Tractable Class of a Problem of Finding Supports *

Student: Pavel Surynek
Supervisor: Roman Barták

Charles University
Faculty of Mathematics and Physics
Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic
{surynek, bartak}@ktiml.mff.cuni.cz

Abstract. We study a tractable class of a problem of finding supporting actions for a goal (briefly support problem) using our method called projection global consistency. This problem is exactly what the GraphPlan planning algorithm must solve many times during plan extraction from the planning graph. The support problem was shown to be *NP*-complete. However, we found that there exists a well defined tractable subclass. We are viewing the support problem as a special intersection graph. If the intersection graph is acyclic then the support problem can be solved in polynomial time. On this basis we propose a heuristic which prefers tractable cases of the problem. The performed experiments showed that preference of tractable cases brings significant improvement in the overall solving time compared to state-of-the-art planners on selected problems.

1 Introduction

We describe a class of a problem of finding supporting actions for a goal in AI planning in this paper. We developed a polynomial time algorithm for solving problems of this class (let us call it a *tractable class*). The tractable class and solving algorithm itself are based on a recently developed concept of projection global constraint [9].

The problem of finding supporting actions for a goal (briefly *support problem*) arises frequently as a sub-problem within the GraphPlan planning algorithm [2]. To be specific, the support problem must be solved many times when a plan is extracted from the planning graph in the backward search style. Since not all support problems arising during backward search belong to the tractable class (the problem is *NP*-complete [9]), we propose a special heuristic for preferring the class. More precisely, the proposed heuristic is trying to prefer decisions during the backward search (selections of actions) that most likely transform the general support problem to the problem belonging to the tractable class. After reaching the tractable class, the general backward search algorithm switches to the specialized polynomial solving algorithm for the problems of the class.

Despite the well known fact that the plan extraction from the planning graph based on backward search is not considered to be state-of-the-art step optimal planning

* This work is supported by the Czech Science Foundation under the contracts 201/07/0205 and 201/05/H014 and by GAUK under the contract 356/2006/A-INF/MFF.

approach, we show that plan extraction enhanced by the integration of the algorithm for the tractable class may improve the extraction process significantly. Moreover, we found that our approach is especially successful on difficult problems which force the planner to really perform search to find the solution or to prove that there is no solution [11]. Such problems typically encapsulates an instance of the Dirichlet's box principle (place $n+1$ pigeons into holes n so that no two pigeons are placed in the same hole). Although these problems are short in length of the input, they are hard to be answered. The solver does not see the principle encoded in the problem formulation. This property of the box principle makes the solver to perform exhaustive search to prove that the principle has no solution. The algorithm for the support problem of the tractable class has an advantage in such situation. It can detect insolvability of some sub-problems quickly in polynomial time and can prune large parts of the search space in this way.

The performed competitive comparison of our experimental planning system implementing the algorithm for the tractable class with several state-of-the-art planners on the mentioned difficult problems showed surprising results. We chose several planners participating in the International Planning Competition (IPC) [3] for our experimental evaluation. Although it was not our goal to compete with planners from the IPC by our experimental planner, the result was that our experimental planner performs better than some of the winners of the IPC on selected problems. Moreover, one of the most successful planners in IPC - SGPlan 5 [6] - provides incorrect answers, it solved several unsolvable problems. Another successful planner in IPC - SATPlan [7] - seems to be unable to prove nonexistence of the solution.

On the other hand, on the standard benchmark problems our experimental planning system performs significantly worse than planners from IPC. Nevertheless, this is expectable because our implementation is not so well optimized and does not use any domain specific heuristic.

