. Secondly, it should provide improvements on both the speed and the accuracy of activity recognition techniques embedded on robots. Finally, it should improve the acceptability of the robot by both the elders and the care givers as the system

would be more reliable and the robot would stop following the elders all the time, as most of the actual systems do.

Acknowledgement

This work was supported by AGE-WELL NCE (Aging Gracefully across Environments using Technology to Support Wellness, Engagement and Long Life NCE).

References

Agrawal, R., and Srikant, R. 1995. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, 3–14. IEEE.

Bai, H.; Hsu, D.; Lee, W. S.; and Ngo, V. A. 2010. Monte carlo value iteration for continuous-state pomdps. In *Algorithmic foundations of robotics IX*. Springer. 175–191.

Beaudry, E.; Kabanza, F.; and Michaud, F. 2010. Planning for Concurrent Action Executions Under Action Duration Uncertainty Using Dynamically Generated Bayesian Networks. In *International Conference on Automated Planning and Scheduling*, 10–17.

Bresina, J.; Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D.; et al. 2002. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, 77–84. Morgan Kaufmann Publishers Inc.

Cesta, A., and Pecora, F. 2005. The robocare project: Intelligent systems for elder care. In *AAAI Fall Symposium on Caring Machines: AI in Elder Care, USA*.

De Raedt, L.; Kimmig, A.; and Toivonen, H. 2007. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *International Joint Conference on Artificial Intelligence*, volume 7, 2462–2467.

Freedman, R. G., and Fukunaga, A. 2015. Integration of planning with plan recognition using classical planners. In *2015 AAAI Fall Symposium Series*.

Freedman, R. G., and Zilberstein, S. 2017. Integration of planning with recognition for responsive interaction using classical planners. In *AAAI*, 4581–4588.

Graf, B.; Reiser, U.; Hägele, M.; Mauz, K.; and Klein, P. 2009. Robotic home assistant Care-O-bot® 3-product vision and innovation platform. In *Advanced Robotics and its Social Impacts (ARSO), 2009 IEEE Workshop on*, 139–144. IEEE.

Jenkins, S., and Draper, H. 2015. Care, monitoring, and companionship: views on care robots from older people and their carers. *International Journal of Social Robotics* 7(5):673–683.

Koppula, H. S., and Saxena, A. 2016. Anticipating human activities using object affordances for reactive robotic response. *IEEE transactions on pattern analysis and machine intelligence* 38(1):14–29.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Kurniawati, H.; Hsu, D.; and Lee, W. S. 2008. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland.

Palumbo, F.; Ullberg, J.; Štimec, A.; Furfari, F.; Karlsson, L.; and Coradeschi, S. 2014. Sensor network infrastructure for a home care monitoring system. *Sensors* 14(3):3833–3860.

Pirsiavash, H., and Ramanan, D. 2012. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2847–2854. IEEE.

Pollack, M. E. 2005. Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. *AI magazine* 26(2):9.

Raghavan, S., and Mooney, R. J. 2011. Abductive plan recognition by extending Bayesian Logic Programs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 629–644. Springer.

Raıssi, C.; Poncelet, P.; and Teisseire, M. 2005. Need for speed: Mining sequential patterns in data streams. *BDA05: Actes des 21iemes Journees Bases de Donnees Avancees*.

Ramırez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the 21st international joint conference on Artifical intelligence. Morgan Kaufmann Publishers Inc*, 1778–1783.

Ramırez, M., and Geffner, H. 2011. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *International Joint Conference on Artificial Intelligence*, 2009–2014. IJCAI/AAAI.

Rashidi, P.; Cook, D. J.; Holder, L. B.; and Schmitter-Edgecombe, M. 2011. Discovering activities to recognize and track in a smart environment. *IEEE transactions on knowledge and data engineering* 23(4):527–539.

Renz, J. 2002. *Qualitative spatial reasoning with topological information*. Springer-Verlag.

Sung, J.; Ponce, C.; Selman, B.; and Saxena, A. 2012. Unstructured human activity detection from rgbd images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 842–849. IEEE.

Weld, D. S., et al. 2005. Concurrent probabilistic temporal planning. In *International Conference on Automated Planning and Scheduling*, 120–129. AAAI Press.

Wu, J.; Osuntogun, A.; Choudhury, T.; Philipose, M.; and Rehg, J. M. 2007. A scalable approach to activity recognition based on object use. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, 1–8. IEEE.

Wu, C.-W.; Lin, Y.-F.; Yu, P. S.; and Tseng, V. S. 2013. Mining high utility episodes in complex event sequences. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 536–544. ACM.

Counterplanning using Goal Recognition and Landmarks*

Alberto Pozanco
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés (Madrid). Spain
apozanco@pa.uc3m.es

Abstract

In non-cooperative multi-agent systems, agents might want to prevent the opponents from achieving their goals. One alternative to solve this task would be using counterplanning to generate a plan that allows an agent to block other's to reach their goals. We introduce a fully automated domain-independent approach for counterplanning. It combines; goal recognition to infer an opponents goal; landmarks' computation to identify subgoals that can be used to block opponents' goals achievement; and classical automated planning to generate plans that prevent the opponent's goals achievement. Experimental results in several domains show the benefits of our novel approach.

Introduction

In non-cooperative multi-agent systems, agents might want to prevent the opponents from achieving their goals. This task has been named counterplanning (Carbonell 1981). Examples of non-cooperative multi-agent domains where this approach can provide great benefits are police controls, cyber security, or real-time strategy games, where this ability has been identified as one of the major challenges for Artificial Intelligence (Ontañón et al. 2013). Most previous counterplanning approaches are based on domain-dependent solutions, such as rule-based systems (Carbonell 1978; Rowe 2003), or Hierarchical Task Networks (HTN) (Willmott et al. 2001).

Recently, there has been increasing interest in the study and generation of agents capable of reasoning about their own and opponents' goals as well as their environment (Cox 2007). Some works follow the Goal-Driven Autonomy (GDA) process, which integrates a diverse set of AI components such as HTN planning or explanation generation (Molineaux, Klenk, and Aha 2010). Other works combine goal recognition and reasoning on actions, applying those techniques in domains such as identifying terrorist activity (Jarvis, Lunt, and Myers 2005), air combat (Borck et al. 2015), real-time strategy games (Kabanza et al. 2010), or cyber security (Boddy et al. 2005; Sarraute, Buffet, and Hoffmann 2012; Hoffmann 2015). Again, these approaches are domain-dependent. On the goal recognition side, they use plan (Kabanza et al. 2010), rules (Carbonell 1978) or behavior (Borck et al. 2015) libraries to detect their opponent's goals. On the action reasoning side, they use stored policies (Carbonell 1981), ask for human guidance following a mixed-initiative paradigm (Jarvis, Lunt, and Myers 2005), or require heavy knowledge engineering processes such as HTN based approaches (Willmott et al. 2001).

We present a fully automatic domain-independent approach for counterplanning. This approach is based on: goal recognition, landmarks, and classical automated planning. Goal recognition aims to infer an agent's plan or goals from a set of observations. In general, the observed agent can be cooperative or competitive. We use this technique to infer an opponent's goals. Fact landmarks are propositions that must be true in all valid solution plans (Hoffmann, Porteous, and Sebastia 2004). We use landmarks to identify subgoals that can be used to block the opponent's goal achievement. Classical automated planning aims to generate a sequence of actions, namely a plan, which achieves some goals from an initial state. We use it to generate plans that prevent the opponent's goal achievement.

The main idea of this novel approach is to: (1) quickly identify the actual opponent's goal g using planning-based goal recognition techniques; (2) compute the set of landmarks involved in the achievement of g; (3) select a *counterplanning landmark*, which is the first landmark where the opponent could be blocked; and (4) generate a plan to achieve the counterplanning landmark, and therefore to block the opponent's goal achievement. This approach shows how an opponent can be effectively blocked in different non-cooperative domains (Pozanco et al. 2018b; 2018a).

Domain-Independent Counterplanning

We define the two actors involved in a counterplanning problem as planning agents:

- The seeking agent ϕ, with an associated planning task Π_ϕ.
- The preventing agent α, with an associated planning task Π_α.

There can be varied relations between Π_ϕ and Π_α, and the information that one agent has from the other. For instance, the actions that both agents can perform could be the same $A_\phi = A_\alpha$, or totally different $A_\phi \cap A_\alpha = \emptyset$. They could

*This is a joint work with Yolanda E-Martín, Susana Fernández and Daniel Borrajo.

also have different or equal observations of the world. By now, we make the following assumptions: (1) ϕ's model is known by α except for its goal G_ϕ; (2) α knows a set of potential goals, $\mathcal{G}_\phi \subset F_\phi$, ϕ could be trying to achieve; (3) deterministic action outcomes and full observability of those actions by α; (4) none of the agents can replan; and (5) the temporal duration of an action $a_i \in A$ is determined by its cost $c(a_i)$. By now, we assume unit action-cost.

We formally describe a counterplanning task as follows.

Definition 1 (Counterplanning task) *A counterplanning task is defined by a tuple $CP = \langle \Pi_\phi, \Pi_\alpha, \mathcal{G}_\phi, O_\phi \rangle$ where Π_ϕ is the planning task of ϕ, Π_α is the planning task for the preventing agent, $\mathcal{G}_\phi$ is the set of sets of goals that ϕ can potentially pursue, and $O_\phi = (o_1, \ldots, o_m)$ is a set of observations by α of the execution of a plan $\pi_\phi = (o_1, \ldots, o_m, a_{m+1}, \ldots, a_k)$ that solves Π_ϕ.*

We assume that ϕ generates a plan π_ϕ to solve its planning task Π_ϕ prior to counterplanning, and that plan (as well as its corresponding goals) is unknown for α. Then, at some time step m of the execution of π_ϕ (where m can range from 1 to k, the length of π_ϕ), given all observed actions from the execution of π_ϕ, α has to infer the ϕ agent goals (from $\mathcal{G}_\phi$) and generate a solution to a counterplanning task, namely a counterplan.

Definition 2 (Counterplan) *Given ϕ agent plan $\pi_\phi = (a_{m+1}, \ldots, a_k)$, a plan $\pi_\alpha = (a_1, \ldots, a_n)$ is a valid counterplan for $\pi_\phi = (a_{m+1}, \ldots, a_k)$ if the joint execution of π_α and π_ϕ does not allow ϕ to achieve the goals in G_ϕ.*

Our approach to solve counterplanning tasks assumes that α can delete (or add in the case of negated literals) some proposition that ϕ *needs* in order to achieve its goals. There could be different definitions for *needed literals*. We use planning landmarks. Therefore, we impose two constraints: the seeking agent ϕ and the preventing agent α share some propositions, $F_\phi \cap F_\alpha \neq \emptyset$; and at least one action a in α model, $a \in A_\alpha$, must delete (add) at least one of ϕ's plan landmarks. We refer to those types of landmarks as counterplanning landmarks.

Definition 3 Counterplanning landmark *Given the set of fact landmarks from Π_ϕ, $\mathcal{L}_{\Pi_\phi}$, a landmark $l_i \in \mathcal{L}_{\Pi_\phi}$ is a counterplanning landmark for α if $\exists a \in A_\alpha$ with $l_i \in$ eff(a). If l_i is a positive literal, l_i should be in $del(a)$. If l_i is a negative literal, l_i should be in $add(a)$.*

Algorithm 1 shows the high-level algorithm used to solve a counterplanning task from the perspective of α. The algorithm first solves a goal recognition problem using RECOGNIZEGOALS given a planning domain, initial conditions, a set of candidate goals $\mathcal{G}_\phi$, and a set of observations O_ϕ. It returns T_ϕ, a probability distribution over the set of candidate goals set $\mathcal{G}_\phi$ in the form of tuples $\langle$goal, probability$\rangle$. Then, the initial state of ϕ, I_ϕ, is updated with the given observations by advancing the state from the initial I_ϕ and applying all actions corresponding to the observations in O_ϕ. Next, we select the set of most probable goals' sets $G\prime_\phi$ from T_ϕ. For each goal $g \in G\prime_\phi$, we extract the landmarks of the new ϕ planning task using EXTRACTLANDMARKS. This computation will return the set of common landmarks among all the most probable sets of goals, $\mathcal{L}_{\Pi_\phi}$. If there are not common landmarks, the counterplanning task cannot be performed. Otherwise, the algorithm selects the set of counterplanning landmarks $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$ in EXTRACTCPLANDMARKS. This process will be explained in detail later. As before, if there are not counterplanning landmarks, the counterplanning task cannot be performed. Otherwise, one of the landmarks in $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$ is negated and returned as the preventing agent's goal G_α in SELECTGOAL. This function returns the first landmark of seeker's planning task that the preventing agent can delete, i.e., it can delete (add) it before its opponent. Finally, a plan π_α is computed to achieve that goal such that it prevents ϕ from achieving its goals.

