Probabilistic Replanning with Guarantees – Dissertation Abstract

Johannes Schmalz

School of Computing, Australian National University
johannes.schmalz@anu.edu.au
Supervisor: Felipe Trevizan

Abstract

State-of-the-art probabilistic replanners are solvers for probabilistic planning problems that offer a very efficient means to generate a suboptimal solution quickly, and in an anytime fashion improve on it. Unfortunately, current approaches do so at the cost of guarantees, i.e. the solution may not be optimal, and it can not guarantee its solution will lead to the goal with certainty. To address this issue we introduce CoGNeRe, a novel probabilistic replanner that uses techniques from operations research to provide guarantees and flexibility that previous replanners can not offer.

$$\begin{aligned}
\min_{\textbf{flow}} \quad & \sum_{X \in \mathcal{Q} \cup \mathcal{W}} \text{flow}_X \cdot \text{cost}(X) && \text{(LP 1)} \\
\text{s.t.} \quad & \text{flow}_X \geq 0 && \forall X \in \mathcal{Q} \cup \mathcal{W} \ \text{(C1)} \\
& \sum_{Q \in \mathcal{Q}} \text{flow}_Q = 1 && \text{(C2)} \\
& \text{regrouping constraints} && \text{(C3)}
\end{aligned}$$

Overview

Probabilistic replanners are a category of planners for Stochastic Shortest Path Problems (SSPs) (Geffner and Bonet 2013, 79, 81), which iteratively work on improving a candidate policy, and are able to return this candidate as a potentially partial policy when prompted for a solution in an anytime manner. Most replanners use the following method: relax the probabilistic effects of the problem (determinisation), solve the ensuing subproblem with strong deterministic solvers, append this plan as part of the candidate policy, and iterate these steps on undefined states to fill in the policy. FF-Replan (Yoon, Fern, and Givan 2007) and its extension Robust-FF (Teichteil-Königsbuch, Infantes, and Kuter 2008) have demonstrated the effectiveness of this approach winning the International Probabilistic Planning Competitions in 2004 and 2008 respectively. In terms of generating solutions quickly they are still considered state-of-the-art. However, these algorithms are not able to provide guarantees of solution quality, in particular, they can not guarantee that their policy has the highest possible probability of reaching a goal, and if they do manage to find such a policy, they can not guarantee the minimal expected cost.

In response, we present our column generation network flow replanner (CoGNeRe), a novel algorithm that combines this replanning approach with the column generation technique from operations research (Desrosiers and Lübbecke 2005). The aim is to exploit the speed of replanning and the mathematical guarantees of column generation. CoGNeRe can indeed provide guarantees of optimality, now we are filling in remaining theoretical gaps and refining the implementation to obtain an efficient planning algorithm. Preliminary results suggest that usually CoGNeRe is slower to output its first useful policy than Robust-FF, but is quickly able to overtake Robust-FF in terms of solution quality. We expect that CoGNeRe will generally converge to the optimal policy slower than state-of-the-art optimal planners like LRTDP (Bonet and Geffner 2003); CoGNeRe sacrifices fast convergence in favour of strong anytime performance.

To describe CoGNeRe, first consider the linear program (LP) that computes the optimal policy for an SSP by minimising a network flow problem (d'Epenoux 1963). A theorem from network flow (Ahuja, Magnanti, and Orlin 1993, 80–81) lets us decompose flow across a deterministic-planning-problem graph into a combination of paths and cycles; we generalised this to probabilistic-problem graphs. With this result we can find an SSP's cheapest flow and therefore its optimal policy as a combination of plans and cycles across the SSP's all-outcomes determinisation, that is, a class of planning problem that relaxes the SSP by mapping each probabilistic effect to a single deterministic action (Yoon, Fern, and Givan 2007). So, we can compute an SSP's optimal policy with LP 1. In LP 1 we consider the set of plans and cycles over the all-outcomes determinisation, $\mathcal{Q}$ and $\mathcal{W}$ respectively; and we introduce variables flow_X for each $X \in \mathcal{Q} \cup \mathcal{W}$ to denote the amount of flow being pumped through the plan or cycle X. Constraint C1 forces flow to be non-negative; the convexity constraint C2 forces a flow of 1 to pass from the initial state to goals, which can be interpreted as the requirement that the corresponding policy reaches goal with probability 1; and the regrouping constraints C3 force the flow to respect the probability distribution of action effects in the SSP.