2 Basics of Planning with Planning Graphs

Definition 1. An *atomic formula (atom)* is a construct of the form $p(c_1,c_2,\ldots,c_n)$ where $p \in P_L$ and $c_i \in C_L$ for $i=1,2,\ldots,n$. A *state* is a finite set of atoms. A *goal* is also a finite set of atoms. The goal g is *satisfied* in the state s if $g \subseteq s$. □

Definition 2. An *action* a is a triple $(p(a),e^+(a),e^-(a))$, where $p(a)$ is a *precondition* of the action, $e^+(a)$ is a *positive effect* of the action and $e^-(a)$ is a *negative effect* of the action. All these three action components are finite sets of atoms. An action a is *applicable* to the state s if $p(a) \subseteq s$. The *result of the application* of the action a to the state s is a new state $\gamma(s,a)=(s-e^-)\cup e^+$. □

Definition 3. A *planning problem* P is a triple (s_0,g,A), where s_0 is an *initial state*, g is a goal and A is a finite set of allowed actions. □

Definition 4. We inductively define *application of a sequence of actions* $\phi=[a_1,a_2,\ldots,a_n]$ to a state s_0 in the following way: a_1 must be applicable to s_0; let us inductively denote the result of application of the action a_i to the state s_{i-1} as s_i for all $i=1,2,\ldots,n$; the condition that a_i is applicable to the state s_i for all $i=1,2,\ldots,n-1$ must hold. The *result of application of the sequence of actions* ϕ to the state s_0 is the state s_n. Sequence $\xi=[a_1,a_2,\ldots\ \ldots,a_n]$ is a *solution* of the planning problem $P=(s_0,g,A)$ if the sequence ξ is applicable to the initial state s_0 and

the goal g is satisfied in the result of application of the sequence ξ and $a_i \in A$ for all $i = 1,2,\ldots,n$. □

The *GraphPlan* [1] algorithm relies on the idea of state reachability analysis. The state reachability analysis is done by constructing a special data structure called *planning graph*. The algorithm itself works in two interleaved phases. In the first phase planning graph is incrementally expanded. Then in the second phase an attempt to extract a valid plan from the extended planning graph is performed. The GraphPlan algorithm uses the standard backtracking to extract a plan from the planning graph. If the second phase is unsuccessful the process continues with the first phase. That is the planning graph is extended again.

The planning graph for a planning problem $P = (s_0, g, A)$ is defined as follows. It consists of two alternating structures called a *proposition layer* and an *action layer*. The initial state s_0 represents the *0*th proposition layer P_0. The layer P_0 is just a list of atoms occurring in s_0. The rest of the planning graph is defined inductively. Consider that the planning graph with layers P_0, A_1, P_1, A_2, P_2, …, A_k, P_k has been already constructed (A_i denotes the *i*th action layer, P_i denotes the *i*th proposition layer). The next action layer A_{k+1} consists of actions whose preconditions are included in the *k*th proposition layer P_k and which satisfy the additional condition that no two preconditions of the action are *mutually excluded*. The next proposition layer P_{k+1} consists of all the positive effects of actions from A_{k+1}.

Definition 5. A pair of actions $\{a,b\}$ is *independent* if $e^-(a) \cap (p(b) \cup e^+(b)) = \varnothing$ and $e^-(b) \cap (p(a) \cup e^+(a)) = . = \varnothing$. Otherwise $\{a,b\}$ is a pair of *dependent* actions. □

Definition 6. We call a pair of actions $\{a,b\}$ within the action layer A_i a *mutex* if either the pair $\{a,b\}$ is dependent or an atom of the precondition of the action a is mutex with an atom of the precondition of the action b (defined in the following definition). □

Definition 7. We call a pair of atoms $\{p,q\}$ within the proposition layer P_i a *mutex* if every action a within the layer A_i where $p \in e^+(a)$ is mutex with every action b within the action layer A_i for which $q \in e^+(b)$ and the action layer A_i does not contain any action c for which $\{p,q\} \subseteq e^+(c)$. □

3 Projection Consistency and Tractability of Support problem

A problem of finding supporting actions for a goal [9] is defined for an action layer of the planning graph and for an arbitrary goal. Let A be a set of actions of the action layer and let μA be a set of mutexes between actions from A. Next let us have a goal g. The task is to determine a set of actions $\zeta \subseteq A$ where no two actions from ζ are mutex with respect to μA and ζ satisfies the goal g. The problem of finding supports for a sub-goal will be called a *support problem* in short. The effectiveness of a method for solving support problem has a major impact on the performance of the planning algorithm as a whole. Unfortunately the problem is *NP*-complete [9].