Algorithm 1 DOMAIN-INDEPENDENT COUNTERPLANNING

Inputs: $\Pi_\phi, \Pi_\alpha, \mathcal{G}_\phi, O_\phi$
Outputs: π_α

```
T_φ ← RecognizeGoals(F_φ, A_φ, I_φ, G_φ, O_φ)
I_φ ← Update(I_φ, A_φ, O_φ)
L_Π_φ ← F_φ
G'_φ ← goal(arg max_{t∈T_φ} probability(t))
π_α ← ∅
for g ∈ G'_φ do
    L_Π_φ ← L_Π_φ ∩ ExtractLandmarks(F_φ, A_φ, I_φ, g)
if L_Π_φ ≠ ∅ then
    L_Π_φ,Π_α ← ExtractCPLandmarks(Π_φ, Π_α, L_Π_φ)
    if L_Π_φ,Π_α ≠ ∅ then
        G_α ← SelectGoal(Π_φ, Π_α, L_Π_φ,Π_α)
        I_α = Update(I_α, A_α, O_φ)
        π_α ← Planner(Π_α = (F_α, A_α, I_α, G_α))
return π_α
```

Example

To illustrate our approach, let us consider a simple domain where a terrorist has committed an attack in the center of a city. Figure 1 shows the road network for this problem. The city police (α) knows that the terrorist (ϕ) wants to leave the city by either $G1$ (airport), $G2$ (train station), or $G3$ (bus terminal); so, $\mathcal{G}_\phi = \{G1, G2, G3\}$. The police has control over some cameras located at key points around the city (represented as nodes in Figure 1). The police actions consist of stopping the terrorist by setting a control at any of those points (A_α). So, the police wants to: (1) quickly know where the terrorist wants to go ($G\prime_\phi$); and (2) stop him/her as soon as possible to avoid panic breaking out. When the cameras observe that the terrorist is at L1 ($O_1 = a_1 \in A_\phi$), the police guesses that his/her goal is to reach $G1$ ($G\prime_\phi$) by doing goal recognition. The police only has resources to set one control. It knows that the terrorist must pass through L1, L2 and L3 to reach the airport. Although these four spots are counterplanning landmarks $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$, the police can only set the control at L2, L3 and $G1$ before the terrorist reaches those places. Finally, the police goal G_α will be to set the control at L2 since it is the FCL; i.e. the first spot where the terrorist can be effectively stopped.

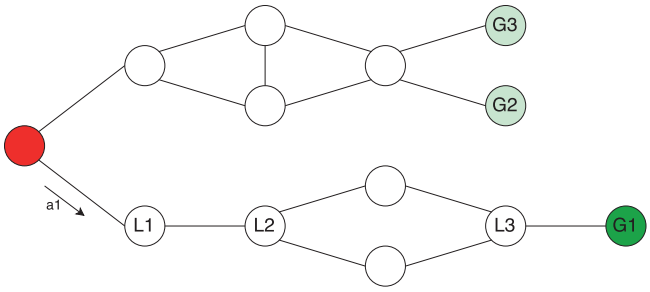


Figure 1: Terrorist capture domain. The red circle represents the initial position of the terrorist. Green circles represent his/her possible goals. Arrow a1 indicates the terrorist first observed action. L1, L2 and L3 refer to the landmark points that the terrorist has to pass through in order to reach $G1$.

Evaluation in Planning Domains

We empirically evaluate our approach on the new previously described TERRORIST domain and in the well known LOGISTICS domain. Each domain and problem conforms Π_ϕ. Additionally, in order to perform counterplanning, for each domain we have generated a new counterplanning domain, which defines the planning task Π_α for α. For each classical domain, we generate 10 random problems for ϕ. All the problems have the same number of objects. However, each problem has a different actual goal for ϕ, which is hidden for α. Their details are explained below.

- LOGISTICS. ϕ can deliver any package to any destination. It can do it by driving either trucks or planes. α can break a truck or a plane to interrupt the delivery.
- TERRORIST. ϕ may want to get to any point in the map. It can do it by navigating through points that are connected. α can set a control at a point so that they can arrest ϕ.

The set of candidate goals $\mathcal{G}_\phi$ always consists of a 20% of all the possible goals in each problem. Therefore, bigger $\mathcal{G}_\phi$ sets for the same domain mean bigger problems. In particular, for the TERRORIST domain with a problem map of 20 nodes, $\mathcal{G}_\phi$ consists of 4 random destinations. he set of observed actions was taken to be a subset of the plan solution π_ϕ, ranging from 10% of the actions, up to 70% of the actions. We did not include tests where the observed sequence is higher than 70% because our counterplanning approach degrades rapidly. The reason for this is that the number of counterplanning landmarks decreases as the number of observed actions increases. Our fully automatic domain-independent counterplanning approach works with any combination of goal recognition and classical planning approaches. For purposes of these tests, we have selected the following configuration of goal recognition techniques and planners. For the goal recognition part of our counterplanning technique, we have tested the aforementioned domains and problems using the Ramrez and Geffner [2010] approach with different the GREDY LAMA planner (Richter, Westphal, and Helmert 2011). For optimal plan computations of FCL, we use HSP*$_f$ (Haslum 2008).

Table 1 summarizes the experimental results. For each planner, each row shows average performance over the 10 problems in each domain. Each column represents different measures of quality and performance:

- $|\mathcal{G}_\phi|$: number of goals in the candidate goal's set.
- $|\pi_\phi|$, $|\pi_\alpha|$: average plan length cost for each agent.
- $|\mathcal{L}_{\Pi_\phi}|$: number of landmarks of the seeking agent planning task.
- %Obs: percentage of the actions of π_ϕ in O_ϕ. Higher percentages of observations mean that more actions of ϕ's plan have already been observed by α and, thus, executed by ϕ.
- Q: fraction of times that the actual goal G_ϕ was found to be the most likely goal $G\prime_\phi$. In our experiments, if $G\prime_\phi$ consist of more than one goal, we select the one with more counterplanning landmarks as the most likely goal. Ideally, $Q = 1$.
- Q_t: average time in seconds taken for solving the goal recognition problems.
- E: fraction of times that α executing π_α succeeds in stopping ϕ in achieving its goals. Ideally, $E = 1$.
- Pe: penalty value computed as the number of steps in π_ϕ that are successfully performed divided by the length of π_ϕ. This penalty value represents the cost paid by π_α at each time step that has not stopped π_ϕ. Lower values of Pe indicate better performance, ideally $Pe = 0$.

						Greedy LAMA			
Domain	$\lvert\mathcal{G}_\phi\rvert$	$\lvert\pi_\phi\rvert$	$\lvert\pi_\alpha\rvert$	$\lvert\mathcal{L}_{\Pi_\phi}\rvert$	%Obs	Q	Q_t	E	Pe
LOGISTICS	6	9.4	1	12.3	10	0.6	2.6	0.6	0.2
					30	0.7	2.6	0.8	0.4
					50	0.7	2.7	0.8	0.5
					70	0.9	2.8	0.7	0.8
TERRORIST	4	3.8	1	3.8	10	0.6	2.6	0.6	0.5
					30	0.6	2.8	0.8	0.7
					50	0.6	3.0	0.8	0.8
					70	0.9	3.2	0.9	1.0

Table 1: Comparison of the counterplanning approach in two domains. Figures shown are all averages over the set of problems as explained in the text.

As we can see in both domains, the higher percentage of observations, the higher Q values, as expected. The goal recognition task becomes easier as more actions have been observed (as reported in other goal recognition works). Regarding E, the fraction of times that α blocks ϕ achieving its goals is clearly related to Q. Guessing the opponent's goal right usually involves more opportunities to block it. However, there are some cases in which we can badly guess ϕ's goal and still block its goal achievement (Q value is lower than E). This happens when our analysis of the goal recognition process identifies a common landmark (to stop ϕ's plan), but selects a wrong goal as in some LOGISTICS instances. The value of E is also closely related to the percentage of observations. Lower percentage values allows α to find many landmarks where to effectively block ϕ. On the other hand, if most of the actions in π_ϕ have already been observed, there will be just a few counterplanning landmarks

Figure 2: Training army mini-game. The preventing agent detects a common landmark for both goals: destroying the barracks. The actual goal is to train a firebat.

to prevent ϕ from achieving its goal. This is also connected with the penalty values Pe. Lower percentage of observations imply that, if the opponent can be blocked, it could be done farther from the goal than if 50% or more of the plan has already been observed.

Evaluation in RTS Games

Real-Time Strategy (RTS) games are a particularly difficult type of games with many analogies to real world problems. One of the main challenges in AI applied to RTS games is to autonomously synthesize plans that counter opponent's strategies given their observations (Ontañón et al. 2013).

We have applied our domain-independent counterplanning algorithm to StarCraft, a well-known RTS game (Pozanco et al. 2018a). The evaluation was conducted in some StarCraft mini-games (as depicted in Figure involving resource gathering, troop confrontations and building order, which are part of a real StarCraft game. The results show how our counterplanning approach can effectively generate plans from scratch to counter opponent's strategies in most of these situations.

Future work

By now, we assume we are given the candidate goals for the goal recognition process (as in the usual literature on goal recognition). Future work would consist on relaxing this assumption and consider F_ϕ as $\mathcal{G}_\phi$. We also assume unit action costs. In future work, we would generalize our approach by considering non-unit action costs. Additionally, a natural extension to current work would be to assume a seeking agent capable of changing his/her plans and goals. It seems to be also possible to extend our approach to deal with noisy observations and uncertainty on the seeking agent's behavior.

Acknowledgements

This work has been partially supported by MINECO projects TIN2014-55637-C2-1-R and TIN2017-88476-C2-2-R and project RTC-2016-5407-4 funded by Ministerio de Economía y Competitividad.

References

Boddy, M. S.; Gohde, J.; Haigh, T.; and Harp, S. A. 2005. Course of action generation for cyber security using classical planning. In *ICAPS*.

Borck, H.; Karneeb, J.; Alford, R.; and Aha, D. W. 2015. Case-based behavior recognition in beyond visual range air combat. In *FLAIRS*.

Carbonell, J. G. 1978. Politics: Automated ideological reasoning. *Cognitive Science* 2(1):27–51.

Carbonell, J. G. 1981. Counterplanning: A strategy-based model of adversary planning in real-world situations. *Artificial Intelligence* 16(3):295–329.

Cox, M. T. 2007. Perpetual self-aware cognitive agents. *AI magazine* 28(1):32.

Haslum, P. 2008. Additive and reversed relaxed reachability heuristics revisited. *International Planning Competition*.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–278.

Hoffmann, J. 2015. Simulated penetration testing: From" dijkstra" to" turing test++". In *ICAPS*.

Jarvis, P. A.; Lunt, T. F.; and Myers, K. L. 2005. Identifying terrorist activity with ai plan recognition technology. *AI Magazine* 26(3):73.

Kabanza, F.; Bellefeuille, P.; Bisson, F.; Benaskeur, A. R.; and Irandoust, H. 2010. Opponent behaviour recognition for real-time strategy games. In *AAAI Workshop on Plan, Activity, and Intent Recognition*.

Molineaux, M.; Klenk, M.; and Aha, D. W. 2010. Goal-driven autonomy in a navy strategy simulation. In *AAAI*.

Ontañón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; and Preuss, M. 2013. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in games* 5(4):293–311.

Pozanco, A.; Blanco, A.; E-Martín, Y.; Fernández, S.; and Borrajo, D. 2018a. Counterplanning in real-time strategy games through goal recognition. In *Proceedings of 6th Goal Reasoning Workshop, IJCAI*.

Pozanco, A.; E-Martín, Y.; Fernández, S.; and Borrajo, D. 2018b. Counterplanning using goal recognition and landmarks. In *Proceedings of IJCAI'18*.

Ramírez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*.

Richter, S.; Westphal, M.; and Helmert, M. 2011. LAMA 2008 and 2011. In *International Planning Competition*.

Rowe, N. C. 2003. Counterplanning deceptions to foil cyber-attack plans. In *IEEE Workshop on Systems, Man and Cybernetics Society*.

Sarraute, C.; Buffet, O.; and Hoffmann, J. 2012. POMDPs make better hackers: Accounting for uncertainty in penetration testing. In *AAAI*.

Willmott, S.; Richardson, J.; Bundy, A.; and Levine, J. 2001. Applying adversarial planning techniques to Go. *Theoretical Computer Science* 252(1-2):45–82.

