Decreasing Uncertainty in Planning with State Prediction*

Senka Krivic[1], Michael Cashmore[2], Daniele Magazzeni[2], Bram Ridder[2], Sandor Szedmak[3], and **Justus Piater[1]**
[1]Department of Computer Science, University of Innsbruck, Austria
[2]Department of Computer Science, King's College London, United Kingdom
[3]Department of Computer Science, Aalto University, Finland
[1]name.surname@uibk.ac.at, [2]name.surname@kcl.ac.uk, [3]name.surname@aalto.fi

Abstract

In real world environments the state is almost never completely known. Exploration is often expensive. The application of planning in these environments is consequently more difficult and less robust. In this paper we present an approach for predicting new information about a partially-known state. The state is translated into a partially-known multigraph, which can then be extended using machine-learning techniques. We demonstrate the effectiveness of our approach, showing that it enhances the scalability of our planners, and leads to less time spent on sensing actions.

1 Introduction

Planning for real world environments often means planning with incomplete and uncertain information. For example, in robotic domains and other dynamic environments there can be large numbers of unknown areas and objects. Moreover, in dynamic environments planning has to be completed quickly. However, exploration and observation in these scenarios can be costly. This uncertainty has severe consequences for planning. The planning problem becomes more complex, taking longer to solve, and exacerbating the issue of scalability.

Some uncertainty can be handled during the planning process with, for example, contingency planning [Bonet and Geffner, 2000; Hoffmann and Brafman, 2005], conformant planning [Smith and Weld, 1998; Palacios and Geffner, 2006], probabilistic planning [Camacho *et al.*, 2016], or replanning techniques [Brafman and Shani, 2014]. These approaches come with an associated cost in terms of complexity and robustness. Moreover, if the lack of information is severe, then the problem can be rendered unsolvable with current approaches.

We enhance the standard procedure of planning by adding the state prediction step between sensing and planning as shown in the Figure 1.

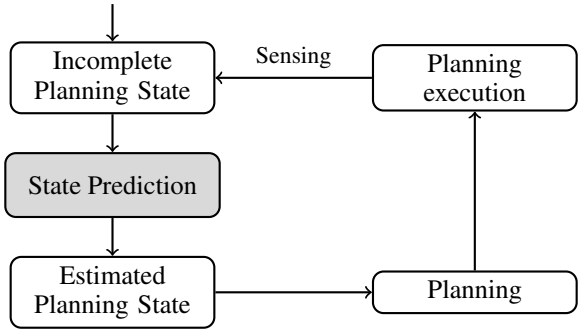

Figure 1: Proposed approach for decreasing uncertainty in planning. Partially-known states are updated through both sensing actions and prediction.

We propose an approach to reduce the uncertainty by making predictions about the state. We translate the state to a partially-known multigraph to enable prediction of the state with machine learning techniques. In particular, we exploit the method for prediction in partially-known multigraphs presented in [Krivic *et al.*, 2015]. Missing edges are predicted by exploiting the similarities between known properties. We introduce a confidence measure for these edges. Edges with high prediction confidence are translated back to the state, decreasing uncertainty.

The resulting problem is consequently less costly to solve and plans generated require fewer sensing actions. The resulting problem can be passed to any kind of planning system. Incorrect predictions can increase the chance of plan failure during execution. We show in our evaluation, over a range of planning domains, the accuracy of the predictions is very high, and these cases are rare.

We integrated the prediction with a planning and execution system. Our system uses an adapted version of *Maximum Margin Multi-Valued Regression* (M^3VR) [Krivic *et al.*, 2015] for learning missing edges. The planner CLG [Bonet and Geffner, 2000] is used to solve the resulting planning problems. We demonstrate in real scenarios, when contingency planning with CLG, prediction increases scalability, allowing the solver to tackle many more problems at an acceptable cost to robustness.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 610532, SQUIRREL.

The paper is structured as follows. In Section 2 we present the problem formulation, and how a state can be represented as a partially-known multigraph. In Section 3 we describe the prediction process that acts upon a multigraph. In Section 4 we present the experimental results and we conclude in Section 5.

2 Predictions in the Planning Problem

In this section we describe in detail the problem of state prediction.