In LP 1 there is a variable for each plan and cycle in the all-outcomes determinisation, which is intractable to enu-

merate explicitly for any interesting problem, so solving this LP directly is impractical. This is where column generation can help us: given the specification of an LP with intractably many variables, referred to as the master problem, column generation works with a smaller LP, the reduced master problem (RMP), which contains a subset of the master problem's variables. The algorithm solves the reduced master problem, and iteratively adds variables as needed to converge to a solution that is optimal for the master problem. The intuition for this approach is that not all variables — and in fact most variables — are not relevant to the optimal solution, so we focus on reduced master problems which are sufficiently small to solve, but still lead us to the optimal solution of the master problem. Column generation is able to determine which variables need to be added to the RMP to converge to the master problem's optimal solution by constructing a series of pricing problems, which are relaxations of the original problem. A pricing problem's solution yields the new variables we need to add, and an absence of a solution informs us that column generation has converged.

CoGNeRe applies column generation to the LP for finding the optimal policy as a combination of plans and cycles on the all-outcomes determinisation as follows:

1. we start with a small set of plans and cycles,
2. solve the reduced master problem,
3. construct and solve the corresponding pricing problem to find a plan or cycle whose addition to the RMP will improve the RMP's objective in the next iteration,
4. if we find such a plan or cycle: add it, and repeat from step 2; if not, we have an optimal solution and can terminate.

In CoGNeRe, the pricing problem turns out to be a deterministic shortest path problem; so, as with other replanners, we repeatedly solve deterministic planning problems to obtain a policy for the original problem.

Unfortunately, the devil is in the details: the pricing problem is given by the all-outcomes determinisation with potentially negative action costs determined by the column generation framework with respect to each state. A solution to the pricing problem is a negative cost plan, or negative cycle, or confirmation that neither exist. So, we are dealing with a deterministic shortest path problem with state-dependent costs, negative costs, and crucially, potentially with negative cycles.

Note that the dynamics of pricing problems remain identical across iterations except for action costs, which get updated at every step, i.e. the states, action, effects, etc. remain unchanged, only the costs of some actions are updated in a state-dependent manner.

Current Work

Dealing With Conditional Negative Costs And Negative Cycles

Recall that to solve the pricing problem we must return a negative plan or cycle if it exists with the updated costs, otherwise we terminate column generation. We were able to get relatively strong performance on small problems using a variant of Bellman-Ford with an early-stop mechanic, i.e. once a negative plan or cycle has been detected it returns it straight away. However, this approach and others described by a survey of similar problems (Cherkassky and Goldberg 1999) are uninformed and polynomial with respect to the state space, so they do not scale to larger problems. So, the challenge is to find an informed algorithm that can cope with negative, state-dependent costs, and can detect negative cycles. In fact A* with minor modifications can solve such problems, as long as the heuristic is admissible; so the issue now is that getting an informative and admissible heuristic with negative state-dependent costs is difficult, and even undefined in the presence of negative cycles.

Negative cycles are particularly difficult to deal with, since they can appear anywhere in the state space, without any indication of their presence from surrounding states or transitions — as long as the negative cycle is reachable, it is an optimal solution. For now we avoid the issue by focusing on acyclic problems. This is a limitation, but still leaves us with a large class of problems, notably any SSPs with the notion of monotonically increasing or decreasing timesteps or resources, e.g. finite-horizon SSPs and vehicle routing with fuel consumption.

