FTSPlan: Task Reformulation via Merge-and-Shrink

Álvaro Torralba[1], **Silvan Sievers**[2], **Rasmus G. Tollund**[1], **Kristian Ø. Nielsen**[1]

[1] Aalborg University, Denmark
[2] University of Basel, Switzerland
alto@cs.aau.dk, silvan.sievers@unibas.ch, rasmusgtollund@gmail.com, kristianodum@gmail.com

Abstract

We present FTSPlan, an entry to the optimal and agile tracks of the International Planning Competition 2023. FTS plan represents the planning task as a set of (labelled) transition systems, allowing for powerful task reformulation methods using the merge-and-shrink framework.

Introduction

The core idea of FTSPlan plan is to apply the merge-and-shrink framework (Sievers and Helmert 2021) for task reformulation, representing the planning task as a set of (labelled) transition systems and applying a series of task transformations before starting the search (Torralba and Sievers 2019).

The merge-and-shrink framework, inspired by previous work in model-checking (Dräger, Finkbeiner, and Podelski 2006, 2009), represents the planning task as a set of labelled transition systems, called factored transition system (FTS). It iteratively applies transformations of that FTS, such as, e.g., the name giving merging and shrinking, until obtaining a single transition system which is an abstraction of the original planning task (Helmert et al. 2014; Sievers and Helmert 2021). In general, the merge-and-shrink framework offers a theory of transformations that can be applied to a planning task in this FTS representation and what desirable properties the resulting task has (Sievers and Helmert 2021). Within that view, merge-and-shrink heuristics use transformations that ensure the final result to induce an admissible heuristic, even if the abstract plan has no relation to any plan in the original task.

By restricting the transformations that can be used, and storing sufficient information on each step, FTSPlan can ensure that any plan of the transformed task can be mapped back into a plan for the original task (Torralba and Sievers 2019). Therefore, if by applying sub-sequent transformations the process concludes in a single transition system, a plan can be directly extracted. However, typically this requires too much time and memory to be computed, so the process is stopped earlier, obtaining a task representd as a set of transition systems. Still, the reformulated task may be easier to solve than the original task as each intermediate transformation can have potential benefits. For example, bisimulation shrinking (Nissim, Hoffmann, and Helmert 2011) can reduce the amount of states, e.g., by reducing equivalent/symmetric states to a single one. Label reduction (Sievers, Wehrle, and Helmert 2014) can represent the task more compactly using fewer labels. We can also leverage dominance analysis techniques (Torralba and Hoffmann 2015; Torralba 2017) to prune irrelevant and/or redundant transitions (Torralba and Kissmann 2015), such that a plan is still guaranteed to remain. Combined with pruning of unreachable and dead-end states (Sievers and Helmert 2021), this can greatly simplify the planning task in some domains.

FTSPlan

FTSPlan is implemented on top of the Fast Downward Planning System (Helmert 2006), using an adaptation of the existing implementation of the merge-and-shrink framework (Sievers 2018).

Task Representation

The input planning task is specified in PDDL (McDermott et al. 1998) and is first converted into finite-domain representation (FDR) using the translator component of Fast Downward (Helmert 2009). It is then simplified by removing actions with the h^2 preprocessor (Alcázar and Torralba 2015). Since FTSPlan (more precisely: the merge-and-shrink implementation it is built on) does not support conditional effects, we use a compilation that creates a copy of each action for each effect condition (Nebel 2000), thus obtaining a planning task in SAS$^+$ representation (Bäckström and Nebel 1995). Even though this compilation is worst-case exponential in the number of conditional effects, this typically scales well in domains with few conditional effects.

The first step of FTSPlan is to further translate the given SAS$^+$ task into the FTS task representation. An FTS planning task is represented as a factored transition systems, which is a set of transition sytems $\mathcal{T} = \{T_1, \dots, T_n\}$ with a common set of labels L. This is directly induced by the task in SAS$^+$ representation: each SAS$^+$ variable is represented by a transition system and each SAS$^+$ action is represented by a label. This is done in such a way that the synchronized product of all transition systems is the same as the state space of the orginal SAS$^+$ task.

From there on, FTSPlan works in two phases. First, it applies a task reformulation procedure to simplify the FTS task as much as possible. Secondly, it uses a search algorithm to find a plan to the reformulated task and reconstructs the plan for the original task.

Task Reformulation

In the first phase, FTSPlan applies the merge-and-shrink framework to reformulate the given FTS planning task. This means to iteratively applies merge, shrink, prune, and label reduction transformations until no more transformations can be applied due to an imposed limit on the representation size or a time limit is reached.

