Domain knowledge approximation for planning and scheduling

Grzegorz Góra

Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
ggora@mimuw.edu.pl

Abstract. We discuss some ideas aiming to improve constructing of complex decisions in solving of planning and scheduling problems. We argue that solving real-world planning/scheduling problems requires the presence of two important factors. First is domain knowledge, which should be transformed from the top level of the human understandable form to the low level understandable for the system. In this process concepts expressed by a human should be appropriately approximated. It is concerned with the hierarchical task networks (HTN), where the mid/high level concepts are crisp. We think it should be extended to deal with fuzzy and rough approximations of concepts. The second thing is concerned with the former one and is human assistance to help the system to build its control knowledge (aiming at finding the high-quality solution faster). Such knowledge is of course enclosed in the HTN, but also additional hints leading to the faster problem solving could be given for the system. We argue that not only knowledge expressed in some control rules should be given by the human, but also explanation should be given. We do not expect that human gives complete and fully understandable knowledge for system. But again approximation of such rules should be made with the use of rough and fuzzy concepts.
We present also the main steps of the process common for many case-based planners and we propose that the expert explanations can be crucial for improving their efficiency.

1 Introduction

From many years planning and scheduling are the key areas in AI. This is due to the fact that planning is one of the most important aspects of intelligent behaviour. It plays a significant role in many complex applications like controlling space vehicles and robot, oil-spill crisis response and many others. The significant challenge was to use AI techniques in order to automate the planning. A big progress was made in this research. A number of systems were created (see e.g. [9]) and many of them were applied for real-world problems. In recent years there was a big focus on the systems for the standard set of benchmark planning problems (e.g. blocks world, simplified logistics) used in the competitions of the Artificial Intelligence Planning and Scheduling (AIPS) Conference. Also more close to real-world applications problems were in interest (e.g. Airport, Satellite) in recent International Planning Competitions (IPC). Although the advance of the research is visible, still even for these simplified problems the lack of the scalability to the real-world sizes are its major disadvantages. Also in many real-world applications there is still great demand to improve both quality and efficiency of the planning systems.

The important factor allowing to achieve good planning systems is the domain knowledge. Naturally to solve the problem all the systems need a knowledge describing the problem: knowledge about the domain (such as the possible actions that may be taken and resources that may be consumed) and knowledge about good plans (e.g. the goals to be achieved, a measure that should be optimised, constraints that should be never violated). We are focusing on the additional kind of knowledge. This is the knowledge being an advice on how to find a good solution quickly (or even to find a solution at all), how to interact with the user, about user's preferences and knowledge about plan repair during execution. The big effort has been made in case of the additional knowledge advising on how to find good solution. Among them are search control knowledge (e.g. TALplanner, see [12]) and hierarchical task network (HTN) planning (well known intensive applications of HTN are SIPE-2 and O-PLAN, see [9]). It was demonstrated that thanks to an additional knowledge impressive improvement in efficiency can be obtained when compared to the corresponding up to date state-of-the-art planners not using additional knowledge (see e.g. [12], [14], [13]). Obviously the performance of the planners depends on the quality of the additional knowledge supported (e.g. the quality of control rules, the quality of splitting the task into the hierarchy of task network).

There are two major sources of achieving domain knowledge: use of machine learning techniques and advice of human expert. We briefly present them in the following paragraphs.

In recent years there were many attempts to achieve additional domain knowledge automatically (see, e.g. [1], [2]). They apply machine learning techniques to automated planning. Beyond speeding up the planning there are also two other goals for using learning in the planning area: improvement of plan quality and learn or improve domain theory. A big disadvantage of this approach is that it is very hard task (if possible) to obtain perfect domain knowledge by machine learning alone. Moreover in many cases the number of control rules that must be learned is very large (e.g. many thousands of rules). On the one hand system learns more and more control rules to improve its search. But on the other hand it must spend more and more time to match this set of rules against the current search state.

