On Improving Plan Quality via Local Enhancements

Tomáš Balyo[1], Roman Barták[1], Pavel Surynek[1,2]

[1]Charles University in Prague, Faculty of Mathematics and Physics, Malostranské nám. 25, Praha, Czech Republic

[2]Kobe University, 5-1-1 Fukae-minamimachi, Higashinada-ku, Kobe 658-0022, Japan

{tomas.balyo, roman.bartak, pavel.surynek}@mff.cuni.cz

Abstract

There exist planning algorithms that can quickly find sub-optimal plans even for large problems and planning algorithms finding optimal plans but only for smaller problems. We attempt to integrate both approaches. We present an anytime technique for improving plan quality (decreasing the plan makespan) via substituting parts of the plan by better sub-plans. The technique guarantees optimality though it is primarily intended to quickly improve plan quality. We experimentally compare various approaches to local improvements.

Introduction

AI planning deals with the problem of finding a sequence of actions that transfers the world from some initial state to a state satisfying certain goal conditions. In this paper we attempt to improve the quality of plans generated by sub-optimal planners via doing local optimizations of the plans. More precisely, we will be improving makespan of parallel plans by optimizing sub-plans using SAT-based techniques such as SASE (Huang et al., 2010) that are successful for finding makespan-optimal plans.

The idea of making local repairs in a sub-optimal plan to improve it towards the optimal makespan already appeared in domain-dependent planning. Surynek (2011) and Wang et al., (2011) proposed techniques for shortening solutions of *cooperative-path finding* (CPF). Applying SAT-based mechanism to improve solutions of CPF problems where sub-solutions are replaced by makespan-optimal ones has been proposed in (Surynek, 2012).

Methodology

Assume that we have a sub-optimal parallel plan and we want to shorten its makespan. We propose a method that selects a sub-plan of the plan, finds a shorter sub-plan, and substitutes the original sub-plan by this shorter sub-plan in the original plan (Figure 1). To formulate this method precisely we must answer three questions: How is the sub-plan selected? How is a better sub-plan found? How many times should we repeat this local improvement process? Our idea is based on using existing planning techniques, namely the SAT-based approach, to find a better plan. Hence, the second question consists of two additional questions: How does a sub-plan define a planning problem? How do we solve optimally that planning problem? We shall now answer all above questions.

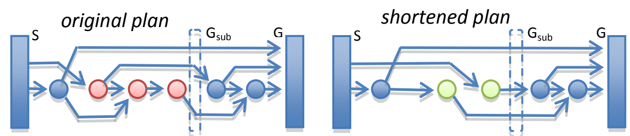

Figure 1: Substituting a sub-plan by a shorter sub-plan (arrows indicate causal relations between the action).

Identifying and Optimizing Local Sub-Problems

Let $P_1,\ldots, P_n$ be a plan reaching goal G from state S and $P_i,\ldots, P_j$ be its sub-plan. We formulate the planning problem P_{sub} whose initial state is the state after applying actions $P_1,\ldots, P_{i-1}$ to state S, i.e., $\gamma(S, (P_1,\ldots, P_{i-1}))$, where γ is a state transition function. The goal of P_{sub} contains the requirements of the sequence of actions $P_{j+1},\ldots, P_n$ and the goal conditions in G, which are not fulfilled by $P_{j+1},\ldots, P_n$., i.e., $\gamma^{-1}(G, (P_n,\ldots, P_{j+1}))$, where γ^{-1} is a regression function. This ensures that any solution of P_{sub} can substitute $P_i,\ldots, P_j$ in the original plan.

For the planning problem P_{sub} and a parameter k we create a SAT formula F_k which is satisfiable if and only if there is a parallel plan for P_{sub} of size k or shorter. To obtain this SAT formula we use the SASE encoding (Huang et al., 2010). If the formula F_k is satisfiable, then we can efficiently extract a parallel plan of size k (or shorter) from its satisfying assignment. We generate and solve F_k for decreasing k, starting with (j–i), until we find the smallest k such that F_k is satisfiable.

Plan Window Shifting

In this section we describe some methods how to select the local sub-plans (plan windows) for improvement.

The simplest idea is selecting the windows randomly (we used windows of maximal size 20). Another approach is to systematically shift a window of a certain size through the plan. A *Systematical window shifting (SWS) procedure* has three parameters: *(window size*, *window shift*, and *fixed*

This research is supported by the Czech Science Foundation (contract no. P103/10/1287) and by the Grant Agency of Charles University (contracts no. 266111 and 600112).