Even in the absence of negative cycles, negative costs make it difficult to use state-of-the-art heuristics. As a case study, consider disjoint action landmarks (in the deterministic setting), i.e. a set of actions $\mathcal{L}$ such that any plan must use at least one action from $\mathcal{L}$. With non-negative cost actions we can give an admissible heuristic by $\min_{l \in \mathcal{L}} \text{cost}(l)$ since the cheapest possible way to pass through the landmark is by applying the cheapest action once. This argument falls apart with negative costs, since the cheapest way to pass through a landmark may collect multiple negative cost actions in the landmark. This issue makes it non-trivial to adapt landmark-based heuristics to pricing problems e.g. LM-cut (Helmert and Domshlak 2009).

Generating heuristics for state-dependent problems is an actively studied research question. One approach uses edge-valued multi-valued decision diagrams to compactly encode cost functions (Geißer, Keller, and Mattmüller 2015, 2016). Unfortunately this doesn't work for us, since we do not have neat algebraic expression for expressing costs, but rather different costs on a per-state basis, as determined by the pricing problem.

We are still exploring which heuristics are most suitable to be adapted to our use-case.

Exploiting Similarity Between Pricing Problems

Each pricing problem is identical in terms of dynamics, i.e. each pricing problem is similar to the original problem's all-outcomes determinisation, except action costs, which are determined in a state-dependent manner by the column generation algorithm. More formally, the transition systems induced by each pricing problem is identical, up to cost and labels of particular transitions. Our current approach for exploiting this is a variant of the Lifelong Planning A* algorithm (Koenig, Likhachev, and Furcy 2004). The idea is that we run A*, but upon returning a solution we do not discard the frontier and current best costs, but store it, and for the

next pricing problem we analyse what cost changes have been made:

- if an edge cost has been decreased then reinsert the affected vertex into the frontier with the new cost, to ensure that the change is propagated to the optimal path if relevant;
- if an edge cost has been increased, then all paths that relied on that edge must be re-evaluated, i.e. descendants of that edge have their current best cost reset to ∞, and they are re-inserted into the frontier.

In the worst case an edge close to the initial state has its cost increased, and we have to recompute the entire search graph, and thus we are running regular A* with the overhead of processing edge updates and the frontier. In practice however, we have found that this approach reduced computation time substantially.

Lower Bound For Policy Cost

The relationship between primal and dual LPs allows us to provide upper and lower bounds for the problem. A novel feature of our approach is that we can leverage the column generation framework to provide a lower bound for the optimal policy cost which becomes tighter as CoGNeRe progresses.

Consider the objective of the optimal solution for the master problem z^*_{MP}. A well-known theorem from column generation (Desrosiers and Lübbecke 2005, 8–9) gives us the bound $\bar{z} + \kappa\bar{c}^* \leq z^*_{\text{MP}} \leq \bar{z}$ where

- $\bar{z}$ is the objective value of the optimal solution for our current RMP;
- $\bar{c}^*$ is the smallest reduced cost for our current RMP, i.e. the cost of the most negative plan or cycle in our pricing problem;
- κ is an upper bound for the sum of all variable assignments in the master problem's optimal solution. In an acyclic problem, thanks to the constraint that ensures a flow of one through the network (convexity constraint C2), we can set $\kappa = 1$.

The value of $\bar{z}$ corresponds to the cost of the current policy, which clearly gives an upper bound to the cost of the optimal policy, hence $z^*_{\text{MP}} \leq \bar{z}$. In the lower bound for z^*_{MP}, $\bar{c}^*$ denotes the most we can possibly decrease the current solution's objective by adding the variable corresponding to the most negative plan or cycle. In the acyclic case we can assign the new variable a value of at most 1 (due to convexity constraint C2), and so the most we can reduce the objective is indeed $\bar{c}^*$. In the presence of cycles it gets a bit more complicated since a cycle's variable is not bounded by the convexity constraint. Note that $\bar{z}$ is monotonically decreasing since solutions can only improve with the introduction of new columns, so the upper bound is only getting tighter; the lower bound has no such guarantee and may fluctuate, but we can take the maximum across all RMPs and thereby get a monotonically increasing lower bound as well.