In order to be able to enforce projection consistency we must construct a clique decomposition of a mutex graph of a given action layer of the planning graph first. Let $G = (A, \mu A)$ be an undirected mutex graph. The task is to find a partitioning of the set of vertexes $A = C_1 \cup C_2 \cup \ldots \cup C_n$ such that $C_i \cap C_j = \varnothing$ for every

$i, j \in \{1,2,\ldots,n\}$ & $i \neq j$ and C_i is a clique with respect to μA for $i = \{1,2,\ldots,n\}$. Let us denote $mA = \mu A - (C_1^2 \cup C_2^2 \cup \ldots \cup C_n^2)$ the set of mutexes outside the clique decomposition (where $C^2 = \{\{u,v\} \mid u,v \in C \,\&\, u \neq v\}$). Our objective is to minimize n and $|mA|$. The problem of clique decomposition of the defined property is obviously *NP*-complete on a graph without any restriction [5]. In [9] it is suggested to use a simple greedy algorithm.

Projection consistency is defined over the above decomposition for a goal p. The goal p is called a *projection goal* in this context.

Definition 8. A *contribution* of a clique $C \in \{C_1, C_2, \ldots \ldots, C_n\}$ to the projection goal p is defined as $\max(|e^+(a) \cap p| \mid a \in C)$. It is denoted by $c(C, p)$. □
Definition 9. An action $a \in C_i$ for $i \in \{1,2,\ldots,n\}$ is *(strongly) supported* with respect to the *(strong) projection consistency* with the projection goal p if $\sum_{j=1, j\neq i}^{n} c(C_j, p - e^+(a)) \geq |p - e^+(a)|$ holds. □
Definition 10. The preprocessed instance of the support problem consisting of actions $A = C_1 \cup C_2 \cup \ldots \cup C_n$, mutexes μA and the goal g is *(strong) projection consistent* with respect to a projection goal $p \subseteq g$, $p \neq \emptyset$ if every action $a \in C_i$ for $i = 1,2,\ldots,n$ is (strongly) supported. □

The concept of clique contribution is helpful when we are trying to decide whether it is possible to satisfy the projection goal using the actions from the clique cover. If for instance $\sum_{i=1}^{n} c(C_i, p) < |p|$ holds then the projection goal p cannot be satisfied and hence neither $g \supseteq p$ can be satisfied.

If cliques of the clique cover are regarded as CSP variables [2] and actions from the cliques are regarded as values for these variables then we can introduce a *projection constraint*. To enforce projection consistency over the support problem for some projection goal p we can easily remove values from the domains of variables. The propagation algorithm for the projection consistency can be implemented to run in $O(|p||A|)$ steps [9]. The interesting property of projection consistency is that it can be enforced with respect to multiple projection goals. Each projection goal provides a different set of inconsistent actions. In order to obtain maximum pruning power we need to use multiple projection goals. The trouble is that there are too many projection goals $p \subseteq g$ for a goal g (exactly $2^{|g|}$). In [9] it is argued that sets of atoms with constant number of supports should be selected as projection goals.

Definition 11. For a clique $C \in \{C_1, C_2, \ldots, C_n\}$ of the action clique decomposition we define a *merged effect* as $\bigcup_{a \in C} e^+(a)$. It is denoted as $me^+(C)$. □
Definition 12. We define a *clique intersection graph* $G_I = (\{C_1, C_2, \ldots, C_n\}, E_I)$ for the action clique decomposition $A = C_1 \cup C_2 \cup \ldots \cup C_n$ as an undirected intersection graph of corresponding merged effects. That is $E_I = \{\{C_i, C_j\} \mid i \neq j \,\&\, me^+(C_i) \cap \cap me^+(C_j) \neq \emptyset\}$. □

Lemma 1. Let $G_I = (V_I, E_I)$ be a clique intersection graph. If the graph G_I is acyclic then a problem of satisfying a goal g by selecting just one action a_i from the clique C_i for every $i = \{1,2,\ldots,n\}$ can be solved in polynomial time after enforcing strong projection consistency. □