Another way of obtaining domain knowledge is the assistance of human expert. He or she can give an advice to the system on how to solve the problem. There are systems for which human can manually define control rules expressed in propositional logic to be used during planning (see e.g. [12]). There are two major disadvantages with such an approach. First, manually defining from scratch the appropriate domain knowledge (e.g. programming all useful specific control rules) for the given planning problem is a hard knowledge engineering task. Second, in many real-world problems human can easily formulate some control rules expressed in his own language, but it would be very hard to express it in propositional logic, even if, the obtained formula can be very complex and possibly erroneous.

We believe that combining the above approaches is crucial in obtaining useful domain knowledge. That is as well machine learning techniques and human expert assistance should be used. There were some approaches in this direction. In [14] authors propose to generate initially domain knowledge by a learning techniques and then to refine them by a human. Such an approach seems to be promising. Nevertheless we emphasise that more should be done in computer-human interaction for planning.

We present our ideas on the base of the widely used machine learning technique, case based reasoning, aiming to reuse parts of old plans to improve efficiency of planning (see, e.g. [6]). This is done by avoiding repetition of the planning effort, which was done for the old plans. In this case we achieve implicit domain knowledge. But we are also interested in explicit domain knowledge which can be extracted from the past cases (represented e.g. in the form of the control rules).

2 The planning problem

Instance of the propositional planning is defined as a tuple $\Pi = \langle L, O, I, G \rangle$. L is a first-order predicate language for describing the domain. O is a finite set of operators. I is the initial state and G the goal statement, both given as conjunctions of literals. A state is represented as a conjunction of positive literals. A *literal* is an instantiated predicate, that is, literal=(predicate argument-value*). For example, $(inroom\ keyA\ room1)$ is a literal in which $inroom$ is the predicate and $keyA$ and $room1$ are its instantiated arguments. A goal is a partially specified state, represented as a conjunction of positive ground literals, such as $(inroom\ keyA\ room1) \wedge (inroom\ boxA\ room1)$. A propositional state s satisfies a goal G if s contains all the atoms in G. An action is specified in terms of the preconditions that must hold before it can be executed and the effects that follow when it is executed. The precondition (effect) is a conjunction of function-free positive literals stating what must be true in a state before the action can be executed (stating how the state changes when the action is executed). We say that an action is applicable in any state that satisfies the precondition. Starting in state s, the result of executing an applicable action a is a state s' that is the same as s except that any positive literal P in the effect of a is added to s' and any negative literal $\neg P$ is removed from s'. A plan is a finite sequence $\langle o_1, ..., o_n \rangle$ of plan steps $o_i \in O$. A plan Δ solves an instance Π of the planning problem if and only if the result of the application of Δ to I leads to a state s that satisfies the goal specification G and operators are applicable in corresponding states.

3 Domain knowledge for planning

The important factor in planning systems that can significantly help to solve real-world problems is additional domain knowledge (see e.g. [9]). There are two major sources from which it can be obtained: from human expert and from the use of machine learning techniques (it can be deduced from the given domain knowledge describing the problem or/and induced from the past planning performances, both solutions and failures). We are convinced that planning for real-world problems could be hardly solved with the only use of the learning techniques (see e.g. [9]). They are naturally very helpful, but starting from some level of problem size they are intractable. Adding domain knowledge by a human expert is critical to obtain fully-scalable planners.

3.1 Acquisition of domain knowledge by a human

The main approaches for supporting planning systems with domain knowledge by a human are: specifying control rules by a human (see e.g. [12]) and HTN decomposition. Let us note that almost all planners for large-scale applications are HTN planners, because HTN planning allows the human expert to provide the crucial knowledge about how to perform complex tasks efficiently. Very well known systems based on this philosophy are SIPE-2 (see e.g. [9]) and O-PLAN (see e.g. [10]).