Concurrent Multi-Agent Planning (Extended Abstract)

Shashank Shekhar
The Department of Computer Science
BGU of the Negev, Beer-Sheva, Israel - 84105
shekhar@cs.bgu.ac.il

Abstract

Multi-Agent Planning (MAP) is the problem of finding a sequence of actions for agents acting in a world such that the world is transformed in a desired way. Many such problems of practical importance call for the use of multiple autonomous agents that work together to achieve a common goal. For example, disaster response teams typically consist of multiple agents that have multiple tasks to perform, some of which require or can benefit from the cooperation of multiple agents. They not only involve reasoning with incomplete or complete information but also the ability to account for different aspects of interactions that, more often than not, include a simultaneous collaboration of multiple agents to perform joint tasks under uncertainty. In such scenarios, it is interesting and quite challenging to enable autonomous agents to perform interacting actions (actions whose joint effect differs from the union of their individual effects) effectively due to their combinatorial nature. In this abstract we present an overarching goal of our research, the challenges already addressed, and the state of the work in progress.

Introduction

There are abundant examples of real and virtual multi-agent systems around us, and with increasing penetration of the Internet of Things (IoT) and advances in robotic technologies, their number will greatly increase. Many of these systems are collaborative by nature, working together to achieve joint-goals. For example, a team of robots in an organization is naturally collaborative, in which, the robots collaborate on common goals. The overall goal of this proposal is to enhance such systems to efficiently and autonomously exploit their combined capabilities. We strive for such enhancements by providing (or intend to) effective models and algorithms for collaborative tasks and executions. We focus on some of the interesting issues that need to be addressed in order to see a widespread use of planning technologies in collaborative multi-agent systems. In our present work, we are interested in two specific modalities.

In the first, we strive to efficiently modeling and planning with collaborative actions – actions that require concurrent execution by multiple agents. Consider an action of pushing a heavy box. It requires multiple agents to push the box concurrently, and if agents execute at different times then it does not move. Perhaps we are surrounded by many such examples like picking up a wounded person on a stretcher and moving, needs concurrent lifting and coordinated motion. We find that succinctly specifying the effects of different combinations of concurrent actions and efficiently planning with such a specification is challenging. Therefore, we develop a formalism that we believe can address this challenge. Our formalism succinctly and naturally specifies the effects of concurrent actions to extend state of the art planning algorithms to handle such specifications.

In the second, we strive to scale up MAP under uncertainty and partial observability. In MAP under uncertainty, agents have only partial information about their environment, however they may actively obtain additional information using their sensors. Given that single-agent systems face similar challenges in this scenario, things become worse in the MAP framework due to the difficulty of coordinating the actions of the agents having different states of information.

We formulate problems in MAP with partial observability based on a qualitative version of Dec-POMDPs (Bernstein et al. 2002; 2009) – an elegant stochastic model. This qualitative framework, however, is an extension of the single-agent contingent planning framework for handling multi-agent planning problems with collaborative actions under uncertainty, called QDec-POMDP (Brafman, Shani, and Zilberstein 2013). It has the potential to provide a better trade-off between expressive power and scalability for practical problems, and a recent development in our group approves that. This work addresses an offline approach called Iterative Multi-Agent Planning (IMAP), which scales well to larger QDec-POMDPs. Unlike Brafman, Shani, and Zilberstein (2013), the authors suggest to construct single-agent policies iteratively, an approach that was successfully tried in the case of deterministic actions with no concurrency involved and full information (Borrajo 2013). The main challenge perhaps, the scenario we are interested in, is to coordinating the actions of the agents despite their different states of information. In the IMAP approach, each agent constructs its own policy assuming the availability of other agent if needed. Later, the agent extracts a set of collaboration constraints/commitments from her policy. The following agents attempt to plan while satisfying these constraints, and if they fail to satisfy, they communicate the constraints that can be

met. The first agent backtracks with this information.

Online planning approaches scale much better than offline approaches such as POMDPs (Silver and Veness 2010) and Contingent Planning (Bonet and Geffner 2011). Online replanning is a popular method of doing this. In a MAP framework this approach does not work when communication is constrained, even if agents initially share a plan. When agents replan, they do not really know whether other agents are following the same old plan or they were forced to change it. Since this seems challenging, and also an important aspect in our formalism, in which, agents collaborate on certain tasks, we intend to study this thoroughly in future. We propose an online replanning approach that seems to work even if agents do not communicate. Following the IMAP algorithm, in future, the major questions are: What should be the form of the commitments? How exactly are they generated?

What happens if we introduce actions with non-deterministic finish times in planning problems with partial observability which are collaborative by nature? In the above proposed online approach, with non-stochastic action finish times, the commitments generated by an agent will be hard deadlines for other agents, given that there is no communication involved during execution. An agent performing her non-collaborative (single-agent) actions must compute the probability with which she can finish her current execution, so that, she will be available to satisfy the future commitments. If this probability is below some threshold, she would replan. In the real world scenarios, it appears as an important aspect to consider for this formulation, unlike IMAP in which non-durative actions are used. For examples, when multiple robots jointly plan to use some common resources, e.g., passing through a corridor, while at most one agent can pass through it at once, or multiple agents have to lift a table together which is kept at some random location, temporal uncertainty appears useful to consider.

This extended abstract is organized as follows. First, we briefly describe the challenges we already address. We cover modeling of joint-actions, planning with multi-actions, and then explain an extended formalism to handle privacy preserving planning with interacting actions. We discuss some of the important results. This is followed by the section of planning under uncertainty and partial observability, where we discuss the state of our current work in progress. This briefly covers our motivation and future endeavors.

Representing and Planning with Interacting Actions and Privacy

This section summarizes our proposed formalism and a compilation based approach to planning by teams of agents with interacting actions (Shekhar and Brafman 2018).

So, what happens when multiple agents perform actions concurrently? In principle, every combination of actions performed concurrently by a group of agents, a *joint-action*, may define a different state-transition function. But as the number of joint-actions is exponential in the number of agents, specifying an explicit model for each combination of single-agent actions is impractical except for very simple cases. Therefore, to be succinct, a representation for joint-actions must be compositional. That is, there must be some way of deducing the effect of the concurrent execution of actions $\langle a_1, \ldots, a_n \rangle$ from the effects of smaller combinations.

The primary challenge for planning for multi-agent systems with interacting actions is to find a model for joint-actions involving a large set of agents and large sets of individual actions, that is both succinct in natural settings and supports efficient planning. Due to the associated difficulties with interacting actions, most work on MAP algorithms ignores this issue and considers concurrent non-interacting or non-interfering actions, or just sequential actions only. While much can be achieved without considering interacting actions, there are settings where agents must coordinate their actions carefully to obtain desirable effects: a single-agent may be unable to lift or push heavy items, whereas this is possible for multiple agents acting together, e.g., if a table is not lifted from both sides concurrently, objects on it will fall. Therefore, our primary contributions in this work are; an intuitive formalism for specifying joint-actions in a compositional way and the definition and empirical evaluation of a compilation-based approach to planning by teams of agents with interacting actions, as well as privacy, for which we introduce a number of new domains. We give an approach for representing interacting actions succinctly and show how such a domain model can be compiled into a standard single-agent planning problem as well as to privacy preserving multi-agent planning.

To define the effect of joint-actions, we introduce *collaborative actions*. A collaborative action is a minimal combination of single-agent actions that cannot be defined as the union of its components. A joint-action is defined as a *well-formed* set of single-agent and collaborative actions. A joint-action is well formed if its components cannot be combined to yield more complex components.

The difficulty of planning with joint-actions depends on what interactions are allowed, and whether a distributed and privacy preserving algorithm is required. In the simplest case, only *non-interfering* (Blum and Furst 1997) concurrent actions are allowed in order to reduce make-span. A slightly more interesting case is when concurrent actions can destroy each other's preconditions, therefore *interfering*. More complicated is the case where the effects of actions performed together differ from the union of their individual effects. Finally, on top of each of these cases, one can introduce the goal of preserving privacy. Our compilation based approach efficiently handles all these difficulties associated with planning with concurrent and interacting actions.

Modeling joint-actions

We address that what happens when multiple agents act concurrently, i.e., when a *joint-action* is performed. To this end, we define multi-actions. A *multi-action* is a set of single-agent and collaborative actions with consistent preconditions and consistent effects such that an agent executes at most one element in this set. The effect of a well-formed multi-action is the union of the effects of the actions it contains. We must associate only one *joint-action* with every well-formed multi-action. The elements of the joint-action

are the single-agent actions contained in this multi-action (a few of them belong to collaborative actions), with a *noop* added for each non-acting agent. The transition function induced by this joint-action is that of its underlying multi-action. We require four types of actions, and later we will use the term *action* to refer to a single-agent or a collaborative action. We drop the distinction between *multi-action* and *joint-action*, simply using the former term.

Model Our formal semantic model is essentially a transition system. Transitions correspond to joint-actions, and hence their special structure needs to be reflected. A *multi-agent planning model* $\langle S, A, s_0, G, \Phi, \{A_i : 1 \leq i \leq n\}\rangle$ is defined as follows: S is a set of states; A is a set of joint-actions; $s_0 \in S$ is the initial state, $G \subseteq S$ is the set of goal states, Φ is the set of agents, where $|\Phi| = n$ by convention; and A_i are the single-agent action symbols for agent $\phi_i \in \Phi$, where A_i will always contain *no-op*$_i$. Every action $a \in A$ consists of a partial functions from S to S and a vector $(a_1, \ldots, a_n)$ of single-agent action symbols. We write $a(s)$ to denote the state obtained when applying a in state s. A plan $\pi = a_1, a_2, \ldots, a_k$ is a sequence of joint-actions such that $a_k(\cdots(a_1(s_0))) \in G$.

Language A MAP domain specification consists of $\langle P, I, g, \Phi, \{A_1, \ldots A_n\}, A_c\rangle$, where P is a set of ground propositions, $I \subset P$ is the initial state, $g \subset P$ is the goal condition, Φ is a set of agent names, A_i is a set of *single-agent* actions, and A_c is a set of *collaborative* actions.

A *single-agent action* has the form $a = \langle symbol, pre(a), \mathit{eff}(a)\rangle$, where *symbol* is the action name, and pre(a) and eff(a) are well-formed sets of literals. pre$(a)^+$ is the set of positive pre-conditions, pre$(a)^-$ is the set of negative pre-conditions, eff$(a)^+$ is the set of add effects, and eff$(a)^-$ is the set of delete effects. While, a *collaborative action* has the form $a_c = \langle symbol, pre(a), \mathit{eff}(a), e = \{a_1, \ldots, a_k\}\rangle$, where *symbol*, pre($a$) and eff($a$) are as above, and e is a set of single-agent action symbols, such that no two action symbols in e belong to the same agent in Φ.

To simplify notation, clarity and reduce clutter we use the generic name *action* to refer to either a single-agent action or a collaborative action whenever possible; we will drop the distinction between an action and its symbol; and we write $e(a)$ to denote the elements of an action a. When a is a single-agent action, $e(a) = a$, and in a collaborative action $e(a)$ is simply e, the set of single-agent actions in a's definition. We also write $Agt(a)$ to denote the set of agents acting in a: $Agt(a) = \{\phi_i | \exists a_i \in e(a), a_i \in A_i\}$, i.e., agents for whom a contains an element from their action set.

Interpretation The correspondence between the model and domain specification is defined as follows. The set of states S corresponds to all possible truth assignments to P. We often equate a state with the list of propositions satisfied in it. Thus, s_0 is the state associated with I. G consists of all states containing the propositions in g.

Given a specification of actions, we formally define a *multi-action* as a set of actions $a_m \subseteq A_c \cup (\cup_{i=1}^{n} A_i)$ such that (1) for every $a, a' \in a_m : Agt(a) \cap Agt(a') = \emptyset$, and (2) $\cup_{a \in a_m} pre(a)$ and $\cup_{a \in a_m} \mathit{eff}(a)$ are both well formed. Condition (1) ensures that no agent will be an actor in more than one action in a_m. Condition (2) ensures that the effects and the preconditions of the actions in a_m do not conflict.

Given a multi-action a, the result of applying a in s, $a(s)$, is consistent when $pre^+(a) \subseteq s$ and $pre^-(a) \cap s = \emptyset$, results in $a(s) = (s \setminus \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$. We naturally extend the notations e and Agt to multi-actions: $e(a_m) = \cup_{a \in a_m} e(a)$; $Agt(a_m) = \cup_{a \in a_m} Agt(a)$. We will refer to members of $e(a_m)$, which are all single-agent actions, as its *elements*, and to the actions in a_m as its *members*.