Merging The merge transformation replaces two transition systems $T_i, T_j \in \mathcal{T}$ by their synchronized product $T_i \otimes T_j$. This is an exact transformation, i.e., it preserves the all path costs in the transformed FTS task. Merging increases the size of the task representation because the size of the product is quadratic $|T_i \otimes T_j| = O(|T_i||T_j|)$. By itself, this may be a harmful reformulation in terms of search time, but this can enable further shrinking, label reduction, and pruning transformations leading to a net benefit. Therefore, we apply merge conservatively, only allowing to merge two transition systems if their product has fewer than 1000 states. To decide which two transition systems to merge, we use the score-based MIASM strategy (Sievers, Wehrle, and Helmert 2016).

Shrinking The shrink transformation replaces a transition system T_i by an abstraction thereof, where two or more states are reduced to a single one. This reduces both the size of the task representation, as well as the state space; so it is always beneficial. However, to ensure that the plan can be reconstructed, only certain shrink strategies can be applied.

The shrink strategy differs depending on whether we are dealing with optimal or satisficing/agile planning. For optimal planning, we use bisimulation shrinking without a limit on the number of states, which guarantees to preserve the cost of all plans (Nissim, Hoffmann, and Helmert 2011) and as such is an exact transformation. For agile planning, we use weak bisimulation shrinking (Hoffmann, Kissmann, and Torralba 2014), which achieves larger size reductions than full bisimulation while ensuring that a (non necessarily optimal) plan can be reconstructed.

Label Reduction Label reduction applies an abstraction on the label set of the factored transition system, thus replacing certain labels by a common (new) one (Sievers, Wehrle, and Helmert 2014). There are polynomial-time computable conditions under which this can be done in a plan-preserving way, i.e., as an exact transformation. The resulting task representation is slightly smaller, but the main advantage is that label reduction enables larger reductions with bisimulation shrinking.

Pruning Pruning removes states and/or transitions if they are unreachable from the initial state and/or dead-ends. This is an exact transformation as long as we are only interested in the reachable and relevant part of the state space.

Search Algorithms

After reformulating the FTS task in the first phase, in the second phase, FTSPlan could in principle use any algorithm to compute a plan for the transformed task, mapping back the plan to a plan for the original task using stored information about each transformation step applied during task reformulation. A practical difficulty, however, is that most planning algorithms are not defined for tasks in the FTS representation.

As the FTS representation is slightly more expressive than FDR (e.g., it can directly represent disjunctive preconditions and/or conditional effects that depend on only one factor), planning algorithms and heuristics need to be slightly adapted to use this representation. In our original work (Torralba and Sievers 2019), we already show how to perform explicit-state search using several heuristics: merge-and-shrink (Sievers and Helmert 2021), $h^{\max}$(Bonet and Geffner 2001), and FF (Hoffmann and Nebel 2001). Since then, we also extended the planner to support symbolic bidirectional search (Torralba et al. 2017).

We use different search algorithms for optimal and satisficing/agile planning.

A*with Merge-and-Shink heuristics For the optimal setting, we use A*with the merge-and-shink heuristic. As merge strategy, we use score-based MIASM (Sievers, Wehrle, and Helmert 2016). As shrink strategy, we use bisimulation shrinking (Nissim, Hoffmann, and Helmert 2011) with a limit of 50 000 abstract states. We further use exact label reduction (Sievers, Wehrle, and Helmert 2014) and pruning (Sievers and Helmert 2021) of all unreachable and irrelevant states.

Symbolic Bidirectional Search with BDDs We use the original implementation (Torralba and Alcázar 2013; Torralba et al. 2017), and adapt the creation of the BDDs representing the initial state, goal, and the transition relation to take as input the task in FTS representation. The BDD representation is akin to the one used for SAS$^+$ planning tasks, where each transition system $T_i \in \mathcal{T}$ corresponds to a finite-domain variable. Thus, each state $s_i \in T_i$ would correspond to a value of a variable and is represented within the BDDs as a binary string $\langle s_i \rangle$.

The initial state and goal are straightforward to construct, as each transition system can be interpreted as a variable with one initial state value and one or more goal values. For example, the goal BDD can be constructed as the following expression

$$\bigwedge_{T_i \in \mathcal{T}} \bigvee_{s_i \in T_i, s \models G} \langle s_i \rangle$$

For the transition relation, we construct a BDD per label l:

$$TR_l = \bigwedge_{T_i \in \mathcal{T}} \bigvee_{s_i \xrightarrow{l} s_i' \in T_i} \langle s_i \rangle \wedge \langle s_i' \rangle$$

Such BDDs are always of polynomial size in the size of the FTS representation, as long as BDD variables representing $\langle s_i \rangle$ and $\langle s_i' \rangle$ are contiguous in the variable ordering.

Once the BDDs for the initial state, goal, and transition relation BDDs are constructed, the rest of the algorithm and implementation can be directly reused (Torralba et al. 2017). An important difference, however, is that we do not have h^2 mutexes available, so we do not leverage them during the search as done by the original symbolic bidirectional search implementation (Torralba and Alcázar 2013; Torralba et al.

2017). The reason is that we apply reformulation methods to the task representation (e.g. by merging multiple variables into the same factor and/or shrinking values of one of the factors), so mutexes obtained by h^2 on the orignal task cannot be directly applied to search in the reformulated task.