Proofs of the lemma 1 and of the following lemmas and theorems are provided in [10]. The problem from the lemma 1 can be completely solved in polynomial time. We use the similar idea as that is commonly used to enforce arc-consistency in an

acyclic constraint network [2]. It is possible to enforce arc-consistency in such a network by enforcing directed arc-consistency in the direction from the root to the leaves of the network and then from the leaves to the root. Almost the same can be done for projection consistency. First we enforce the consistency for the projection goals $g \cap (me^+(C_i) - \bigcup_{j=1,\, j\neq i}^{n} me^+(C_j))$ for every $i = \{1,2,\ldots,n\}$ which is easy because no interference with other cliques occurs. Then cliques of the decomposition are ordered according to the breadth first search and strong projection consistency is enforced over the edges of the intersection graph. It is done in the direction from the root to the leaves of the clique intersection graph first and then from the leaves to the root.

Definition 13. A *mutex network* for the action clique decomposition $A = C_1 \cup C_2 \cup \ldots \cup C_n$ and for the set of mutexes outside the decomposition mA is a graph $G_m = (\{C_1, C_2, \ldots, C_n\}, E_m)$, where $E_m = \{\{C_i, C_j\} \mid i \neq j \;\&\; (\exists a_i \in C_i,\ \exists a_j \in C_j)\ \{a_i, a_j\} \in mA\}$. □

Lemma 2. Let $G_m = (V_m, E_m)$ be a mutex network. If the graph G_m is acyclic then a problem of selecting just one action a_i from the clique C_i for every $i = \{1,2,\ldots,n\}$ such that no two selected actions are mutex with respect to mA can be solved in polynomial time. □

Theorem 1. Let $G_I = (V_I, E_I)$ be a clique intersection graph and let $G_m = (V_m, E_m)$ be a mutex network. If the graph $G = (\{C_1, C_2, \ldots, C_n\}, E_I \cup E_m)$ is acyclic then the corresponding support problem can be solved in polynomial time. □

The major obstacle is that not every instance of the support problem belongs to the described class. Determining the smallest set of vertexes (cycle-cut-set) which removal makes the graph acyclic (problem tractable) is *NP*-complete [2]. However, for our purposes we do not need optimal cycle-cut-set. For selecting actions we suggest to use highest degree heuristics. That is an action from the clique of the highest degree in the clique intersection graph is selected first. Our experiments showed that this heuristics sufficiently prefers the tractable case.

4 Experimental Results

Our experimental planning system follows the standard GraphPlan algorithm except the part for solving the support problem. For this we use the heuristic for preferring the tractable case (action from a clique of the highest degree is preferably selected) and when the tractable case is reached we switch to maintaining strong projection consistency. We compared our planner with several state-of-the-art planners. We selected planners MaxPlan [13], SATPlan [7], CPT 1.0 [12] and IPP 4.1 [8], SGPlan 5.1 [6], and LPG-td 1.0 [4]. The set of testing problems can be found at http://ktiml.mff.cuni.cz/~surynek/research/cp2007. The results are shown in table 1. All tests were performed on the machine with two AMD Opteron 242 processors (1,6 GHz) and 1 GB of memory under Mandriva Linux 10.2.

The improvement obtained by using tractable class preference algorithm is in order of magnitudes compared to the selected state-of-the-art planners on the selected problems. Although the selected class of testing problems is rather limited, it represents an

important class of difficult problems [11] which intrinsically requires search to be answered (the solution cannot be guessed).

Table 1. Comparison of selected state-of-the-art planners with our experimental planning system on several hard planning problems encoding Dirichlet's box principle. The symbol N/A indicates that a comparison cannot be performed.