To address this in our interpretation, we require multi-actions to be *well-formed*.

Definition 1. *A multi-action a_m is* well-formed *if no subset of its elements $\{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\} \subset e(a_m)$ satisfies the following two conditions: (1) $\{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ are not elements of a* single *collaborative action $a_c \in a_m$. (2) there exists a collaborative action $a_c \in A_c$ such that $e(a_c) = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$.*

If we have an action *push* and a collaborative action *2push* then the multi-action: a_m = {*2push*(a_1, a_2, b),*push*(a_3, b)} is not well-formed. This is because there is a subset of its elements: {push(a_2, b), push(a_3, b)} that are not part of the same collaborative action in a_m, yet there is a collaborative action 2push(a_2, a_3, b) whose elements are exactly {push(a_2, b), push(a_3, b)}.

With Definition 1, we state the following (the proof is excluded from the text, and the reader is referred to the longer/journal version of the paper).

Lemma 1. *Let $\{a_1, \ldots, a_m\}$ be a joint-action. Then, there exists at most one well-formed multi-action a_m such that $e(a_m) = \{a_1, \ldots, a_m\}$.*

Planning With Multi-Actions

We separate the compilation treatment into two cases: multi-actions whose member actions do not interfere, and in a more general case, referred to as multi-actions with *precondition/effect* interaction.

Non-Interfering Actions If multi-actions with interferences are not allowed, all the allowed interactions are already captured in the collaborative actions. Hence, the only benefit of performing them jointly is make-span reduction. That is, the set of states reachable with multi-actions and with their component (single-agent and collaborative) actions is identical. Thus, we can use any single-agent classical planning algorithm to solve the problem by combining single-agent and collaborative actions together to obtain a single-agent planning problem in which the agents are simply objects.

Multi-Actions with *Pre/Eff* Interactions The former scheme may become both unsound and incomplete when we allow multi-actions that contain actions that delete or add preconditions of other actions. Such action interactions seem natural when we consider true concurrency. For example, there is no reason we would want to exclude two agents from concurrently pushing a box, even though each push action deletes the precondition of the other, as it changes the

location of the box. In this scenario we need to actively generate well-formed multi-actions. This requires a non-trivial compilation scheme. If we allow actions that delete preconditions of each other, we must also address the subtle semantic issue of when do two actions conflict. If we allow a multi-action containing *sail*(*a1,boat,origin,destination*) and *sail*(*a2,boat,origin,destination*), why we should not allow a multi-action containing *sail*(*a1,boat,origin,destination1*) and *sail*(*a2,boat,origin,destination2*). Intuitively, we view the effects: *at*(*boat,destination1*) and *at*(*boat,destination2*) as inconsistent. While this would be clear with a multi-valued formulation of the problem, it is not obvious in the boolean case, as the two propositions are logically consistent. Thus, in this work we assume that additional declarative information about when actions conflict is provided.

The compilation scheme We give a very high level details of compiled problems with collaborative actions into single-agent planning problems. They (1) Properly address *Pre/Eff* conflicts; (2) Support collaborative actions; and (3) Ensure that multi-actions are well-formed. Our compilation alters the action description so that such serialization can still work in the more general case. For MAP problems $\langle P, I, g, \Phi, \{A_1, \ldots A_n\}, A_c\rangle$, classical planning problems $\langle P_{Cl}, A_{Cl}, I_{Cl}, g_{Cl}\rangle$ are generated using our compilation approach. However, the compilation details are skipped from the running text.

Adding Privacy

Privacy Preserving Planning (PPP) (Nissim and Brafman 2014) supports agents that wish to plan collaboratively without revealing private information about their local state, their private actions, and their cost. PPP algorithms are able to compute a joint-plan in a distributed manner without revealing private information. We briefly explain how to modify our current specification and compilation technique to support PPP with interacting actions. So, in the input language to a PPP, each agent has a separate domain specification that contains a description of its actions, which maintains privacy. In PPP, the domain description of each agent contains: a complete description of all its actions and the public projection of the public actions of other agents; together with public propositions and propositions private to the agent. We extend this description with collaborative actions. While a collaborative action is public by definition, some of its preconditions or effects could be private to one of the agents. Notice that while the specification is now distributed among n agents, the semantics remain the same.

PPP with Collaborative Actions Existing PPP algorithms are distributed, and this raises the question of when to insert a collaborative action into the plan. One agent cannot commit to a collaborative action in the name of another agent because of being uninformed of her private precondition. We believe that splitting collaborative actions into individual action components is a simple solution which allows us to use any existing PPP planner. We add this to the compilation approach to get regular Distributed-MAP problems. For example, we split *2push*(a_1, a_2, b) into *2push*$_1(a_1, a_2, b)$ and *2push*$_2(a_1, a_2, b)$. If a_2 is not mentioned in the description of *2push*$_1$ and a_1 is not mentioned in the description of *2push*$_2$, we obtain *2push*$_1(a_1, b)$ and *2push*$_2(a_2, b)$. Accordingly we also split the propositions in the action schema. Thus, the first pushing agent does not need to commit to the identity of the second agent.

Empirical Evaluation

As there is no implemented algorithm to compare against and no established domain with interacting actions, we define a new set of domains and instances. We used four MA-PDDL domains in our experiments. The two new domains are TableMover and ApartmentMover, and the other two are modified versions of Maze and BoxPushing domain. All these domains contain set of single-agent and collaborative actions with the elements of privacy. For each domain the compilation generates a centralized version and a distributed privacy preserving planning version. We evaluate the scalability of our compilation approach over these domains.

Our compilation approach scales well in each domain with no privacy. Each translation single-agent problem is given to Fast-Downward (FD) (Helmert 2006). The search approach used in FD is *lazy-greedy* with h_{FF} heuristic (Hoffmann and Nebel 2001). We also generate results for distributed PPP, for which we used the distributed PPP solver GPPP (Maliah, Shani, and Stern 2017). This planner is less optimized than FD (for single-agent problems it was 123 times slower, with average ratio per domain ranging from 40 to 287). Hence, we used simpler problems than the ones used in the no privacy case. Compilation time is negligible.

Work In Progress

We focus on devising online algorithms for effective collaboration under partial-observability which can enable agents to perform joint tasks effectively. Based on this approach, we are also interested in formulating single-agent problems that pose uncertainty about the action execution time and support exogenous events. We relate this to our proposed formalism for MAP with collaborative actions under partial observability, and question that when an agent should replan if she is committed to participate in a collaborative task at some point in future.

Online Planning

Online replanning is a popular method for this, where an initial partial plan is generated under certain assumptions. As long as these assumptions are not refuted by observations, it is followed. Once an inconsistent observation is obtained, a new plan is generated. In the multi-agent case, this approach fails when communication is constrained, even if agents initially share a plan. A major difficulty is that agents do not know which observations were sensed by other agents, upon replanning. Hence, they do not know whether other agents still follow the agreed plan or were forced to modify their plan. This raises a serious difficulty if the plan requires collaboration with other agents.

One way of addressing this problem is by communicating local observations that are inconsistent (Wu, Zilberstein, and Chen 2009). However, we suggest a different approach

that can work even when communication is not possible and is based on the idea of conditional coordination constraints used in the IMAP algorithm. When the partial plan is generated off-line, we extract the relevant constraints from it. The agents will replan taking these commitments into account. The commitments serve as anchors for the other agents, facilitating coordination. The major questions to be dealt with in future are: Developing the right representation for such commitments; Generating commitments off-line based on a partial plan; Identifying conditions where the computed commitments can be ignored. We consider this as one of the most challenging and important directions of research.

Online Replanning Under Temporal Uncertainty

As we discussed before, in an online approach, when communication is constrained, future commitments of the agents work as hard deadlines. Meanwhile, to begin with this, we formulate a quite similar single-agent formalism that poses actions with stochastic finish durations and deterministic exogenous events (Garrido, Fox, and Long 2002). We propose to handle this formalism as follows: If an action takes more time than expected, an agent would replan so that the future deadlines (unaltered) can be met. We motivate this formulation by a suitable real world scenario. For example, suppose an agent plans to buy at two shops A and B. A opens between 2PM and 6PM while B opens between 3PM and 4PM. Her current plan is to start from her house, go to shop A, do shopping at A, go to shop B, do shopping at B, and from B go home. If reaching A takes more time than expected time, and given that B is far from her house than A and the above given opening timings of the shops, she would replan. The more feasible looking plan may be: reach B first from wherever she is, shop at B, come to shop A, do shopping at A, come back home.

In future, we intend to focus on different approaches like monitoring the likelihood of success of meeting deadlines, and replanning based on this. One can maintain multiple plans – they can be generated online, and tracking the likelihood of each and selecting actions that correspond to the best suited plan.

References

Bernstein, D. S.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.* 27(4):819–840.

Bernstein, D. S.; Amato, C.; Hansen, E. A.; and Zilberstein, S. 2009. Policy iteration for decentralized control of markov decision processes. *J. Artif. Intell. Res.* 34:89–132.

Blum, A., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90:281–300.

Bonet, B., and Geffner, H. 2011. Planning under partial observability by classical replanning: Theory and experiments. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 1936–1941.

Borrajo, D. 2013. Multi-agent planning by plan reuse. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, 1141–1142.

Brafman, R. I.; Shani, G.; and Zilberstein, S. 2013. Qualitative planning under partial observability in multi-agent domains. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*.

Garrido, A.; Fox, M.; and Long, D. 2002. A temporal planning system for durative actions of PDDL2.1. In *Proceedings of the 15th Eureopean Conference on Artificial Intelligence, ECAI'2002, Lyon, France, July 2002*, 586–590.

Helmert, M. 2006. The fast downward planning system. *J. Artif. Int. Res.* 26(1):191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: fast plan generation through heuristic search. *J. Artif. Int. Res.* 14(1):253–302.

Maliah, S.; Shani, G.; and Stern, R. 2017. Collaborative privacy preserving multi-agent planning - planners and heuristics. *Autonomous Agents and Multi-Agent Systems* 31(3):493–530.

Nissim, R., and Brafman, R. I. 2014. Distributed heuristic forward search for multi-agent planning. *J. Artif. Intell. Res.* 51:293–332.

Shekhar, S., and Brafman, R. I. 2018. Representing and planning with interacting actions and privacy. In *Proceedings of ICAPS, June 24 - 29, 2018*.

Silver, D., and Veness, J. 2010. Monte-carlo planning in large pomdps. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, 2164–2172.

Wu, F.; Zilberstein, S.; and Chen, X. 2009. Multi-agent online planning with communication. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*.

Explanations as Model Reconciliation: A Foil Based Method for Multi-Model Plan Explanation

Sarath Sreedharan*
School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ 85281 USA

Abstract

There is a growing interest within the AI research community to develop autonomous systems capable of explaining their behavior to users. While there have been many previous works that have tried to tackle the problem of plan explanations, most of these works have viewed the task as that of merely describing the plans using the domain model used in generating that plan. This approach misses the point that explanations are by its very nature, a multi-model problem. Explanations should help clarify the explanandum (a plan in this case) to some external agent that is evaluating it with their model and as such any effective explanation must consider the beliefs, presumptions, and expectations of the explainee. In this abstract, I provide a summary of my research related to the development of algorithms for generating multi-model explanations for human in the loop planning problems.

1 Introduction

As AI systems become an increasing part of our daily lives, it is of utmost importance that these autonomous systems can work with humans in a fluent and explicable manner. This would require the humans working with these systems to understand the reasoning and intention behind the actions taken by these systems at an intuitive level. To achieve this we need to ensure that the AI systems that we build are capable of explaining its plans and behavior to its (often AI illiterate) end users in a meaningful manner. For designing such systems, we need to have a clear idea of the kind of explanations that humans would consider acceptable and helpful. Thankfully, we can draw from the rich literature on everyday explanations that exist in social sciences. As [Miller, 2017] points out, most of these work seems to agree that explanations should be contrastive, selective and social. While most previous works on explaining plans have focused on merely describing the plan in question ([Seegebarth et al., 2012; Bercher et al., 2014; Kambhampati, 1990]) our attempt at solving the problem of plan explanation was designed around these central tenants from social sciences. We view the problem of plan explanation as essentially a multi-model problem and posit that the goal of an explanation should be to help the explainee (generally a human) better understand the plan in question and contrast it with some alternative plans. We will call such alternative plans the foils to the original plan in question. Following this reasoning, it is evident that any helpful explanation needs to take into account the explainee's believes about the task and her/his expectations about the agent generating the behavior. Thus the explanation should help bring the explainee's model of the task and the robot (i.e., the model that they are using to evaluate the plan in question) closer to the robot model, so they can correctly understand the plan and realize why it is preferred over the foils in question. Figure 1 presents the basic setup of this multi-model planning problem and how the robots can explain its plans to help reconcile the human model.