Lazy Greedy Best-first Search with the FF heuristic For the satisficing/agile setting, we use greedy best-first search, which explores the state space by exploring first those states with lowest heuristic value (i.e., estimated as closest to the goal by the heuristic). The search uses lazy evaluation and preferred operators (Richter and Helmert 2009).

As heuristic, we use the popular FF heuristic (Hoffmann and Nebel 2001), which is based on computing a delete-relaxed solution. To compute the FF heuristic, we use the procedure detailed by Torralba and Sievers (2019), which can be interpreted as computing an hypergraph from the FTS representation (Steinmetz and Torralba 2019) to extract the relaxed plan in the same way as the implementation of the FF heuristic in Fast Downward (Helmert 2006).

While this is not a state-of-the-art configuration, the planner has not been extended to support other heuristics yet such as the landmark heuristic used by LAMA (Richter and Westphal 2010).

Conclusion

We present FTSPlan, a planner based on reformulating the planning task into FTS task representation. While the search algorithms currently implemented in the planner are slightly inferior to the state of the art, FTSPlan aims to overcome this limitation by simplifying the planning task before the search starts. Future work will consider implementing other modern heuristics and/or search strategies in the planner.

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS$^+$ Planning. *Computational Intelligence*, 11(4): 625–655.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1): 5–33.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2006. Directed Model Checking with Distance-Preserving Abstractions. In Valmari, A., ed., *Proceedings of the 13th International SPIN Workshop (SPIN 2006)*, volume 3925 of *Lecture Notes in Computer Science*, 19–34. Springer-Verlag.

Dräger, K.; Finkbeiner, B.; and Podelski, A. 2009. Directed model checking with distance-preserving abstractions. *International Journal on Software Tools for Technology Transfer*, 11(1): 27–37.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *Artificial Intelligence*, 173: 503–535.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the ACM*, 61(3): 16:1–63.

Hoffmann, J.; Kissmann, P.; and Torralba, Á. 2014. "Distance"? Who Cares? Tailoring Merge-and-Shrink Heuristics to Detect Unsolvability. In Schaub, T.; Friedrich, G.; and O'Sullivan, B., eds., *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*, 441–446. IOS Press.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14: 253–302.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL – The Planning Domain Definition Language – Version 1.2. Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, Yale University.

Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *Journal of Artificial Intelligence Research*, 12: 271–315.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing Perfect Heuristics in Polynomial Time: On Bisimulation and Merge-and-Shrink Abstraction in Optimal Planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1983–1990. AAAI Press.

Richter, S.; and Helmert, M. 2009. Preferred Operators and Deferred Evaluation in Satisficing Planning. In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, 273–280. AAAI Press.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research*, 39: 127–177.

Sievers, S. 2018. Merge-and-Shrink Heuristics for Classical Planning: Efficient Implementation and Partial Abstractions. In Bulitko, V.; and Storandt, S., eds., *Proceedings of the 11th Annual Symposium on Combinatorial Search (SoCS 2018)*, 90–98. AAAI Press.

Sievers, S.; and Helmert, M. 2021. Merge-and-Shrink: A Compositional Theory of Transformations of Factored Transition Systems. *Journal of Artificial Intelligence Research*, 71: 781–883.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In Brodley, C. E.; and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An Analysis of Merge Strategies for Merge-and-Shrink Heuristics. In

Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 294–298. AAAI Press.

Steinmetz, M.; and Torralba, Á. 2019. Bridging the Gap between Abstractions and Critical-Path Heuristics via Hypergraphs. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 473–481. AAAI Press.

Torralba, Á. 2017. From Qualitative to Quantitative Dominance Pruning for Optimal Planning. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 4426–4432. IJCAI.

Torralba, Á.; and Alcázar, V. 2013. Constrained Symbolic Search: On Mutexes, BDD Minimization and More. In Helmert, M.; and Röger, G., eds., *Proceedings of the Sixth Annual Symposium on Combinatorial Search (SoCS 2013)*, 175–183. AAAI Press.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient Symbolic Search for Cost-optimal Planning. *Artificial Intelligence*, 242: 52–79.

Torralba, Á.; and Hoffmann, J. 2015. Simulation-Based Admissible Dominance Pruning. In Yang, Q.; and Wooldridge, M., eds., *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 1689–1695. AAAI Press.

Torralba, Á.; and Kissmann, P. 2015. Focusing on What Really Matters: Irrelevance Pruning in Merge-and-Shrink. In Lelis, L.; and Stern, R., eds., *Proceedings of the Eighth Annual Symposium on Combinatorial Search (SoCS 2015)*, 122–130. AAAI Press.

Torralba, Á.; and Sievers, S. 2019. Merge-and-Shrink Task Reformulation for Classical Planning. In Kraus, S., ed., *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 5644–5652. IJCAI.