Table 1. Method description and experimental results

method	*size change*	*shift*	*fp*	*total makespan*		*joint makespan*		*joint runtime*	
LPG	N/A	N/A	N/A	14531	100%	6752	548%	6 604	28%
expo-fullstep	size*3/2	size	no	4411	30%	1911	155%	63 195	272%
expo-fullstep-fp	size*3/2	size	yes	3434	24%	1665	135%	54 568	235%
expo-halfstep	size*3/2	size/2	no	3577	25%	1660	135%	55 652	240%
expo-halfstep-fp	size*3/2	size/2	yes	3138	22%	1550	126%	52 018	224%
turbo-fullstep	size+1	size	no	3426	24%	1589	129%	52 446	226%
turbo-fullstep-fp	size+1	size	yes	3156	22%	1563	127%	53 834	232%
turbo-halfstep	size+1	size/2	no	3076	21%	1515	123%	50 251	217%
turbo-halfstep-fp	size+1	size/2	yes	3013	21%	1540	125%	52 568	226%
random	random size ≤ 20	random	yes	6351	44%	1935	157%	63 082	272%
SASE	N/A	N/A	N/A	1506	N/A	1232	100%	23 220	100%

point). It works by moving a window of the specified size through a plan from its beginning increasing its starting position by the window shift parameter until the end of the plan is reached. The fixed-point parameter of the SWS procedure specifies whether the iteration is repeated if any window has been improved. We start with windows of size 2 and increase the size either by 1 or exponentially by the factor 3/2. Table 1 gives an overview of the parameters of methods that we studied.

Experimental Study

To evaluate properties of the proposed methods we did an experimental study comparing various combinations of the methods (see Table 1). We used the LPG planner (Gerevini and Serina, 2002) to generate the initial plans. Because we are improving the makespan, we used the SASE planner (Huang et al., 2010) to compare the quality of plans generated by our method. We used eight classical STRIPS domains from the International Planning Competition (Koenig, 2012) with 232 total problems and allocated 30 minutes (1800 seconds) to each method per problem (run on Intel Core i7 920@2.67GHz with 6 GB RAM).

LPG solved 189 problems while SASE solved only 151 problems. Both planers solved 135 common problems Table 1 shows the detailed results, namely the total makespan for all solved problems, and the makespan and runtime for jointly solved problems. We can see that SASE can generally solve problems with short plans only (total makespan). The plans generated by LPG have more than five times larger makespan than the plans generated by SASE (joint makespan). All our methods significantly reduce the makespan of plans by LPG, which was our goal. In fact, the plans improved by our methods are very close to the optimal plans produced by SASE. The table also shows that it is worth to scan the windows systematically over the plan rather than trying them completely randomly. It also seems that keeping the plan windows smaller is beneficial. The turbo method is better both in runtime and makespan than the expo method due to conservative increase of window size. Also using the fixed point improves the expo method because it forces it to iterate longer over the smaller windows. The fixed point does not work well only for the method turbo-halfstep. The reason is that it forces re-optimizing the plan windows that are already optimal, which only adds overhead. In fact, we have found that all the proposed methods suffer from the problem of re-optimizing already optimal plan windows.

Conclusions

In this paper we proposed a method for improving quality of plans by doing local enhancements of sub-optimal plans. The method significantly reduced the makespan of plans produced by the LPG planner and made them comparable to the optimal plans. Though the method is still slower than SASE, it can find solutions for more problems thanks to exploiting the LPG planner (any sub-optimal planner can be used to find the initial plan). It is especially beneficial for problems with large plans where SASE fails to find any plan. The general conclusion from the experimental study is that it is worth optimizing a larger number of smaller sub-plans than trying a smaller number of larger sub-plans.

References

Gerevini, A., Serina, I. 2002. *LPG: a Planner based on Local Search for Planning Graphs.* Proceedings of AIPS-2002, pp. 13-22, AAAI Press.

Huang, R., Chen, Y., Zhang, W. 2010. *A Novel Transition Based Encoding Scheme for Planning as Satisfiability*. Proceedings of AAAI 2010, pp. 89-94, AAAI Press.

Koenig, S. 2012 (editor). International Planning Competition (IPC), http://ipc.icaps-conference.org/, University of Southern California, [accessed on April, 2012].

Surynek, P. 2011. *Redundancy Elimination in Highly Parallel Solutions of Motion Coordination Problems*, Proceedings of ICTAI 2011, pp. 701-708, IEEE Press.

Surynek, P. 2012. *A SAT-Based Approach to Cooperative Path-Finding Using All-Different Constraints*, Proceedings of SoCS 2012, to appear.

Wang, K. C., Botea, A., Kilby, P. 2011. *Solution Quality Improvements for Massively Multi-Agent Pathfinding*. Proceedings of AAAI 2011, AAAI Press.