Lower bounds are not by themselves novel, since heuristics can provide lower bounds; but especially in a probabilistic setting these tend to be very loose, and the lower bounds from column generation can be tighter. This technique enables our anytime solver to give a more precise optimality gap.

Future Work

Here we outline some potential directions of future research that CoGNeRe and more generally, operations research in planning can take.

Dealing with Cyclic Problems

CoGNeRe is able to solve cyclic problems by running an algorithm that can detect negative cycles e.g. Bellman-Ford on the pricing problems, and then adding any found negative cycles as columns, just like with plans. As discussed before, the issue is that we need admissible heuristics to scale up to larger problems. First, we need to redefine what it means for a heuristic to be admissible in the presence of negative cycles, and there are two options: (1) an admissible heuristic must still underestimate the actual cost, so if a negative cycle is reachable from state s then $h(s) = -\infty$; (2) we only require the heuristic to reason about plans, so an admissible heuristic must underestimate the cost of the cheapest plan, and may ignore negative cycles.

Approach (1) has the potential of destroying a lot of information about negative plans, and in a sense prioritises negative cycles. Approach (2) suggests that cycle and plan search should be separated, e.g. we run a plan finding algorithm with some cycle detection mechanisms, as in (Cherkassky and Goldberg 1999), which can prioritise plans over negative cycles. In both cases the issue is that CoGNeRe is sensitive to the order in which columns are added so prioritising cycles or plans tends to perform well on some problems, but poorly on others. A strong solution needs to balance these issues, either by intelligently deciding whether a cycle or plan are more useful, or by adding both.

Solving Pricing Problems as Diverse Plans Problems

In column generation, for some problems, it is very efficient to extract multiple solutions from a single pricing problem, and add all of them as columns to the reduced master problem at once. The idea is that a column that is guaranteed to improve the RMPs solution at one step, is likely to be useful later as well; it is also a way to deal with the property of column generation that the column with most negative reduced cost may not correspond to the column that lets us converge most quickly. So by adding multiple columns we increase the chances of adding columns that lead to a solution quicker, and potentially allows us to reuse the results of the pricing problem in future iterations. Often this approach relies on the columns being sufficiently diverse. As motivation, in CoGNeRe, if we add multiple plans that share an action with an undesirable probabilistic effect, then that action's determinisation will receive a high cost in future iterations, which indicates to us that the columns are not useful.

These requirements can be expressed as a bounded quality diverse planning problem (Katz and Sohrabi 2020), where a solution is a set of plans, where the plans are sufficiently

diverse, and each plan's cost is bounded by some constant value. For CoGNeRe, the bound is 0 so that we only consider negative-cost plans in the pricing problem; diversity can be defined in terms of how many actions are shared. Katz and Sohrabi (2020) propose a flexible method for this style of problem which works by forbidding certain plans at the level of the planning task, which is not amenable to our method for exploiting similarity between pricing problems. So the challenge becomes how to combine these concepts.

Generalising CoGNeRe To More Problems

The LP approach endows CoGNeRe with a lot of flexibility with respect to objective functions and additional constraints. For instance, we can search for a policy that maximises the probability of reaching a goal; or, we can even stop CoGNeRe once it has reached some measurement of quality, e.g. return the partial policy as soon as the probability of reaching goal is > 0.9.

SSPs can be extended to constrained shortest path problems (CSSPs) which allows us to bound the expectation of different cost functions. More complex constraints are also possible, for instance using probabilistic linear temporal logic (PLTL) formulae. Such constraints can be compiled into constraints over expectations (Baumgartner, Thiébaux, and Trevizan 2018), i.e. of the form $\mathbb{E}[\text{cost}(\phi)] \geq \rho$ where ϕ is a linear temporal logic formula, $\rho \in [0, 1]$, and the cost function is very similar to the reward function for maximising probability: it is zero everywhere except for a subset of the goal states. It is difficult to generate informative heuristics for this kind of problem. The upshot is that there are currently no strong heuristics, and so informed CSSP planners suffer.