In HTN planning, the initial plan, which describes the problem, is viewed as a very high-level description of what is to be done - for example, building a house. Plans are refined by applying action decompositions. Each action decomposition transforms an action from some level to a partially ordered set of lower-level actions. The process continues until only primitive, low-level actions remain in the plan. In this sense action decomposition represents a kind of knowledge how to implement actions.

3.2 Acquisition of domain knowledge by machine learning

Let us now look for planning task as for constructing a complex decision. We assume that a planner performed a number of planning trials. Then each past case can be described by the corresponding initial state and the goal statement, I^P and G^P, respectively, i.e., $P = (I^P, G^P)$. For each past case a solution $D(P)$ is given, which is a kind of complex decision.

Because we want to induce some higher-level concepts or not all of the information included in P is relevant for the solution thus we use an information vector of attributes $inf(P)$ describing P, i.e., $inf(P) = (a_1(P), a_2(P), ..., a_n(P))$. In the simplest case one group of attributes would just describe an initial state, and the latter a goal state. There would also be a set of special attributes describing the solution $inf(D(P)) = (d_1(D(P)), d_2(D(P)), ..., d_m(D(P)))$ called decision attributes.

Our goal is to find a function D, which for a given problem P finds a solution $D(P)$, i.e., complex decision, in an efficient way, that is, it can be evaluated in polynomial time in the size of its input problem. In case of planning it is a very complex task in general. Thus it is well to reuse old cases and construct decision function D as a function $f(P', S_1, S_2, ..., S_k)$, where k is the number of past cases and $S_i = (inf(P_i), inf(D(P_i)))$ is a pair of *i-th* problem and its solution. In other words we want to induce function D basing on the old cases.

Generally there are two approaches for applying learning in planning. These are induction and deduction. Among deduction there is widely used explanation-based learning (EBL) (see, e.g., [5]), which requires domain theory what can help learning useful search control rules. Among induction methods there is case-based reasoning, which is a kind of analogical reasoning.

4 Domain knowledge approximation by human-computer collaboration

We believe that both expert knowledge and machine learning techniques are important for efficient planning. The commonly used learning techniques for planning base on the low-level, primitive operators. On the other hand expert knowledge is usually expressed in his or her own language. The gap between these two languages is one of the key problems in AI. We propose two things to deal with it.

First, the advice from an expert should be transformed from the top-level of the human understandable form to the low-level understandable for the system. In this process concepts expressed by a human should be appropriately approximated. Such a process is in a sense realised by hierarchical task networks (HTN). In this approach the problem and subproblem concepts are crisp. The important factor by which it should be extended is the possibility to express problem and subproblem concepts as fuzzy and rough approximations of human concepts. In many situations human could lead the planner to solve the problem, but he would articulate the hints in his or her own language. It means that the concepts he could use would be fuzzy. Moreover we could only approximate the concepts he uses. Those concepts in general are not known to the planner. So they should be learned from the examples and from the explanation of the user.

The second thing is concerned with the human assistance to help the system to build its control knowledge (aiming at finding the high-quality solution faster). We think that very important is the use of the explanation of the human leading the planner to the solution. Although there are many human-assisted planning approaches (see e.g. [9]) they lack the information from the user why he or she has chosen to do some action or prevented of doing others. The explanation again could be expressed in his own language. So it could contain some concepts well understandable for humans, but computer could only approximate them. For example: "Do not overtake if the situation on the road nearby is not safe". It means that the planner should apply action "overtake" while the precondition "situation on the road nearby is not safe" is true. Naturally this precondition is the concept which requires approximation. Let us take another example. During scheduling, time-tabling problem one of the conditions which should be fulfilled could be expressed as follow: ”Tasks in the specified problem should be honestly divided among the currently available persons”. It could be hard for the user to express what ”honestly” means in terms of low-level specification scheduling language. But he or she could gradually during scheduling execution inform the system that some of the conditions should be also fulfilled or not in order to preserve the honesty between persons. We would like to trace the reasoning process of the human by building rough approximation of it. We do not expect that human gives complete and fully understandable knowledge for system from a scratch. But again approximation of such knowledge should be made with the use of rough and fuzzy concepts.