Definition 1 (Planning Problem) *A planning instance* Π *is a pair* $\langle D, P\rangle$*, where domain* $D = \langle Pred, A, arity\rangle$ *is a tuple consisting of a finite set of predicate symbols* $Pred$*, a finite set of (durative) actions* A*, and a function* $arity$ *mapping all symbols in* $Pred$ *to their respective arities. The triple describing problem* $P = \langle O, s', G\rangle$ *consists of a finite set of domain objects* O*, the partial state* s'*, and the goal specification* G*.*

The atoms of the planning instance are the (finitely many) expressions formed by grounding; applying the predicate symbols $Pred$ to the objects in O (respecting arities). The resultant expressions are the set of propositions $Prop$.

A state s is described by a set of literals formed from the propositions in $Prop$, $\{l_p, \neg l_p, \forall p \in Prop\}$. If every proposition from $Prop$ is represented by a literal in the state, then we say that s is a *complete state*.

A *partial state* is a set of literals $s' \subset s$, where s is a complete state. A partial state s' can be *extended* into a more complete state s'' by adding literals.

Definition 2 (Extending a Partial State) *Let* s' *be a partial state of Planning problem* Π*. Extending the state* s' *is a function* $Extend(\Pi, s') : s' \rightarrow s''$ *where* $s' \subseteq s''$ *and* $s'' \subseteq s$*.*

We describe a processing step implementing $Extend$. In order to be able to apply a machine learning technique to extend an incomplete state, we first translate it to a multigraph. Thus, the function $Extend(\Pi, s')$ is implemented as follows: (1) the partial state s' is converted into a multigraph; (2) edges in the multigraph are learned using M^{3}VR ; (3) the new edges are added as literals to the partially-known state.

2.1 Constructing the Multigraph

We represent a partially-known state s' as a partially-known multigraph M'.

Definition 3 (Partially-known Multigraph) *A partially-known multigraph* M' *is a pair* $\langle V, E'\rangle$*, where* V *is a set of vertices, and* E' *a set of values of directed edges.*

The values assigned to all possible edges are $\{0, 1, ?\}$ corresponding to $\{$*not-existing*, *existing*, *unknown*$\}$. We use E' to denote a set of edge values in a partially-known multigraph, while E denotes the set of edges values in a completed multigraph M. The partial state *s'* is described as a partially-known multigraph with an edge for each proposition $p \in P$ that is either unknown or known to be true. That is:

$$\begin{aligned} V &\equiv O \\ E' &= \{e_{pred}(b,u) | (b,u) \in V \times V\} \end{aligned} \quad (1)$$

The existence of a *directed edge* $e_{pred}(b, u)$ between two vertices b and u for a predicate $pred$ is described by the function $L_{pred} : V \times V \rightarrow \{0, 1, ?\}$. Edges are directed in the order the object symbols appear in the proposition.

For example, let b and u be two vertices in set V. For proposition p involving objects b and u, $L_{pred}(b, u) = 0$ if $\neg l_p \in s'$, $L_{pred}(b, u) = 1$ if $l_p \in s'$, and $L_{pred}(b, u) =?$ otherwise.

For an origin object b and a destination object u, we define the *edge-vector* as the vector:

$$\mathbf{e}_{bu} = [L_{pred}(b, u), \forall pred]$$

This vector describes the existence of all edges directed from b to u. This is illustrated in an example below.

2.2 Example

Consider the problem where a robot is able to move between waypoints, pick up, push, and manipulate objects, and put them in boxes. The predicates `can-pickup`, `can-push`, `can-stack-on`, and `can-fit-inside` describe whether it is possible to perform certain actions upon objects in the environment.

In this problem we restrict our attention to four objects: `robot`, `cup01`, `box01`, and `block01`. In PDDL 2.1 [Fox and Long, 2003] literals that are absent from the state are assumed to be false. However, we assume those literals to be unknown.

A graph M is generated, the vertices of which are $O := \{robot, cup01, box01, block01\}$ (Figure 2). An edge-vector example in this graph is given with:

$$\mathbf{e}_{robot,block01} = \begin{bmatrix} L_{can-fit-inside}(robot, block01) \\ L_{can-stack-on}(robot, block01) \\ L_{can-push}(robot, block01) \\ L_{can-pickup}(robot, block01) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ ? \end{bmatrix}$$

This example edge-vector describes the edge between the nodes *robot* and *block01*. In the example state it is known that the propositions `(can-fit-inside robot block01)` and `(can-stack-on robot block01)` are false. It is also known that the proposition `(can-push robot block01)` is true. Finally, `(can-pickup robot block01)` is unknown.