This motivates that CoGNeRe may be a strong candidate for this type of problem, since it does not require heuristics for probabilistic problems, only heuristics for the deterministic pricing problem.

Another complex variant of probabilistic problems requires the solver to take into consideration the variance of trajectories as defined by the candidate policy. This is difficult for current solvers because they are designed around the construction and improvement of policies, and information about the possible trajectories needs to be extracted afterwards. CoGNeRe on the other hand, natively reasons about all possible trajectories in a compact, finite way, so it should offer a clean solution.

Heuristics For Planning Under Uncertainty Models

Rather than directly solving extensions of SSPs and CSSPs like MDPIPs (Shirota Filho et al. 2007), PLTL-constrained CSSPs, etc. we can explore the idea of solving relaxations with CoGNeRe in order to obtain heuristics for the original problem. We have already discussed CoGNeRe's ability to generate lower bounds; this combined with different pricing problem heuristics and techniques for reusing partial solutions may set CoGNeRe up as a good method for obtaining heuristics. CoGNeRe with its guarantees of optimality is likely to be too slow for this, so we can explore how to generate potentially non-admissible, informative heuristics using relaxation techniques from operations research, based on approaches like Dantzig-Wolfe decomposition and Benders decomposition.

Acknowledgments

Thanks to J. Christopher Beck for his feedback.

References

Ahuja, R.; Magnanti, T.; and Orlin, J. 1993. *Network flows : Theory, Algorithms, and Applications*. Englewood Cliffs, N.J: Prentice Hall.

Baumgartner, P.; Thiébaux, S.; and Trevizan, F. 2018. Heuristic Search Planning With Multi-Objective Probabilistic LTL Constraints. In *Proc. of 16th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*.

Bonet, B.; and Geffner, H. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *Proc. of 13th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Cherkassky, B. V.; and Goldberg, A. V. 1999. Negative-cycle detection algorithms. *Mathematical Programming*.

d'Epenoux, F. 1963. A probabilistic production and inventory problem. *Management Science*.

Desrosiers, J.; and Lübbecke, M. E. 2005. *A Primer in Column Generation*, 1–32. Boston, MA: Springer US.

Geffner, H.; and Bonet, B. 2013. A Concise Introduction to Models and Methods for Automated Planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*.

Geißer, F.; Keller, T.; and Mattmüller, R. 2015. Delete Relaxations for Planning with State-Dependent Action Costs. *Proc. of the Int. Symposium on Combinatorial Search*.

Geißer, F.; Keller, T.; and Mattmüller, R. 2016. Abstractions for Planning with State-Dependent Action Costs. *Proc. of 26th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? *Proc. of 19th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.

Katz, M.; and Sohrabi, S. 2020. Reshaping Diverse Planning. *Proc. of the AAAI Conference on Artificial Intelligence*.

Koenig, S.; Likhachev, M.; and Furcy, D. 2004. Lifelong planning A*. *Artificial Intelligence*.

Shirota Filho, R.; Cozman, F. G.; Trevizan, F.; de Campos, C. P.; and Barros, L. N. 2007. Multilinear and Integer Programming for Markov Decision Processes with Imprecise Probabilities. In *Proc. of 5th Int. Symposium On Imprecise Probability: Theories And Applications*.

Teichteil-Königsbuch, F.; Infantes, G.; and Kuter, U. 2008. RFF: A robust, FF-based MDP planning algorithm for generating policies with low probability of failure. *Sixth International Planning Competition at ICAPS*.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In *Proc. of 17th Int. Conf. on Automated Planning and Scheduling (ICAPS)*.