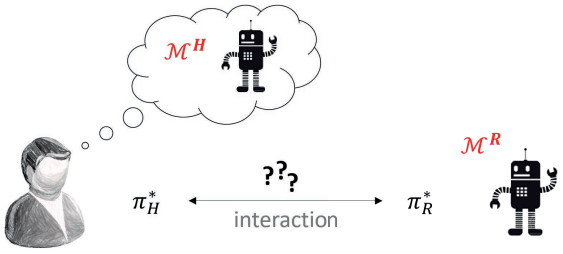


Figure 1: The multi-model planning setup as presented in [Chakraborti et al. 2017]. The robot here generates an optimal plan π_R^* with respect to its model $\mathcal{M}^R$, which is interpreted by the human with respect to his model $\mathcal{M}^H$. Explanations are needed when π_R^* is *not* an optimal plan with respect to $\mathcal{M}^H$.

The rest of the document is structured as follows. In section 2, I will introduce the basic setup of the problem and formally define foil based explanations. Next, we will look at ways we can generate such explanations when the foils may not be explicitly provided. In section 4, we will relax the assumption that the human model is completely known. Finally, I will conclude the document by discussing how we could potentially connect problem of explanation generation to that of

*All works described in this document are joint work with Tathagata Chakraborti, Yu Zhang and Subbarao Kambhampati. In all these works, I was either the lead author or one of the lead authors.

generating explicable plans and will discuss some future directions.

2 Explanation as Model Reconciliation

Let us consider a a classical planning problem defined as a tuple $\mathcal{M} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$ - where F is a set of fluents that define a state $s \subseteq F$, and A is a set of actions - and initial and goal states $\mathcal{I}, \mathcal{G} \subseteq F$. As mentioned earlier, we will be considering multi-model planning problems in this setting. We define a multi-model planning problem by the tuple $\Psi = \langle \mathcal{M}^R, \mathcal{M}_h^R \rangle$ where $\mathcal{M}^R = \langle D^R, \mathcal{I}^R, \mathcal{G}^R \rangle$ and $\mathcal{M}_h^R = \langle D_h^R, \mathcal{I}_h^R, \mathcal{G}_h^R \rangle$ are respectively the planner's model of a planning problem and the human's understanding of the same. For keeping the discussions simple, we will focus on cases where the human is merely an observer and all actions are taken by the robot. In this setting, human requires an explanation for the robot's optimal plan π_R^*, if any of the following two conditions hold.

(1) $\pi_R^*(I_h^R) \not\supseteq G_h^R$;

(2) There exists a set of plans $F = \pi_1, .., \pi_m$ (called foils), such that for each $\pi_i \in F$, we have $\pi_i(I_h^R) \supseteq G_h^R$ and $C(\pi_i, \mathcal{M}_h^R) < C(\pi_R^*, \mathcal{M}_h^R)$

This above conditions posit that the plan needs to be explained if the human thinks that the plan is incorrect or believes there are better plans. Now an explanation in this setting can be a set of model updates $\mathcal{E}$ such that

(1) $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$; and

(2) $\forall \pi_i \in F, C(\pi_R^*, \widehat{\mathcal{M}}_h^R) \leq C(\pi_\rangle, \widehat{\mathcal{M}}_h^R)$.

Which means that in the model $\widehat{\mathcal{M}}_h^R$ obtained by applying explanation $\mathcal{E}$ on the human model $\mathcal{M}_h^R$, the robot plan will be correct and preferred over the foils.

3 Explaining the Optimality of a Plan

The above definition of explanation assumes that the set of foils F is explicitly provided. In many cases, the human may not bother with (or capable of) providing the system with a set of explicit foils. One way to handle such situation would be by generating explanations that can resolve all possible foils. In other words, identify an explanation that ensures the optimality of the given plan in the resultant model. This direction was investigated in our paper [Chakraborti et al., 2017], where we introduced algorithms that are capable of identifying minimal explanations (referred to as Minimally Complete Explanations or MCE) that can ensure the optimality of a given plan in the human model. We formulated these algorithms as a heuristic search over the space of models that can be generated from the human mental model. The paper also performed a very detailed analysis of the various desirable properties of such explanations. An interesting property we identified was that such minimal explanations are not necessarily monotonic. This means that it is possible for an explanation (i.e model updates) to invalidate a plan that was previously explained by the robot. In the paper, we also showed that we could generate explanations that are guaranteed to be monotonic (called Minimally Monotonic Explanations or MME) by a similar model space search. Figure 2 presents a graphical representation of the model space search to identify MCE and MME.

4 Explaining with Multi-Model and Incomplete Models

One of the assumptions made by the earlier mentioned approaches is the availability of complete and correct representation of the human mental model. This could be an extremely hard requirement to meet, and the task of learning these models is further complicated by the fact that it may not be possible to observe any traces generated from these models. One possibility would be to use incomplete models that are easier to learn. We will consider one such model called Annotated PDDL model ([Nguyen, Sreedharan, and Kambhampati, 2017]) and see how we can use it for explanation generation (the details of this work can be found at [Sreedharan, Chakraborti, and Kambhampati, 2017b]).

An annotated model is given by the tuple ${}^a\mathcal{M} = \langle {}^a\mathcal{D}, {}^a\mathcal{I}, {}^a\mathcal{G} \rangle$ with a domain ${}^a\mathcal{D} = \langle F, {}^aA \rangle$ – where F is a finite set of fluents that define a state $s \subseteq F$, and ${}^aA$ is a finite set of annotated actions – and annotated initial and goal states ${}^a\mathcal{I} = \langle \mathcal{I}^0, \mathcal{I}^+ \rangle$, ${}^a\mathcal{G} = \langle \mathcal{G}^0, \mathcal{G}^+ \rangle$; $\mathcal{I}^0, \mathcal{G}^0, \mathcal{I}^+, \mathcal{G}^+ \subseteq F$. Action $a \in^a A$ is a tuple $\langle c_a, \textit{pre}(a), \widetilde{\textit{pre}}(a), \textit{eff}^{\pm}(a) \rangle, \widetilde{\textit{eff}}^{\pm}(a) \rangle$ where c_a is the cost and, in addition to its preconditions and add/delete effects $\textit{pre}(a), \textit{eff}^{\pm}(a), \subseteq F$ each action also contains *possible preconditions* $\widetilde{\textit{pre}}(a) \subseteq F$ containing propositions that action a *might* need as preconditions, and *possible add (delete) effects* $\widetilde{\textit{eff}}^{\pm}(a) \subseteq F$) containing propositions that the action a *might* add (delete, respectively) after execution.

An *instantiation* of an annotated model ${}^a\mathcal{M}$ is a classical planning model where a subset of the possible conditions have been realized, and is thus given by the tuple $ins({}^a\mathcal{M}) = \langle \mathcal{D}, \mathcal{I}, \mathcal{G} \rangle$ with domain $\mathcal{D} = \langle F, A \rangle$, initial and goal states $\mathcal{I} = \mathcal{I}^0 \cup \chi$; $\chi \subseteq \mathcal{I}^+$ and $\mathcal{G} = \mathcal{G}^0 \cup \chi$; $\chi \subseteq \mathcal{G}^+$ respectively, and action $A \ni a = \langle c_a, \textit{pre}(a) \leftarrow \textit{pre}(a) \cup \chi;\ \chi \subseteq \widetilde{\textit{pre}}(a), \textit{eff}^{\pm}(a) \leftarrow \textit{eff}^{\pm}(a) \cup \chi;\ \chi \subseteq \widetilde{\textit{eff}}^{\pm}(a) \rangle$. Given an annotated model with k possible conditions, there may be 2^k such instantiations, which forms its *completion set* [Nguyen, Sreedharan, and Kambhampati, 2017] given as $\langle\langle \hat{\mathcal{M}} \rangle\rangle$.

If we use an incomplete model to represent the human mental model, this, in turn, means that the actual human mental model is one of the possible models within the completion set. So a safe approach to explanation generation in this setting would be to find a single explanation that is a satisfactory explanation for the entire set of models (Figure 3). This may sound quite daunting but in the following section, we will show, how we can solve this problem quite efficiently. We will mainly achieve this by focusing our attention on just two of the models in the completion set. Namely, the most relaxed model $\mathcal{M}_{max}$ possible and the least relaxed one $\mathcal{M}_{min}$. The former is the model where all the possible add effects (and none of the possible preconditions and deletes) hold, the state has all the possible conditions set to true, and the goal

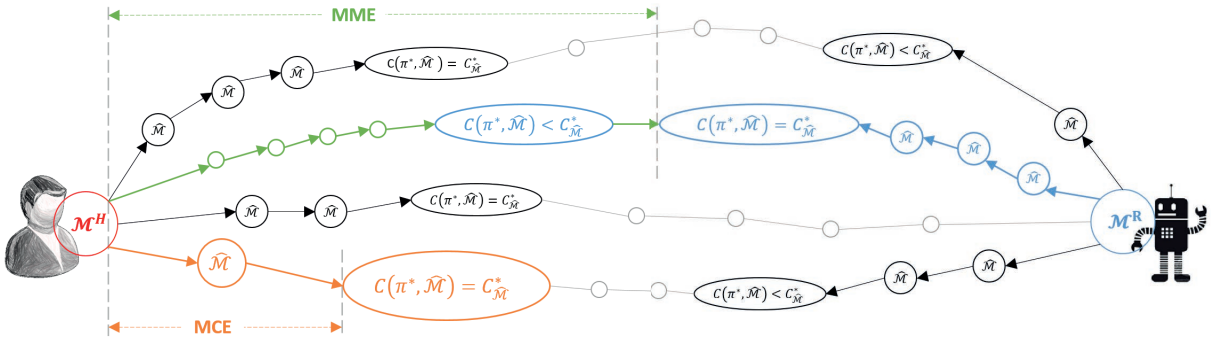


Figure 2: Illustration of model space search for MCE & MME.

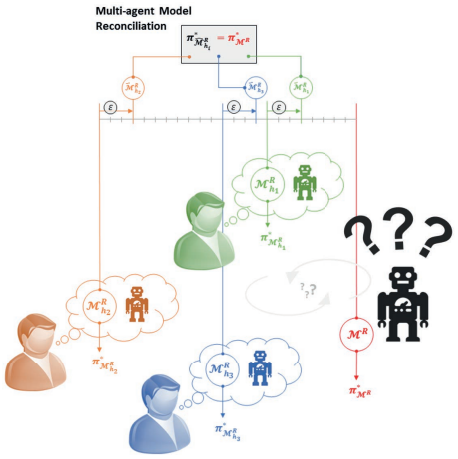


Figure 3: The model reconciliation process in case of model uncertainty regarding $\mathcal{M}^R_h$.

is the smallest one possible; while in the latter all the possible preconditions and deletes (and none of the possible adds) are realized and with the minimal start state and the maximal goal. This means that, if a plan is executable in $\mathcal{M}_{min}$ it will be executable in all the possible models. Also, if this plan is optimal in $\mathcal{M}_{max}$, then it must be optimal through out the set. Of course, such a plan may not exist, but we are not trying to find one either. Instead, we are trying to find a set of model updates which when applied to the annotated model, produces a new set of models where a *given* plan is optimal. In providing these model updates, we are in effect reducing the set of possible models, to a smaller set. The new set need not be a subset of the original set of models but will be equal or smaller in size to the original set. For any given annotated model, such an explanation exists, and we intend to find the smallest one.

Now we can find this explanation by updating the MCE search described in the previous section. The difference being that, rather than considering a single $\mathcal{M}^R_h$ model, we will now need to consider two models $\mathcal{M}_{max}$ and $\mathcal{M}_{min}$. While the goal test for the original MCE only included an optimality test at $\mathcal{M}^R_h$, here we need to both check the optimality of the plan in $\mathcal{M}_{max}$ and verify the correctness of the plan in $\mathcal{M}_{min}$. As stated before, the plan is only optimal in the entire set of possible models if it satisfies both tests. Since the correctness of a given plan can be verified in polynomial time with respect to the plan size, this is a relatively easy test to perform. The other important point of difference between the search algorithms is how we calculate the applicable model updates. Here we consider the superset of the model difference between $\mathcal{M}^R$ and $\mathcal{M}_{min}$ and the difference between $\mathcal{M}^R$ and $\mathcal{M}_{max}$.