5 Concept approximations in Case-Based Reasoning for planning

In this section we present the ideas presented in the previous section for the Case-Based Reasoning scheme.

The idea of case based planning is to choose some nearest neighbours $S_1, S_2, \ldots, S_k$ of the given problem P' from the set of all past cases and then on the base of corresponding solutions refine these solutions to obtain the solution for P'.

We want to emphasise the main scheme of case base reasoning for planning, which is used in many case based planning systems. First there is used a kind of similarity metric (or relation). For the new case we choose the closest problems according to this similarity measure. In order to appropriately reuse old cases we need to know what are the differences between new problem and the old ones. We want also to induce the relation between differences of problems and differences of solutions. Then we want to inference about the changes of solutions that are based on the changes of problems. Here arise three problems:

1. How to measure the similarity between problems (by means of relevant similarity measures)?
2. How to measure the differences between problems and solutions (by means of relevant discernibility measures)?
3. How to derive new solutions on the basis of the old ones (by reusing of old plans)?

In the following subsections we discuss in more detail these problems.

5.1 Similarity measure

Given two planning problems P_1 and P_2 we are interested to measure how close these problems are to each other. We are also interested to measure how close are the solutions $D(P_1)$ and $D(P_2)$. It is important to find similarity measures that have such a property that close problems have close solutions. By close solutions we mean that they can be easily converted into each other. It can be formalised as follows:

Let $\rho^P(P_1, P_2)$ be a metric measuring closeness of two planning problems P_1 and P_2. Let $\rho^D(D(P_1), D(P_2))$ be a similarity metric between two solutions of the given problems. We are looking for such metrics ρ^P and ρ^G that:

1. if $\rho^P(P_1, P_2)$ is small then $\rho^D(D(P_1), D(P_2))$ is also small
2. if $\rho^D(D(P_1), D(P_2))$ is small then it is easy to reconstruct solution $D(P_1)$ in order to obtain solution $D(P_2)$ and vice versa

There were many attempts to construct such metrics. The up to date trials show that it is hard problem. We believe that there are two factors which could help us to better approximate this similarity measure: (i) the use of relevant high-level concepts used to define the similarity measure and (ii) the expert advise in constructing such complex concept approximations and similarity measures. For example, expert can formulate such similarity measure in terms of natural language. The crucial could be then methods for inducing approximations of concepts described by an expert.

5.2 Discernibility measure

We want to describe differences between problems among similar cases in order to recognise what parts of the solution should be reconstructed to obtain the other ones. Here, we also would like to use an expert knowledge to build such a measure making it possible to extract differences between similar objects.

5.3 Reuse of the old plans

We want to deduce about changes in solutions in terms of the changes in problems. It could be formalised as follows. Let P_1, P_2 be two planning problem and $D(P_1)$, $D(P_2)$ their corresponding solutions. Let us assume that $\rho^P(P_1, P_2)$ is small. Then from the definition $\rho^D(D(P_1), D(P_2))$ is also small. We want to induce in this step what reconstruction should be made to obtain $D(P_2)$ from the $D(P_1)$ and vice versa. Having this information we finally want to induce the construction of complex decision $D(P')$ on the base of all plans $P_1, ..., P_k$ such that $\rho^P(P', P_i)$ is small for $i = 1, ..., k$.

Here, also the help of an expert seems to be crucial. He could give us some hints about reconstruction process for a number of examples. Then we would like to use these hints in inducing of function that reconstructs the old plans for problems closest to the new problem.