3 Predicting Missing Edges in a Multigraph

Once a multigraph is created, a machine learning method can be used for completing a multigraph [Cesa-Bianchi *et al.*, 2013; Gentile *et al.*, 2013; Latouche and Rossi, 2015]. In our system we use the Maximum Margin Multi-Valued Regression (M^{3}VR) which was used at the core of a recommender system [Ghazanfar *et al.*, 2012] and for an affordance learning problem [Szedmak *et al.*, 2014]. Their results show that it can deal with sparse, incomplete and noisy information. Moreover, Krivic et al. [2015] use M^{3}VR to refine spatial relations for planning to tidy up a child's room. We build on this, describing how the approach can be generalised for use in planning domains.

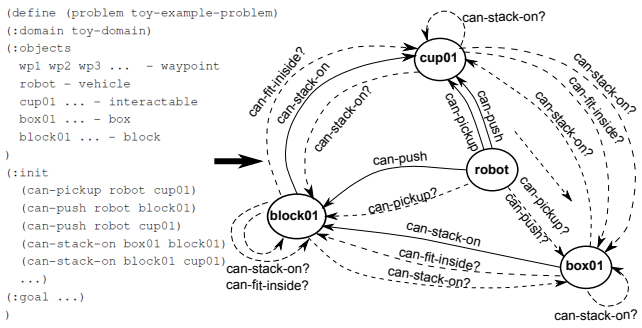

Figure 2: A fragment of an example problem from the `tidy-room` domain translated to the multigraph. Known edges are denoted by solid lines and unknown ones by dashed lines.

The structure of the multigraph represents the structure of the domain. Edges in the multigraph are partially known and thus this is a supervised learning problem. The main idea of M^{3}VR is to capture this hidden structure in the graph with a model and use it to predict missing edges. This model is represented by a collection of predictor functions that learn mappings between edges and vertices.

Edges are directed. Therefore we differentiate between origin and destination vertices for each edge. B denotes the set of origin vertices and U the set of destination vertices, where $B = U = V$. Edges between the vertices describe a relation between the sets B and U. To predict a missing edge between two vertices, knowledge about other known edges involving those vertices can be exploited (Figure 3). This concept extends to predictions for n-ary relations.

Using M^{3}VR we construct a function which captures knowledge about existing edges. This is done by assigning a predictor function f_b to each origin vertex $b \in B$ that maps all destination vertices to corresponding edges. Thus the number of predictor functions is equal to the number of vertices.

These predictor functions have to capture the underlying structure of a graph which can be very complex and non-linear. Thus it can be very hard to define in Euclidean space. Therefore these functions are defined on feature representations of vertices and edges. Thus, we choose:

- A function ψ that maps the edges into a Hilbert space H_ψ.
- Another function ϕ that maps the destination vertices $u \in U$ into the Hilbert space H_ϕ which can be chosen as a product space of H_ψ.

H_ϕ and H_ψ are feature representations of the domains of U and E. The vectors $\phi(\cdot)$ and $\psi(\cdot)$ are called *feature vectors*. This allows us to use the inner product as a measure of similarity.

Now prediction functions for each origin vertex b can be defined on feature vectors, i.e., $F_b : H_\phi \rightarrow H_\psi$. We assume that there is a linear relationship in feature space between the feature vector of all destination vertices H_ϕ and the feature vector of all connected edges H_ψ. This is represented by linear operator $\mathbf{W}_b$. The non-linearity of the mapping f_b can be expressed by choosing non-linear functions ψ and ϕ.

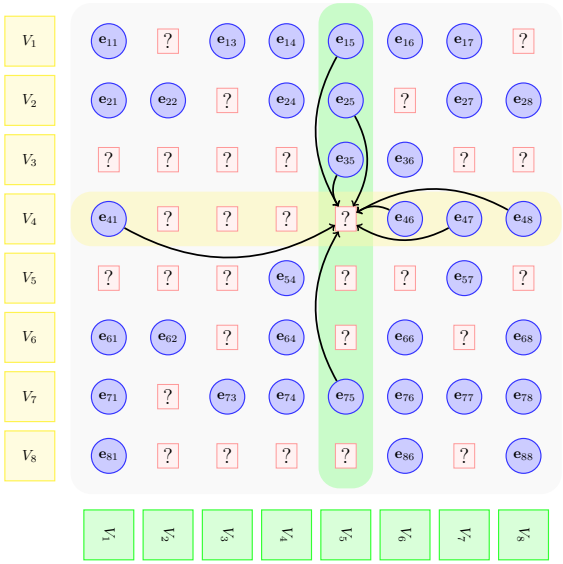

Figure 3: The core mechanism of M^{3}VR to predict a missing edge based on the available information on the object relation graph. For simplicity in this example, origin vertices are represented as rows and destination vertices as columns for a single predicate $pred$. The edge $e_{pred}(b = 4, u = 5)$ is missing. To predict the existence of the edge $e_{pred}(b, u)$, those edges that share the same origin or destination vertices as the missing edge can be exploited.