Instance	Our planner (seconds)	Speedup ratio w.r.t SGPlan 5.1	Speedup ratio w.r.t IPP 4.1	Speedup ratio w.r.t MaxPlan	Speedup ratio w.r.t SATPlan	Speedup ratio w.r.t CPT	Speedup ratio w.r.t LPG-td
ujam-02_01	**0.06**	N/A	N/A	N/A	N/A	**1.00**	N/A
ujam-03_02	**0.54**	N/A	**0.02**	**0.04**	N/A	N/A	***9.25***
ujam-04_03	**3.39**	N/A	**0.19**	**0.30**	N/A	N/A	***1.76***
ujam-05_04	**23.88**	N/A	***3.50***	**0.35**	N/A	N/A	**0.37**
ujam-06_05	**177.9**	N/A	***> 3.37***	***> 3.37***	N/A	N/A	***> 3.37***
jam-06_05	**1.29**	N/A	***20.00***	***2.14***	**0.71**	***177.32***	N/A
jam-07_06	**2.48**	N/A	***> 241.93***	***12.46***	***1.21***	***> 241.93***	N/A
jam-08_07	**4.60**	N/A	***> 130.43***	***49.56***	***3.19***	***> 130.43***	N/A
jam-09_08	**8.77**	N/A	***> 68.42***	***> 68.42***	***17.33***	***> 68.42***	N/A
jam-10_09	**17.05**	N/A	***> 35.19***	***> 35.19***	***> 35.19***	***> 35.19***	N/A
holes-06_05	**0.27**	N/A	**0.44**	***+ 1000.00***	N/A	N/A	***18.51***
holes-07_06	**0.55**	N/A	***3.44***	***> 1090.00***	N/A	N/A	***10.90***
holes-08_07	**1.08**	N/A	***28.09***	***> 555.55***	N/A	N/A	***12.96***
holes-09_08	**2.08**	N/A	***276.25***	***> 288.46***	N/A	N/A	***> 288.46***
holes-10_09	**4.06**	N/A	***> 147.78***	***> 147.78***	N/A	N/A	***> 147.78***

References

1. Blum, A. L., Furst, M. L.: Fast Planning through planning graph analysis. Artificial Intelligence 90, 281-300, AAAI Press, 1997.
2. Dechter, R.: Constraint Processing. Morgan Kaufmann, 2003.
3. Gerevini, A., Bonet, B., and Givan, B. (editors): 5th International Planning Competition. Event in the context of ICAPS 2006 conference, United Kingdom, 2006, http://ipc5.ing.unibs.it, University of Brescia, Italy, June 2006.
4. Gerevini, A. and Serina, I.: Homepage of LPG. Research web page, http://zeus.ing.unibs.it/lpg/, University of Brescia, Italy, (April 2007).
5. Golumbic, M. C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, 1980.
6. Hsu, C. W., Wah, B. W., Huang, R., and Chen, Y. X.: SGPlan 5: Subgoal Partitioning and Resolution in Planning. Research web page. http://manip.crhc.uiuc.edu/programs/SGPlan/index.html, University of Illinois, USA, (April 2007).
7. Kautz, H., Selman, B., and Hoffmann, J.: SATPLAN. Research web page. http://www.cs.rochester.edu/u/kautz/satplan/index.htm, University of Rochester, USA, (April 2007).
8. Koehler, J.: Homepage of IPP. Research web page, http://www.informatik.uni-freiburg.de/~koehler/ipp.html, University of Freiburg, Germany, (April 2007).
9. Surynek, P.: Projection Global Consistency: An Application in AI Planning. Accepted to CSCLP 2007 - ERCIM workshop, France, 2007.
10. Surynek, P.: Tractable Class of a Problem of Finding Supporting Actions for a Goal in AI Planning, submitted to CP 2007, Technical programme.
11. Urquhart, A.: Hard examples for resolution. Journal of the ACM, Volume 34, 209-219, ACM Press, 1987.
12. Vidal, V. and Geffner, H.: CPT Description. Research web page, http://www.cril.univ-artois.fr/~vidal/cpt.html, Université D'Artois, France, (April 2007).
13. Zhao, X., Chen, Y., Zhang, W., and Huang, R.: MaxPlan. Research web page. http://www.cse.wustl.edu/~chen/maxplan/, Washington University in St. Louis, USA, (April 2007).