5.4 Higher-level concepts

In order to construct appropriate similarity (and discernibility) measures relevant higher-level concepts should be induced for expressing similarity (dissimilarity). There were some trials made in this direction, for example in PRODIGY system (see, e.g. [6]). These concepts are described in the expert language, i.e., natural language. Of course such concepts could be hardly found in general. But approximation of them would be useful. Such approximations can be induced using layered (hierarchical) learning. In such learning we use expert explanation for particular cases. Any such explanation can be treated as a reasoning scheme (e.g., in natural language) justifying to satisfactory degree similarity (or dissimilarity) of cases and is used as a hint in searching for patterns, called approximate reasoning schemes (AR schemes), relevant for approximations of concepts in hierarchical learning [7, 8]. AR schemes are approximations of reasoning made by expert. In our project we plan computer experiments for planning problems to verify this point of view checked already in case of inducing complex concepts related to road simulator [8].

6 Conclusions

The paper presents the general idea of the use of domain knowledge for planning. We argued that a bridge between human expressed knowledge and system low-level knowledge should be build. The extension of HTN planning to the case of tasks expressed as fuzzy and rough approximations of concepts could be helpful. Domain knowledge can be collected with the use of machine learning for planning and dialog with the user repeated cycle. This dialog should be also based on the translation of the human concepts into the hierarchy of simpler concept rough approximations. The approximation of the concepts should be made on the base of the dialog with a human. The ideas presented here need now to be tested in some planning/scheduling problems.

References

1. Zimmerman T., Kambhampati S.: Learning-Assisted Automated Planning: Looking Back, Taking Stock, Going Forward. AI Magazine 24(2): 73-96, 2003.
2. Minton S., ed.: Machine learning Methods for Planning. San Francisco, Calif.: Morgan Kaufmann, 1993.
3. Russell S., Norvig P.: Artificial Intelligence. A Modern Approach. Second edition. Upper Saddle River, New Jersey: Prentice Hall, 2002.
4. Yang Q.: Intelligent Planning. A Decomposition and Abstraction Based Approach, Springer-Verlag, 1997.
5. Minton S., Carbonell J.G., Knoblock C.A., Kuokka D.R., Etzioni O., Gil Y.: Explanation-based learning: A problem-solving perspective. Artificial Intelligence 40:63-118, 1989.
6. Veloso M., Carbonell J.: Derivational Analogy in PRODIGY: Automatic Case Aquisition, Storage, and Utilization. Machine Learning 10(3): 249-278, 1993.
7. Skowron A., Stepaniuk J.: Informatuion Granules and Rough-Neural Computing, In: S.K. Pal, L. Polkowski, A. Skowron (eds.), Rough-Neural Computing: Techniques for Computing with Words, Cognitive Technologies Series, Springer-Verlag, Berlin, 43-84, 2004.
8. Nguyen S.H., Bazan J.A., Nguyen H.S., Skowron, A.: Layered Learning for Concept Synthesis, Journal LNCS Transactions on Rough Sets, Heidelberg, Springer-Verlag vol. 1, LNCS 3100, 193-214, 2004.

9. Wilkins D.E., desJardins M.: A call for knowledge-based planning. AI Magazine, 22(1): 99-115, 2001.
10. Tate A., Drabble B., Kirby R.: O-plan2: An open architecture for command planning and control. In M.Fox and M. Zweben, editors, Intelligent Scheduling. Morgan Kaufmann, 1994.
11. Haslum P., Scholz U.: Domain Knowledge in Planning: Representation and Use. In Proc. ICAPS 2003 Workshop on PDDL.
12. Kvarnström J., Doherty P.: Talplanner: A temporal logic based forward chaining planner. Annals of Mathematics and Articial Intelligence, 30:119-169, 2001.
13. Wilkins D.E.: Domain-independent planning: representation and plan generation. Articial Intelligence, 22:269-301, 1984.
14. Aler R., Borrajo D.: On control knowledge acquisition by exploiting human-computer interaction. In Malik Ghallab, Joachim Hertzberg, and Paolo Traverso, editors, Proceedings of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS-02), 112-120, Toulouse (France), April 2002. AAAI Press.
15. Mitchell T.: Machine Learning. Mc Graw Hill, 1998.