To exploit the knowledge about existing edges linking the same destination vertices $u \in U$, predictor functions for origin vertices are coupled by shared slack variables representing the loss to be minimized by the learners F_b. In this way, knowledge about existing edges is exploited to train prediction functions. Once determined, the linear mappings $\mathbf{W}_b$ allow us to make predictions of missing edges for elements b.

The similarity between the vectors $\mathbf{W}_b\phi(u)$ and $\psi(\mathbf{e}_{bu})$ is described by the inner product $\langle \psi(\mathbf{e}_{bu}), \mathbf{W}_b\phi(u) \rangle_{H_\psi}$. If the similarity between $\mathbf{W}_b\phi(u)$ and $\psi(\mathbf{e}_{bu})$ is higher, the inner product will have a greater value. As a consequence $\psi(\mathbf{e}_{bu})$ can be predicted as

$$\psi(\mathbf{e}_{bu}) \leftarrow \mathbf{W}_b\phi(u), (b, u) \in B \cap U. \tag{2}$$

Detailed descriptions of this procedure can be found in related work [Krivic *et al.*, 2015; Ghazanfar *et al.*, 2012].

In the optimization procedure for determining linear mappings there are as many constraints as the number of known edges in E'. Therefore the complexity of the prediction problem is equal to $O(|E'|)$, where $|E'|$ stands for the cardinality of the set E'.

The value of the inner product of the edge feature vector $\psi(\mathbf{e}_{bu})$ and $\mathbf{W}_b\phi(u)$ should be interpreted as a measure of confidence of the edge belonging to a specific class (in this case 0 or 1):

$$\text{conf}\{L^*_{pred}(b, u) = k\} = \langle \psi(\mathbf{e}_{bu} | L^*_{pred}(b, u) = k), \mathbf{W}_b\phi(u) \rangle_{H_\psi}$$

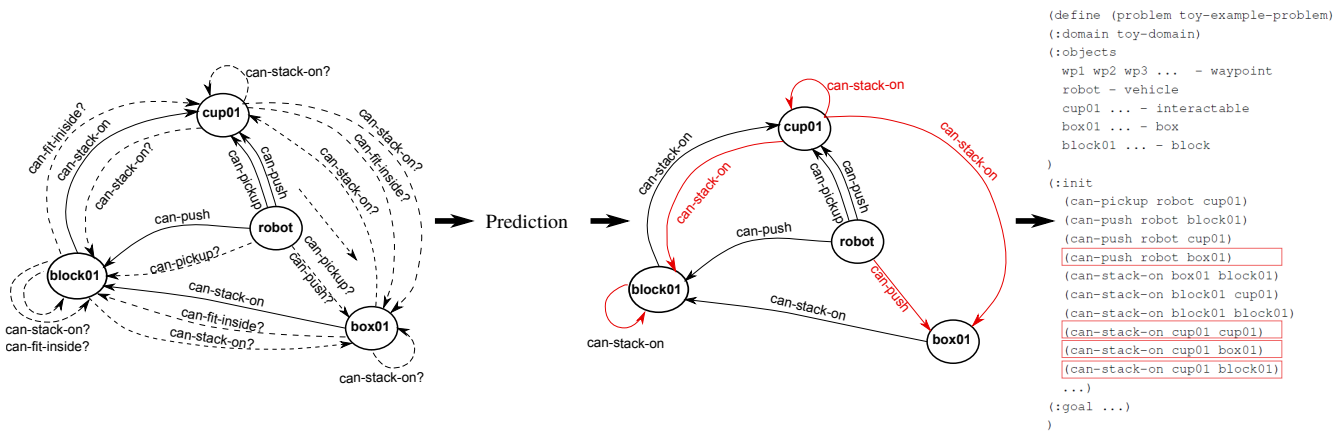

Figure 4: Extending the state. Example problem. After predictions, new literals are added.

where $k \in \{0, 1\}$, and $L^*_{pred}(b, u)$ is an unknown value in $\mathbf{e}_{bu} \in E \setminus E'$ of predicate $pred$.