5 Balancing Explanation and Explicability

While plan explanations tries to resolve any confusions that the human observer may have about the plan, a related problem is that of generating explicable plans. In "explicable planning" a solution to the human-aware planning problem is a plan π^e such that it is executable (buy may no longer be optimal) in the robot's model (1) but is also close to the expected plan in the human's model (2) –

(1) $\delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi^e) \models \mathcal{G}^R$; and

(2) $\pi = \arg\min_{\pi^e}(dist(\pi_{\mathcal{M}^R_h}, \pi))$.

There are a number of plan distance metrics that one could consider here, including the difference in plan utility, action distance, causal link distance, etc. The exact metric may depend on the specific scenario being considered and may, in fact, be a combination of these individual metrics.

Plan explanation and explicability represent two possible ways a robot could potentially deal with model differences in such multi-model planning settings. While plan explanations actively try to reconcile the differences between these models, explicable planning seeks to generate plans that conform to the expectation of the human observing the robot, at the cost of using a potentially costlier plan. While in many cases any one of these approaches may suffice, there may be situations where a combination of both provide a much better course of action – if the expected human plan is too costly in the planner's model (e.g., the human might not be aware of some safety constraints) or the cost of communication overhead for explanations is too high (e.g., limited communication bandwidth). Consider, for example, a human working with a robot that has just received a software update allowing it to perform new complex maneuvers. Instead of directly trying to conceive all sorts of new interactions right away that might

end up spooking the user, the robot could instead reveal only certain parts of the new model while still using its older model (even though suboptimal) for the rest of the interactions so as to slowly reconcile the drifted model of the user. In such a setting the goal would be to design an algorithm capable of generating a plan π and an explanation $\mathcal{E}$ such that (i) π is executable in the robot's model, (ii) the explanation $\mathcal{E}$ is in the form of model updates, (iii) π is optimal[1] in the updated human model and (iv) π is selected by trading explicability cost and explanation with a hyperparameter α:

(1) $\delta_{\mathcal{M}^R}(\mathcal{I}^R, \pi) \models \mathcal{G}^R$;

(2) $\widehat{\mathcal{M}}_h^R \longleftarrow \mathcal{M}_h^R + \mathcal{E}$;

(3) $C(\pi, \widehat{\mathcal{M}}_h^R) = C^*_{\widehat{\mathcal{M}}_h^R}$; and

(4) $\pi = \arg\min_\pi \{ |\mathcal{E}| + \alpha \times | C(\pi, \mathcal{M}^R) - C^*_{\mathcal{M}^R} | \}$.

Clearly, with higher values of α, the planner will produce plans that require more explanations, while lower α values will generate more explicable plans. We can use a modified version of the model space search mentioned in explanation generation section to calculate the expected plan and explanations for any given α value. For each possible model we come across during our model space search, we test if the objective value of the new node is smaller than the current min node. The objective value reflects the combined cost of explaining the plan in the node and the additional cost the robot need to bear to execute a plan optimal within that node –

$$V = |\mathcal{E}| + \alpha \times | C(\pi^*_{\widehat{\mathcal{M}}}, \mathcal{M}^R) - C^*_{\mathcal{M}^R} |$$

So larger the α value, the more, the robot is concerned about the additional cost of executing a plan that is not optimal. We stop the search once we identify a model that is capable of producing a plan that is also optimal in the robot's model. This is different from the stopping condition used by the original MCE-search, where we were just trying to identify the first node where the given plan is optimal. An interesting effect of this stopping condition is the fact that even when the α value is high the search is guaranteed to compute the best possible plan for the planner as well as the smallest explanation associated with it. The details related to this work can be found in our paper [Sreedharan, Chakraborti, and Kambhampati, 2017a].

6 Conclusion and Future Work

The works described in the document presents our initial efforts at developing approaches to generating explanations for human in the loop planning. One of the biggest challenges in this direction is to create and maintain models of its human teammates that capture their capabilities, preferences, intentions, etc. – i.e. M_h^R(and even M_r^H). These models are inherently partial from the agent's perspective, and the problem becomes even harder in case of mental models as these cannot be learned directly from observed plan traces but through the interactions that the agent has with the human. An important assumption we made throughout the works described in this paper is that the human and robot understands the domain at the same level of abstraction. In most scenarios, the model that the human has access to would be some abstraction of the actual robot model, and the robot needs to keep this difference in mind while coming up with this explanation. [Sreedharan, Srivastava, and Kambhampati, 2018] presents some initial work we have done towards this direction. Additionally, we would also like to extend our approach also to support stochastic and decision-theoretic planning models.

[1] or ideally the most explicable, but for now we will limit our attention to cost difference as the only measure of explicability

References

[Bercher et al., 2014] Bercher, P.; Biundo, S.; Geier, T.; Hoernle, T.; Nothdurft, F.; Richter, F.; and Schattenberg, B. 2014. Plan, repair, execute, explain-how planning helps to assemble your home theater. In *ICAPS*.

[Chakraborti et al., 2017] Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*.

[Kambhampati, 1990] Kambhampati, S. 1990. A classification of plan modification strategies based on coverage and information requirements. In *AAAI 1990 Spring Symposium on Case Based Reasoning*. Citeseer.

[Miller, 2017] Miller, T. 2017. Explanation in artificial intelligence: Insights from the social sciences. *CoRR* abs/1706.07269.

[Nguyen, Sreedharan, and Kambhampati, 2017] Nguyen, T.; Sreedharan, S.; and Kambhampati, S. 2017. Robust planning with incomplete domain models. *Artificial Intelligence*.

[Seegebarth et al., 2012] Seegebarth, B.; Müller, F.; Schattenberg, B.; and Biundo, S. 2012. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

[Sreedharan, Chakraborti, and Kambhampati, 2017a] Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2017a. Balancing explicability and explanation in human-aware planning.

[Sreedharan, Chakraborti, and Kambhampati, 2017b] Sreedharan, S.; Chakraborti, T.; and Kambhampati, S. 2017b. Handling model uncertainty and multiplicity in explanations via model reconciliation.

[Sreedharan, Srivastava, and Kambhampati, 2018] Sreedharan, S.; Srivastava, S.; and Kambhampati, S. 2018. Hierarchical expertise-level modeling for user specific robot-behavior explanations. *arXiv preprint arXiv:1802.06895*.

Stochastic Goal Recognition Design

Christabel Wayllace
Computer Science and Engineering Department
Washington University in St. Louis
St. Louis, MO 63130-4899, USA
cwayllace@wustl.edu

1 Introduction

Discovering the objective of an agent based on observations of its behavior is a problem that has interested both AI and psychology researchers for many years [Schmidt *et al.*, 1978; Kautz, 1987]. In AI, this problem is known as *goal recognition* or, more generally, *plan recognition*, and it has been used to model a number of applications ranging from software personal assistants [Oh *et al.*, 2010; 2011a; 2011b]; robots that interact with humans in social settings such as homes, offices, and hospitals [Tavakkoli *et al.*, 2007; Kelley *et al.*, 2012]; intelligent tutoring systems that recognize sources of confusion or misunderstanding in students through their interactions with the system [McQuiggan *et al.*, 2008; Johnson, 2010; Lee *et al.*, 2012; Min *et al.*, 2014]; and security applications that recognize the plan or goal of terrorists [Jarvis *et al.*, 2005].

The traditional approach to solve goal recognition problems involves finding better and more efficient algorithms to infer the agent's objective from online collected observations, however, a newly formulated problem takes another perspective. The problem, proposed by Keren *et al.* [2014], is called *goal recognition design* (GRD) and it is intended to reduce the complexity of the online goal recognition task by performing an offline modification of the underlying environment that the agent operates in. The goal is to find the smallest set of modifications that *forces the agent to reveal its goal as early as possible*. This problem finds itself relevant in many of the same applications of goal recognition because, usually, the underlying environment can be easily modified.

Typically, a GRD problem has two components: (1) A measure of the efficacy of goal recognition and (2) a model of possible design changes one can make to the underlying environment. In the seminal work by [Keren *et al.*, 2014], they proposed the *worst case distinctiveness* (wcd) metric, which aims at capturing the maximum number of steps an agent can take without revealing its goal, as a measure of the goal recognition efficacy. Further, they considered only the removal of actions as possible design changes to the environment. This definition is made for the problem under three key assumptions:

- **Assumption 1**: The agents in the system will act optimally (i.e., agents will move along a shortest path to its goal);
- **Assumption 2**: The actions of the agents are deterministic; and
- **Assumption 3**: The environment is fully observable (i.e., both the state and the action of the agent are observable).

Since then, researchers have generalized the GRD problem to relax these assumptions [Keren *et al.*, 2015; 2016a; 2016b] and have also proposed alternative algorithms to solve it [Son *et al.*, 2016]. Additionally, [Keren *et al.*, 2016b] have proposed the refinement of sensors, which decreases the degree of observation uncertainty of the state and/or action of the agent, as a possible design change on the environment.

Our work aims to further extend the GRD problem with the objective to take into account the stochasticity and limitations present in the physical world, where the agents do not control the outcomes of their actions and/or the observer is limited by the quantity and quality of the sensors.

With this objective in mind we have proposed the *Stochastic GRD* (S-GRD) problem, where the outcomes of the agent's actions are stochastic [Wayllace *et al.*, 2016]. Aside from this relaxation, we have also proposed a new metric called *expected case distinctiveness* (*ecd*) [Wayllace *et al.*, 2017] , which weighs the possible goals based on their likelihood of being the true goal. Table 1 summarizes the generalizations, metrics, and possible designs of existing GRD models.

2 Goal Recognition Design (GRD) and Stochastic GRD (S-GRD)

A *Goal Recognition Design* (GRD) problem [Keren *et al.*, 2014] is represented as a tuple $P = \langle D, \mathbf{G} \rangle$, where

$D = \langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C} \rangle$ captures the domain information and $\mathbf{G}$ is a set of possible goal states of the agent. The elements in the tuple D are as they are described in MDPs, except that the transition function T is deterministic and the cost function C is restricted to positive costs.[1] We assume that the cost of all actions is 1 for simplicity.

The *worst case distinctiveness* (wcd) of a GRD problem is a metric representing the length of a longest sequence of actions that an agent can execute without revealing its goal.

The objective in GRD is to find a subset of actions such that if they are removed from the domain, the wcd of the re-

[1] The domain information D was originally described by Keren *et al.* [2014] using a classical planning model [Geffner and Bonet, 2013].

	Generalizations			Metrics		Possible Designs	
	Suboptimal Plans	Partially Obs. Env.	Stochastic Actions	*wcd*	*ecd*	Action Removal	Sensor Refinement
[Keren *et al.*, 2014]				✓		✓	
Son *et al.* (2016)				✓		✓	
Keren *et al.* (2015)	✓			✓		✓	
Keren *et al.* (2016a)	✓	✓		✓		✓	
Keren *et al.* (2016b)	✓	✓		✓		✓	✓
Wayllace *et al.* (2016)			✓	✓		✓	
Wayllace *et al.* (2017)			✓	✓	✓	✓	

Table 1: Properties of Current Goal Recognition Design Models

sulting problem is minimized. This optimization problem is subject to the requirement that the cost of cost-minimal plans to achieve each goal $g \in \mathbf{G}$ is the same before and after removing the subset of actions.

A *Stochastic Goal Recognition Design* (S-GRD) problem [Wayllace *et al.*, 2016] is an extension of a GRD problem that assumes the actions executed by the agent have stochastic outcomes. It is represented as a tuple $P = \langle D, \mathbf{G} \rangle$, where, like in GRDs, $D = \langle \mathbf{S}, s_0, \mathbf{A}, \mathbf{T}, \mathbf{C} \rangle$ captures the domain information and $\mathbf{G}$ is a set of possible goal states of the agent. The elements in the tuple D are as they are described in GRDs, except that the transition function T is now stochastic. The *worst case distinctiveness* (wcd) of problem P is the largest expected cost incurred by the agent over all non-distinctive policy prefixes. A non-distinctive policy prefix is an optimal policy common to a pair of goals.

Like in GRDs, the objective in S-GRD is to find a subset of actions $\hat{\mathbf{A}}^* \subset \mathbf{A}$ such that if they are removed from the set of actions $\mathbf{A}$, then the wcd of the resulting problem is minimized. This optimization problem is subject to the requirement that the expected cost of the optimal policies to achieve each goal $g \in \mathbf{G}$ is the same before and after removing the subset of actions and that the number of reduced actions is less than or equal to a user-defined parameter k.

The intuition behind the *worst case distinctiveness* (wcd) is that it measures the longest path (i.e., a path with the largest cost) an agent can take without revealing its goal. Therefore, a natural extension of this definition for S-GRD problems is that the wcd represents now the largest *expected* cost incurred by the agent before its goal is revealed.