For each prediction $L^*_{pred}(b, u)$, we update the existence of the directed edges:

$$L_{pred}(b, u) = \arg\max_{k \in \{0,1\}} \text{conf}\{L^*_{pred}(b, u) = k\}$$

Thus, the graph is completed and each edge is labelled by a confidence value.

3.1 Extending a Partialy-known State with Predictions

Given a complete multigraph and confidences, we extend the partially-known state s' by adding literals to the state where confidence exceeds a predefined confidence threshold, that is,

$$l_p \in s' \leftrightarrow$$
$$(L_{pred}(b, u) = 1) \wedge conf\{L_{pred}(b, u) = 1\} > c_t,$$

where l_p is the positive literal of proposition p, formed from predicate $pred$ linking objects b and u, and c_t is the confidence threshold.

For edges predicted not to exist in a graph with confidence higher than a threshold value c_t, we extend the partially-known state s' in a similar fashion:

$$\neg l_p \in s' \leftrightarrow$$
$$(L_{pred}(b, u) = 0) \wedge conf\{L_{pred}(b, u) = 0\} > c_t,$$

For example, the partially-known graph in Figure 4 contains a dashed edge representing that the proposition `(can-pickup robot block01)` is unknown. Initially,

$$L_{can-pickup}(robot, block01) = ?$$

After prediction, $L_{can-push}(robot, block01) = 1$, with confidence higher than c_t. Therefore, the literal `(can-pickup robot block01)` is added to the current state.

4 Evaluation

The system integrates the prediction into a planning and execution framework, ROSPlan [Cashmore *et al.*, 2015]. The M^{3}VR method is integrated as a ROS service enabling its use as an automatic routine. Gaussian kernels, which appeared the best for this family of the problems, are used as the feature functions in M^{3}VR with equal parameters for all domains and experiments.

With this framework we were able to generate randomised problem instances from four domains: *tidy-room*, inspired by the problem of cleaning a child's room with an autonomous robot presented in Krivic et al. [2015], *course-advisor*, adapted from Guerin et al. [2012]; *mars-rovers*, a multi-robot version of the navigation problem of Cassandra et al. [1996], in which several robots are gathering samples; and *persistent-auv*, described by Palomeras et al. [2016]. The system together with all domains and test scripts is available and can be found at *github.com/Senka2112/IJCAI2017*. The system can be easily used with other domains as well.

4.1 Evaluating State Predictions

To evaluate prediction accuracies we generated examples of states in each domain varying the size of the problem and the percentage of the knowledge on the state. The problem size is varied by increasing the number of objects from 5 to 100 by an increment of 5. The knowledge is varied by generating complete states and removing literals at random. Percentages of knowledge used in tests are 0.5%, 1%, 2%, 3%, 5%, 8%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%. To examine the reproducibility of prediction problems we randomly generated 10 states for each combination of the percentage of knowledge and problem size utilizing 10-cross-fold-validation. Thus, for each domain were generated $20 \times 14 \times 10$ problems in total.

The prediction was applied to every problem and results were compared to the ground truth. To examine the lower limit on problem size, we extracted which amount of the known data that is needed to achieve accuracy higher than 90%. This is shown in the Figure 5. The number of learned relations is large for each domain: with 20% of the known predicates and with 20 objects, accuracy is higher than 90% for all domains except *mars-rovers*.

Accuracy increases with the size of the problems. With

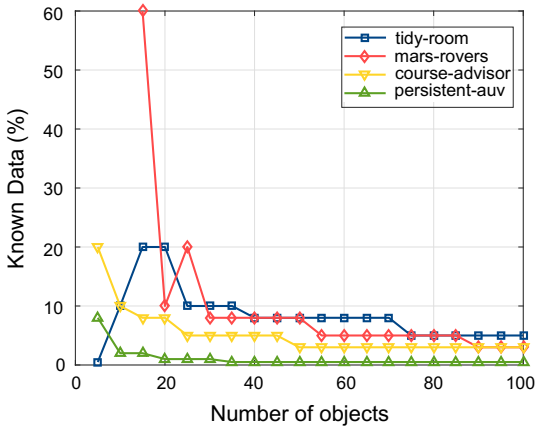

Figure 5: Minimal percentage of Known Data which gives stable accuracy equal to or higher than 90% for all four domains and different problem sizes. Each data point is averaged over 10 instances of a problem.

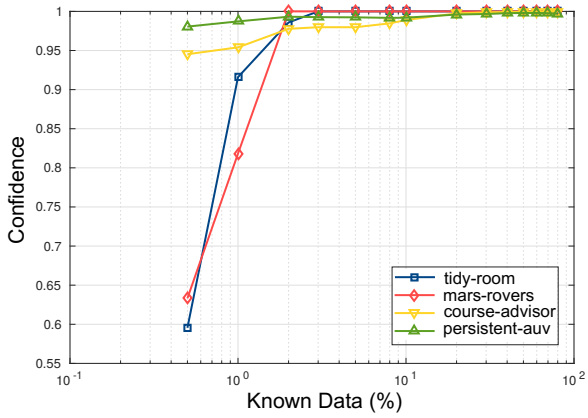

Figure 6: Mean confidence values from 10-cross-fold validation for 20 objects. The x axis representation is logarithmic.

40 objects in the each problem domain only 8% of known data is enough for accuracy over 90%. *Course-advisor* and *persistent-auv* domains contain more instances of objects and more predicates compared with the other two domains. This results in larger networks. Thus, for these domains, the accuracy is better for smaller numbers of objects as well. The high accuracy of predictions in these domains indicates domains with structure and order. Compared to other domains, the rovers domain appears to be the most unbalanced. With a small number of objects and initial knowledge, it is harder to capture the structure. Thus 60% initial knowledge is required to achieve 90% accuracy in the case of 15 objects.

The confidence for predictions is taken directly from the learning algorithm as described in the Section 3. It represents the measure of similarity between the learner output vector and the feature vector representing the existence of an edge. Figure 6 demonstrates the mean values of confidences for all four domains in case of 20 objects and varying percentage of initial data. Confidence saturates very quickly which corresponds the high accuracy results given in the Figure 5. This is the result of the structure which appears in the networks and which are captured by learners. Moreover, since the datasets were created by randomly removing knowledge, many vertices remained connected to the graph and their existing edges were exploited for predictions. The prediction confidence is expected to be very low for vertices for which no known edges exist within the rest of the graph.

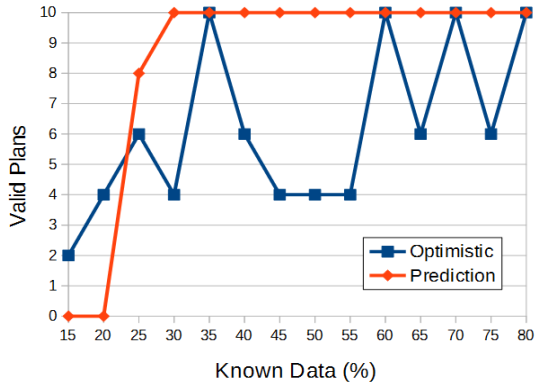

Figure 7: Number of problems for which valid plans were generated, for varying amounts of knowledge. Experiments were done for the *tidy-room* domain with 70 objects.

The high accuracy and confidence allows us to solve many otherwise unsolvable instances, while maintaining a high degree of robustness.

4.2 Evaluating Robustness

We investigate whether the high accuracy results in robust prediction. We make the problem deterministic by accepting all predictions with any positive confidence. The problem instances were made deterministic using prediction, and also by a baseline of assuming unknown propositions to be true. The problems were solved using POPF [Coles *et al.*, 2010]. The resulting plans were validated against the ground truth using VAL [Fox, 2004]. The results are shown in the Figure 7, showing that even for $> 30\%$ of Known Data, the prediction leads to robust plans.

Without prediction, none of our problems could be solved using a deterministic planner. Even with knowledge of 80% of the state, the goal was not in the reachable state-space. To provide a more illustrative analysis, we generate a relaxed plan-graph (RPG) and count the number of reachable actions before and after prediction.

Figure 8 shows the number of new reachable actions as a percentage of the total number of reachable actions in the ground truth. The increase in reachable actions is very large, especially with a smaller amount of Known Data. This increase in valid actions enabled by the prediction demonstrates an increase in size of reachable state space.