Computing the wcd for S-GRD problems however, is not a straightforward extension of the wcd computation for deterministic GRD problems, as it has been shown in [Wayllace *et al.*, 2017]. While in the deterministic case the wcd is computed by finding the longest path of actions common to a *pair* of goals and then finding the maximum value among all pairwise combinations, in the stochastic case we need to consider all goals at the same time. Additionally, we also made another key observation: that the set of possible goals for a particular state can differ based on the observed path of the agent to that state. Therefore, the set of possible goals of the agent is not Markovian as it depends on the entire history of states visited. We have proposed a new augmented MDP structure that carries the history information and the wcd is found using VI-like algorithms. Additionally, we observed that the augmented state space of the augmented MDP can often be segmented into *strongly connected components* (SCCs) – each SCC contains the augmented states with the same set of possible goals, and the set of possible goals is non-increasing. Therefore, we also propose a TVI-like algorithm that uses Tarjan's algorithm [Tarjan, 1972] to segment the augmented state space into SCCs first before running VI on each SCC in reverse topological order. This should significantly speed up the solving time if there are large numbers of SCCs, but may have the opposite effect if there are few SCCs due to the overhead incurred by Tarjan's algorithm.

3 Expected-Case Distinctiveness (*ecd*)

An implicit assumption made by the *worst-case distinctiveness* (wcd) metric is that there is no prior information on which is the true agent's goal. While this assumption is reasonable in many problems, it may be the case that some information is available. For example, in human-computer interaction applications, user profiles may be used to assign different weights to each goal, where the weights correspond to the prior probabilities of an agent choosing its goal.

Further, it may often be the case where the wcd cannot be reduced (i.e., the longest non-distinctive path cannot be shortened). However, other shorter non-distinctive paths can be shortened. Thus, intuitively, one should prefer the solution that shortens the shorter non-distinctive paths. In such a scenario, the wcd metric fails to distinguish between these solutions as the wcd remains the same in both cases. This situation is further exacerbated when the longest path are to goals with low weights!

In response to these two observations, we proposed a new metric, called the *expected-case distinctiveness* (ecd), for S-GRDs that weighs the length of a path to a goal by the probability of an agent choosing that goal and takes the sum of all the weighted path lengths.

To reduce the wcd or ecd of a problem, we enumerate through all possible combinations of actions to remove, compute the resulting wcd or ecd, and store the best solution.

4 Future Work

We next plan to extend the S-GRD framework by assuming a more realistic situation for the observer, that is, to take into account first that the actions are usually non-observable, what

we really observe is the current state of the agent after an action is executed, specially in stochastic environments. We also plan to take into account the sensor (e.g., GPS) resolution, which may cause that several nearby states are indistinguishable. Thus, the state of the agent is only partially observable. Additionally, we want to incorporate new sensor refinement modifications in the design, to help us figure out which sensors need to improve their resolution in order to facilitate goal recognition.

Our plan for future work also includes the generalization of the ecd metric to the other GRD variants as well as the investigation of heuristic search techniques to compute the wcd and ecd values. We suspect that such techniques may be useful in cases one can prune significant portions of the search space due to the differences in the weights of the goals.

References

[Bellman, 1957] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Dai *et al.*, 2011] Peng Dai, Mausam, Daniel S Weld, and Judy Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence Research*, 42:181–209, 2011.

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

[Geib and Goldman, 2001] Christopher W Geib and Robert P Goldman. Plan recognition in intrusion detection systems. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 1, pages 46–55. IEEE, 2001.

[Jarvis *et al.*, 2005] Peter Jarvis, Teresa Lunt, and Karen Myers. Identifying terrorist activity with AI plan recognition technology. *AI Magazine*, 26(3):73–81, 2005.

[Johnson, 2010] W. Lewis Johnson. Serious use of a serious game for language learning. *International Journal of Artificial Intelligence in Education*, 20(2):175–195, 2010.

[Kautz, 1987] Henry A Kautz. *A Formal Theory of Plan Recognition*. PhD thesis, Bell Laboratories, 1987.

[Kelley *et al.*, 2012] Richard Kelley, Liesl Wigand, Brian Hamilton, Katie Browne, Monica Nicolescu, and Mircea Nicolescu. Deep networks for predicting human intent with respect to objects. In *Proceedings of the International Conference on Human-Robot Interaction (HRI)*, pages 171–172, 2012.

[Keren *et al.*, 2014] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 154–162, 2014.

[Keren *et al.*, 2015] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design for non-optimal agents. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3298–3304, 2015.

[Keren *et al.*, 2016a] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design with non-observable actions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3152–3158, 2016.

[Keren *et al.*, 2016b] Sarah Keren, Avigdor Gal, and Erez Karpas. Privacy preserving plans in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3170–3176, 2016.

[Lee *et al.*, 2012] Seung Lee, Bradford Mott, and James Lester. Real-time narrative-centered tutorial planning for story-based learning. In *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS)*, pages 476–481, 2012.

[Mausam and Kolobov, 2012] Mausam and Andrey Kolobov. *Planning with Markov Decision Processes: An AI Perspective*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[McQuiggan *et al.*, 2008] Scott McQuiggan, Jonathan Rowe, Sunyoung Lee, and James Lester. Story-based learning: The impact of narrative on learning experiences and outcomes. In *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS)*, pages 530–539, 2008.

[Min *et al.*, 2014] Wookhee Min, Eunyoung Ha, Jonathan Rowe, Bradford Mott, and James Lester. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, pages 37–43, 2014.

[Oh *et al.*, 2010] Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy Norman. ANTIPA: An agent architecture for intelligent information assistance. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 1055–1056, 2010.

[Oh *et al.*, 2011a] Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy Norman. An agent architecture for prognostic reasoning assistance. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2513–2518, 2011.

[Oh *et al.*, 2011b] Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy Norman. Probabilistic plan recognition for intelligent information agents: Towards proactive software assistant agents. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, pages 281–287, 2011.

[Schmidt *et al.*, 1978] Charles Schmidt, N. Sridharan, and John Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(1–2):45–83, 1978.

[Son *et al.*, 2016] Tran Cao Son, Orkunt Sabuncu, Christian Schulz-Hanke, Torsten Schaub, and William Yeoh. Solving goal recognition design using ASP. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3181–3187, 2016.

[Tarjan, 1972] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[Tavakkoli *et al.*, 2007] Alireza Tavakkoli, Richard Kelley, Christopher King, Mircea Nicolescu, Monica Nicolescu, and George Bebis. A vision-based architecture for intent

recognition. In *Proceedings of the International Symposium on Advances in Visual Computing*, pages 173–182, 2007.

[Wayllace *et al.*, 2016] Christabel Wayllace, Ping Hou, William Yeoh, and Tran Cao Son. Goal recognition design with stochastic agent action outcomes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3279–3285, 2016.

[Wayllace *et al.*, 2017] Christabel Wayllace, Ping Hou, and William Yeoh. New metrics and algorithms for stochastic goal recognition design problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4455–4462, 2017.

Extended abstract: SMT Encoding for Planning for Hybrid Systems

Parisa Zehtabi
Kings College London, London, WC2R 2LS
parisa.zehtabi@kcl.ac.uk

1 Introduction

PDDL+, as the augmented version of PDDL2.1, was designed to model real-world applications, through continuous processes and exogenous events. As the consequence of this, a number of approaches have been developed that can handle *subsets* of PDDL+. In the next section, hybrid systems and the importance of modeling this group of problems are briefly explained. Also, a short introduction on PDDL+ is given and later the limitations and challenges that the current planners have to deal with, regarding all PDDL+ features, are discussed. Finally, we present a new approach for solving the PDDL+ models which is able to deal with all PDDL+ characteristics respects to (Fox and Long 2003).

2 Background

2.1 Hybrid Systems

A hybrid system is one in which there are both continuous variables and discrete logical modes of operation. It represents a powerful model to describe the dynamic behaviour of modern engineering artefacts. Hybrid systems frequently occur in practice, e.g., in robotics or embedded systems. Some example applications include coordination of activities of a planetary lander, oil refinery management, autonomous vehicles, chemical plant (Della Penna et al. 2010), smart grid (Campion et al. 2013), and battery management (Fox, Long, and Magazzeni 2011). Such scenarios motivate the need to reason with mixed discrete-continuous domains.

2.2 PDDL+ Planning

PDDL+ uses processes and events to model mixed discrete-continuous problems. Processes and events are triggered automatically as their preconditions have been satisfied, however it is the planner's decision to choose whether an action is executed or not, and as a result of applying an action, processes or events can be triggered. If one event $e1$ is triggered, and the effects of it can satisfy the preconditions of another event $e2$, and all of their effects fulfilling the preconditions of a process $p1$, then $e1$, $e2$, and the start of $p1$ happen at the same time-point. It is due to this behaviour that PDDL+ semantics place a bound on the number of cascading (parallel) events.

A PDDL+ planning problem is a tuple $\Pi+ := \langle P, V, A, Ps, E, I, G \rangle$ where P, V and A are the sets of all propositions, real variables and actions respectively. The set of processes and events are shown by Ps and E. Moreover, $I(P, V)$ and $G(P, V)$ represent the initial and the goal state. Considering that the PDDL+ semantics can model the hybrid domains, finding an efficient planner that can handle events and processes is the next crucial step.

2.3 Planning as SMT

Different approaches have been used to overcome the challenges that planning is facing. The method that is explained in this report is based on the satisfiability modulo theory (SMT). SMT is the problem of deciding the satisfiability of a first-order formula expressed in a given theory. In this method, the numeric values are encoded based on a propositional logic. As the consequence of this encoding, and also the efficiency of the propositional (boolean) satisfiability problem (SAT) solvers, SMT relates the theory specific solver to the SAT solvers. Later we will explain in more detail how we have used this method.

3 Related Work

Various techniques and tools have been proposed to deal with hybrid domains (Penberthy and Weld 1994; McDermott 2003; Li and Williams 2008; Coles et al. 2012; Shin and Davis 2005). These methods are described in more detail in (Ghallab, Nau, and Traverso 2004), so the rest of this section focuses on the class of planning for dealing with hybrid systems. More recent approaches in this direction have been proposed by (Bogomolov et al. 2014), where the close relationship between hybrid planning domains and hybrid automata is explored, and (Bryce et al. 2015) where hybrid domains are handled using SMT. Nevertheless, none of these approaches are able to handle the full set of PDDL+ features, namely nonlinear domains with processes and events. To date, the only viable approach in this direction is PDDL+ planning via discretisation. UPMurphi (Della Penna, Magazzeni, and Mercorio 2012), which implements the discretise and validate approach, is able to deal with the full range of PDDL+ features. However, it only does blind search, which limits its scalability.

3.1 PDDL+ as SMT

Planning through SMT is not new. A recent approach, dReach, described in (Bryce et al. 2015), uses a non-linear SMT solver for planning in hybrid systems. That approach is promising, although it suffers some important limitations. Firstly, it does not use PDDL+, as it relies on the language of dReach, in which the hybrid problems have to be manually encoded. Secondly, dReach can only handle a restricted subset of the language features contained in PDDL+, and, in particular, it cannot handle events. Thirdly, dReach is tailored to be used only with the dReal solver (Gao, Avigad, and Clarke 2012).

In the next section the new SMT encoding for PDDL+ which overcomes these limitations is discussed. This encoding is able to capture all features of PDDL+ (including events and the ϵ-separation of effects and action preconditions (Fox and Long 2006)) and works by directly translating standard PDDL+ domain and problem files. The output of the translation is a standard SMT encoding that can be used with any SMT solver in the theory of quantifier-free nonlinear arithmetic (QF_NRA). Furthermore, this approach proves to also be efficient in proving plan-non-existence, along with dramatically improving over dReach in solvable problems. In terms of the dynamics, this approach is complete for nonlinear polynomial change.

4 Encoding of PDDL+ Domains

The encoding of a PDDL+ problem that is described in this section is based on the current encoding of (Cashmore et al.). The notion of *happening* has been introduced to capture the change in the state at a given time point due to the effects of actions, processes, or events. Namely, each happening encodes the causal chain of events, processes and instantaneous actions which might occur simultaneously at a given time point. As has been explained earlier, a bound B has been defined as the length of the causal cascading instantaneous events.