The prediction can be applied with a high confidence threshold, removing some uncertainty. Conditional planning can be used to deal with the remaining uncertainty by execut-

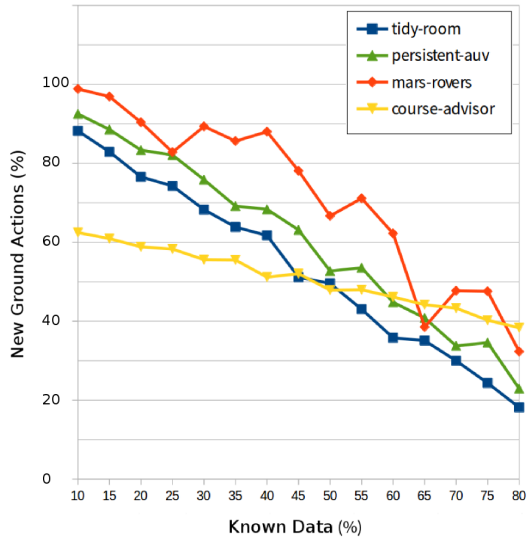

Figure 8: Number of newly reachable actions after prediction, as a percentage of actions in the ground truth. Tests were done for problems with 20 objects and varying amounts of knowledge in all four domains.

ing sensing actions to observe remaining unknown propositions.

4.3 Evaluating with Conditional Planning

Sensing actions were introduced into the *tidy-room* domain, allowing the agent(s) to determine the ground truth of unknown propositions. Problems were generated as described above, with 10% to 80% initial knowledge and 10-fold cross validation. All of the problems involve 20 objects. The number of goals was varied from 2 to 6. We used the planner CLG [Albore and Geffner, 2009] (offline mode) to solve problems in these extended domains, with a time limit of 1800 [s]. The prediction was applied before a single planning attempt. We recorded the time taken to solve and the duration of the execution trace of the contingent plan. The solution times are shown in Figure 9.

For problems with 5 goals and 6 goals, the problem could not be solved by CLG without predictions within the time limit. With prediction these problems were solved with an average of 89 [s]. These results illustrate the benefit of prediction in enhancing the scalability of contingent planning. The plan duration shows another benefit of state prediction in the quality of the plans produced: predictions can be used to avoid spending time on sensing actions when the confidence is high.

5 Conclusion

An agent can build hypotheses on unknown relations in the world model by exploiting similarities among the existing and possible relations. In this paper we have shown how uncertainty in planning can be decreased with predictions made by exploiting these similarities. We presented a system for such an approach where a state with uncertainty is represented as

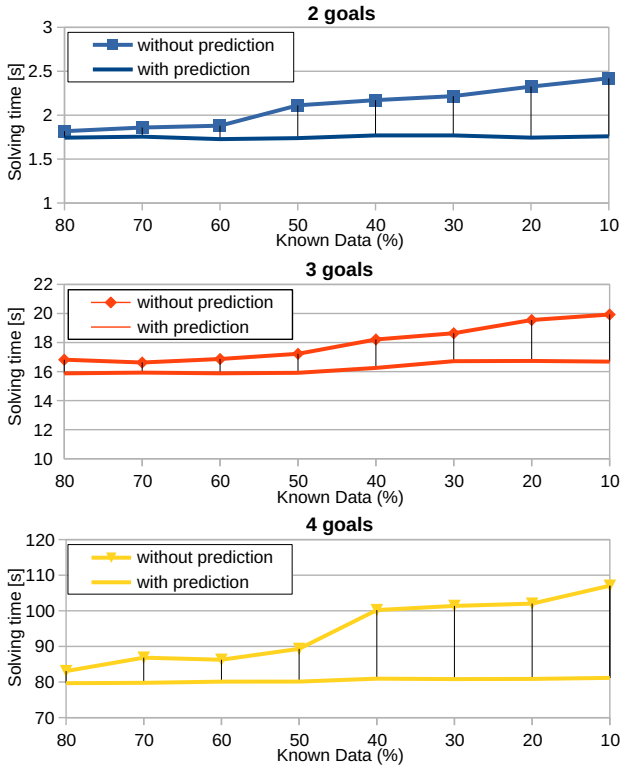

Figure 9: The time taken by CLG to solve problems with 2, 3, and 4 goals, with prediction (lower lines) and without prediction (upper lines).

a partially-known multigraph, we showed how M^{3}VR is used to predict edges in such a graph, and finally showed how these edges are reintroduced into the state as predicted propositions. This procedure is performed online.

We use the confidence values of predictions, filtering the number of unknown facts verifiable by sensing action. A lower confidence indicates a proposition that should be verified by a sensing action, while a high confidence prediction indicates a proposition that can be assumed true (or assumed false).

We have shown that with 20% knowledge of the state the accuracy of the state prediction is 90% with prediction tests in four different domains. We also demonstrated that the accuracy of the predictions leads to plans that are robust, and increase the scalability of our planners.

We noted that the high accuracy is in part due to the fact that many vertices remained connected to the graph after knowledge was randomly removed. This is not always the case in practice. In future work we intend to investigate how predictions can be used in order to inform a contingent planning approach. In particular, by directing the executive agent to perform sensing actions that connect disconnected nodes. These actions, while not supporting actions leading towards the goal, will allow for a higher confidence prediction of many other facts involving similar objects. This also might include new definitions of the prediction confidence measure.

References

[Albore and Geffner, 2009] H.; Albore, A.; Palacios and H. Geffner. A translation-based approach to contingent planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, 2009.

[Bonet and Geffner, 2000] Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems (AIPS'00)*, pages 52–61, 2000.

[Brafman and Shani, 2014] Ronen I. Brafman and Guy Shani. Replanning in domains with partial information and sensing actions. *CoRR*, 2014.

[Camacho *et al.*, 2016] Alberto Camacho, Christian Muise, and Sheila A. McIlraith. From fond to robust probabilistic planning: Computing compact policies that bypass avoidable deadends. In *The 26th International Conference on Automated Planning and Scheduling*, pages 65–69, 2016.

[Cashmore *et al.*, 2015] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. Rosplan: Planning in the robot operating system. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 2015.

[Cassandra *et al.*, 1996] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete bayesian models for mobile robot navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1996.

[Cesa-Bianchi *et al.*, 2013] Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. Random spanning trees and the prediction of weighted graphs. *Journal of Machine Learning Research*, 14:1251–1284, 2013.

[Coles *et al.*, 2010] Amanda Coles, Andrew Coles, Maria Fox, and Derek Long. Forward-chaining partial-order planning. In *Proceedings of the 20rd International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 42–49, 2010.

[Fox and Long, 2003] Maria Fox and Derek Long. PDDL2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Res. (JAIR)*, 20:61–124, 2003.

[Fox, 2004] R. Howey; D. Long; M. Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04)*, 2004.

[Gentile *et al.*, 2013] Claudio Gentile, Mark Herbster, and Stephen Pasteris. Online similarity prediction of networked data from known and unknown graphs. In *COLT 2013*, 2013.

[Ghazanfar *et al.*, 2012] Mustansar Ali Ghazanfar, Adam Prügel-Bennett, and Sandor Szedmak. Kernel-mapping recommender system algorithms. *Information Sciences*, 208:81–104, 2012.

[Guerin *et al.*, 2012] Joshua T Guerin, Josiah P Hanna, Libby Ferland, Nicholas Mattei, and Judy Goldsmith. The academic advising planning domain. In *Proceedings of the 3rd Workshop on the International Planning Competition at ICAPS*, pages 1–5, 2012.

[Hoffmann and Brafman, 2005] Jörg Hoffmann and Ronen I. Brafman. Contingent planning via heuristic forward search witn implicit belief states. In *Proceedings of the 15th International Conference on AutomatedPlanning and Scheduling (ICAPS'05)*, pages 71–80, 2005.

[Krivic *et al.*, 2015] Senka Krivic, Sandor Szedmak, Hanchen Xiong, and Justus Piater. Learning missing edges via kernels in partially-known graphs. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[Latouche and Rossi, 2015] Pierre Latouche and Fabrice Rossi. Graphs in machine learning: an introduction. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Proceedings of the 23-th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2015), pages 207–218, Bruges, Belgium, April 2015.

[Palacios and Geffner, 2006] Hector Palacios and Hector Geffner. Compiling uncertainty away: Solving conformant planning problems using a classical planner (sometimes). In *Proceedings of the 21st Conference on Artificial Intelligence (AAAI'06)*, 2006.

[Palomeras *et al.*, 2016] N. Palomeras, A. Carrera, N. Hurts, G. C. Karras, C. P. Bechlioulis, M. Cashmore, D. Magazzeni, D. Long, M. Fox, K. J. Kyriakopoulos, P. Kormushev, J. Salvi, and M. Carreras. Toward persistent autonomous intervention in a subsea panel. *Autonomous Robots*, 2016.

[Smith and Weld, 1998] David E. Smith and Daniel S. Weld. Conformant graphplan. In *Paper presented at the meeting of the AAAI/IAAI (AAAI'98)*, pages 889–896, 1998.

[Szedmak *et al.*, 2014] S. Szedmak, E. Ugur, and J. Piater. Knowledge propagation and relation learning for predicting action effects. In *Intelligent Robots and Systems (IROS'14)*, pages 623–629, 2014.