4.1 Happening

A happening is the tuple $x_t := \{P_t, V_t, E_t, Ps_t, A_t\}$, where the variables of the happening are defined as follows:

- $P_t = \{P_{0,t}, \ldots, P_{B,t}\}$ represents the causal change in the set of propositional state variables $P_{i,t}$ where $i \in \{0, ..., B\}$ at time t;
- $V_t = \{V_{0,t}, \ldots, V_{B,t}\}$ represents the causal change in the real state variables $V_{i,t}$ where $i \in \{0, ..., B\}$ at time t;
- $E_t = \{E_{0,t}, \ldots, E_{B,t}\}$ represents the chain of set of events, $E_{i,t}$ where $i \in \{0, ..., B\}$, triggered at time t;
- Ps_t represents the set of active processes at time t;
- A_t is the set of actions applied at time t.

An example of happening is shown in Figure 1. The bound for the chain of events limits the number of layers which are represented as circles in this figure.

As has been explained earlier, a happening describes a moment of discrete change, corresponding to the discrete transition *Trans* of the hybrid systems. Between happenings, there is only continuous numeric change (*Flow*). The key difference is that: multiple actions can be performed in a single happening in parallel, meaning that while the hybrid systems model is exponential in the size of the PDDL+ description, our encoding will be linear. Having defined happenings, a PDDL+ model can be described as a bounded set of happenings $X := \{x_{t_1} ... x_{t_n}\}$ encoded as an SMT formula, such that any proof for the SMT formula represents the trace of a valid plan for $\Pi+$. The *plan* corresponding to that trace is the set of action assignments $A_{t_1} \cup ... \cup A_{t_n}$.[1] The following describes the encoding of a single happening, and later the encoding of the formula for a planning problem.

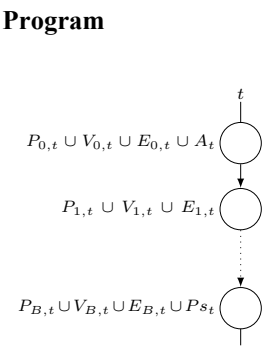


Figure 1: A single happening occurs at time t. The circles (layers) describe a causal chain of instantaneous events.

4.2 Encoding of a single Happening

Following the definition of a happening, a happening x_t is encoded as:

$$x_t := \left\langle \begin{array}{l} t, P_{0,t}, ..., P_{B,t}, A_t \\ V_{0,t}, ..., V_{B,t}, flow_{V_t} \\ E_{0,t}, ..., E_{B,t}, Ps_t, dur_{Ps_t} \end{array} \right\rangle$$

The set $flow_{V_t} := \{flow_{v_t} | \forall v_t \in V_t\}$ is a numerical expression that represents the change in value of v from this time point to the next. Finally, $dur_{Ps_t} := \{dur_{ps_t} | \forall ps_t \in Ps_t\}$ represents the remaining duration of each process. The constraints within a happening are classified as following.

Proposition and Real Variable support: The following constraint ensures that the propositional value remains consistent at the first layer of a happening.

$$\bigwedge_{p_{i,t} \in P} p_{1,t} \rightarrow (p_{0,t} \vee \bigvee_{e_{i,t} | p_{i,t} \in eff^+_{e_{i,t}}} e_{0,t} \vee \bigvee_{a_t | p_{i,t} \in eff^+_{a_t}} a_t)$$

Similarly, other constraints are generated that ensure the value of propositions and real variables remain consistent from $P_{0,t} \cup V_{0,t}$ to $P_{B,t} \cup V_{B,t}$.

Event Preconditions and Effects: This set of constraints enforce that an event is triggered if and only if its precondition holds.

$$\bigwedge_{i=0}^{B} \bigwedge_{e_{i,t} \in E} e_{i,t} \leftrightarrow p_{i,t} \in pre_{e_{i,t}}$$

[1] As processes and events do not appear in a PDDL+ plan.

The equation below ensures that the effects of events should be presented in the next layer of that happening.

$$\bigwedge_{i=0}^{B-1} \bigwedge_{e_{i,t} \in E} e_{i,t} \rightarrow p_{i,t+1} \in eff_{e_{i,t+1}}$$

Action Preconditions and Effects: Similar to Event Preconditions and Effects constraints, these equations ensure the same rules apply for action variables in A_t.

Process Triggering: Similar to the Event Preconditions, this group of constraints enforces that a process is active if and only if its preconditions are satisfied in set $P_{B,t} \cup V_{B,t}$. It also enforce that the real variable dur_{ps_t} for each process is greater than or equal to zero, and that happens if and only if the process is active. These constraints will be used to ensure that a process cannot finish outside of a happening.

Action Mutexes: This constraint includes a collection of binary constraints, enforcing that no two mutex actions can be applied simultaneously.

$$\bigwedge_{a_t \in A} \bigwedge_{a'_t \in A | a_t \nparallel a'_t} (\neg a_t \vee \neg a'_t)$$

4.3 Encoding of a Planning Problem

The existence of a plan for a *PDDL+ planning problem (*$\Pi+$*) with bound* n is proved by building the SMT formula ($\Pi+_n$) in the theory of quantifier-free (nonlinear) real arithmetic with n copies of the set of variables over the set of happenings $T = \{t_1, ..., t_n\}$ for $n \geq 1$. The encoding is illustrated by Figure 2.

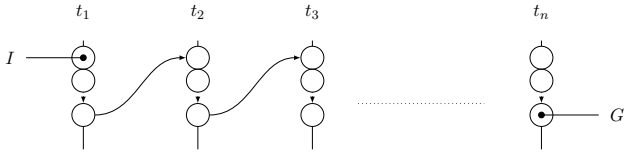


Figure 2: A plan is found by building a formula with n copies of the set of variables over the happenings t_j: $t_1 ... t_n$. The initial state is modeled in t_1, and the goal constraints are added to t_n.

The constraints for a happening are copied for each happening $t_1, ..., t_n$. Additional constraints in the SMT formula $\Pi+_n$ are shown in the following.

Instance description: These constraints enforce the initial state to hold in the first happening, and that the goal is achieved in the final happening.

$$I(P_{0,t_1} \cup V_{0,t_1}), G(P_{B,t_n} \cup V_{B,t_n})$$

Also we need to constrain the timing of happenings to respect the epsilon separation.

$$\bigwedge_{i=2}^{n} t_i \geq t_{i-1} + \epsilon$$

Proposition support: These constraints ensure that the discrete state variables p_{i,t_j} do not change between happenings.

$$\bigwedge_{j=2}^{n} \bigwedge_{p_{i,t_j} \in P_{i,t_j}} p_{0,t_j} \rightarrow p_{B,t_{j-1}}$$

Invariant: These constraints ensure that the continuous numeric change between happenings is valid. The following equation ensures that if a process is active in the previous happening, its duration is decreased by the time between happenings. This constraint, in combination with the Progress Triggering constraints in the previous section ensures that a process cannot end between happenings.

$$\bigwedge_{j=2}^{n} \bigwedge_{ps_{t_j} \in Ps_{t_j}} ps_{t_{j-1}} \rightarrow dur_{ps_{t_j}} = dur_{ps_{t_{j-1}}} + t_j - t_{j+1}$$

Furthermore, the following constraint enforces the invariant of the process.

$$\bigwedge_{j=1}^{n-1} \bigwedge_{ps_{t_j} \in Ps_{t_j}} ps_{t_i} \leftrightarrow pre_{\leftrightarrow ps_{t_j}}$$

If a process is active, the preconditions of the process are active over the whole interval between the happenings. Violating these preconditions during the execution of a process is known as zero cross problem. Hence, it is crucial to check this during the execution of a process. For constraints over real valued variables, this is done by checking the value either side of the interval. However for the nonlinear changes it is possible that the preconditions are satisfied at the sides of the interval, but they are violated between the happenings. As is shown in Figure 3, the red section highlights the situation where the preconditions are violated within the interval. Therefore, the following constraint should be added to make sure that a process won't violate its preconditions.

$$ps_{t_j} \rightarrow \bigwedge_{a_{t_j}=1}^{A_{t_j}} \left((\frac{d^a f}{dt^a})_j (\frac{d^a f}{dt^a})_{j+1} >= 0 \right)$$

where f is the numeric, non-constant part of the invariant, and where the $(A+1)$th derivative of f is identically zero. This constraint ensures that the derivatives of the function do not cross zero over the interval, thus a fluctuating value of f cannot violate the invariant condition between t_j and t_{j+1}. Therefore, any solution must include a happening at each point a derivative crosses zero.

Similarly an additional constraint ensures that an event is not triggered during an interval.

Continuous Change on Real Variables: The model is enforced to calculate the change over the interval and apply it to each real variable. In order to calculate the change, the indefinite integral of the process' effects upon the variable must be computed.

$$\bigwedge_{j=1}^{n-1} \bigwedge_{v_{t_j} \in V_{t_j}} flow_{v_{t_j}} = \int_{t_j}^{t_{j+1}} \sum_{ps_{t_j} \in Ps_{t_j}} eff^{num}_{\leftrightarrow ps_{t_j}}[v_{i,t_j}] dt$$

Note that in this approach, the integration and differentiation are performed by SymPy (SymPy 2013) outside the solver,

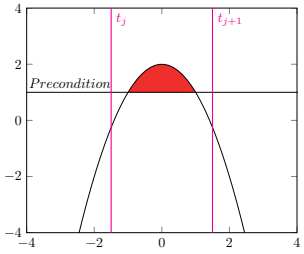


Figure 3: Zero crossing problem for nonlinear changes; ensures the preconditions of a process will hold over the execution of that process

during the encoding. Hence the integration is done only once for each domain.

5 PhD Road Map

SMTPlan encodings for planning in hybrid systems is quite new. Thus there are many opportunities for improvement. We envisage two main streams of future plans, as described in the following.

5.1 Scalability of SMT-based Planning

- **Heuristics:** Considering the fact that SMTPlan does not use any heuristic searches, we are exploring new heuristics to improve the efficiency and scalability of the planner.
- **Improving the Discrete Scalability:** Based on the experiments, SMTPlan is able to deal with complex continuous numerical dynamics. However this encoding is not effective on the discrete state explosion. Improving the discrete scalability of the encoding is the main focus in our next steps.

5.2 Improving the Encoding and Robustness

- **Changing to the Change-based Encoding:** Currently, this encoding models all the boolean and numerical variables and check the satisfiability of the encoded constraints at each happening. The value of these variables won't change unless they become manipulated by the affects of the actions, processes or events. Considering this, we are aiming to only use the set of variables who's values are manipulated at each step in our modeling. This will dramatically reduce the number of variables in our model.
- **Robust Envelops:** For this step we are planning to define how much the suggested plan by our encoding, is flexible. The model used for planning is normally not completely adherent to the ground reality in which the plan is supposed to operate. Considering this fact, we are planning to find a new flexible plan that is more robust in different planning conditions. Furthermore, we are interested in defining a range for the propositions of the world in which the suggested plan remains valid.

References

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle, M. 2014. Planning as model checking in hybrid domains. In *Proceedings of AAAI*.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P. 2015. SMT-based nonlinear PDDL+ planning. In *Proceedings of AAAI*.

Campion, J.; Dent, C.; Fox, M.; Long, D.; and Magazzeni, D. 2013. Challenge: Modelling unit commitment as a planning problem. In *Proceedings of ICAPS*.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. A compilation of the full PDDL+ language into SMT. In *Proceedings of ICAPS*.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2012. COLIN: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research (JAIR)* 44:1–96.

Della Penna, G.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2010. A PDDL+ benchmark problem: The batch chemical plant. In *Proceedings of ICAPS*.

Della Penna, G.; Magazzeni, D.; and Mercorio, F. 2012. A universal planning system for hybrid domains. *Applied Intelligence* 36(4):932–959.

Fox, M., and Long, D. 2003. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *Jorunal of Artificial Intelligence Research (JAIR)* 27:235–297.

Fox, M.; Long, D.; and Magazzeni, D. 2011. Automatic construction of efficient multiple battery usage policies. In *Proceedings of IJCAI*.

Gao, S.; Avigad, J.; and Clarke, E. M. 2012. Delta-complete decision procedures for satisfiability over the reals. In *Proceedings of IJCAR*.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: theory and practice*. Elsevier.

Li, H. X., and Williams, B. C. 2008. Generative planning for hybrid systems based on flow tubes. In *Proceedings of ICAPS*.

McDermott, D. V. 2003. Reasoning about autonomous processes in an estimated-regression planner. In *Proceedings of ICAPS*.

Penberthy, J. S., and Weld, D. S. 1994. Temporal planning with continuous change. In *Proceedings of AAAI*.

Shin, J.-A., and Davis, E. 2005. Processes and continuous change in a sat-based planner. *Artificial Intelligence* 166(1).

SymPy. 2013. Website. http://www.sympy.org/.