Tractable Classes of a Problem of Finding Supporting Actions for a Goal in AI Planning

Pavel Surynek

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

Abstract

We are studying tractable classes of a problem of finding supporting actions for a goal using projection global consistency. The projection consistency is a recent technique designed to prune the search space along the search for supports for a sub-goal in AI planning context. The problem of finding supporting actions for a sub-goal (briefly supports problem) is exactly what the GraphPlan planning algorithm must solve many times during plan extraction from the planning graph. The supports problem was shown to be *NP*-complete. We found that there exist tractable instances of the problem. We define a special intersection graph according to the supports problem. We showed that if the intersection graph is acyclic then the supports problem can be solved in polynomial time by using projection consistency. On this basis we propose a heuristic which prefers tractable cases of the problem and we integrate it into our experimental planning system written in C++. The performed experiments showed that preference of tractable cases bring significant improvement in the number of backtracks as well as in the overall solving time.

Introduction

In this paper, we describe a class of a problem of finding supporting actions for a goal in AI planning. The class is interesting for practical use since a polynomial time solving algorithm for the problems of this class exists.

Planning as a task of finding a sequence of actions resulting in achieving some goal is one of the most challenging problems of artificial intelligence (Allen et al., 1990). The need of solving planning problems arises almost every time when a complex autonomous behavior of a certain agent is required (Ai-Chang et al., 2004; Bernard et al., 1998). There are many successful approaches of how to solve planning problems. One of them is usage of so called *planning graphs*. The concept of planning graphs introduced by Blum and Furst (Blum, Furst, 1997) brought a substantial break-through in solving of planning problems. Many of the consequent achievements in planning are based the idea of planning graphs (Kambhampati, 2000; Lopez and Bacchus, 2003). In this paper we are studying planning graphs from the perspective of constraint programming (Dechter, 2003). Particularly we use recently proposed projection global consistency (Surynek, 2007b) to define a tractable class of a certain sub-problem which arises during search for a plan by the GraphPlan algorithm.

Organization of the paper is following. First we recall basic definitions from AI planning, constraint programming and projection consistency. Then we introduce stronger version of the projection consistency and the mentioned tractable class of the problem. Finally we show some experimental results and discuss our contribution in relation to other works.

Basic Definitions from Planning and CP

To describe a planning problem we use a finite set of predicates P_L and a finite set of constants C_L.

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 $\gamma(s,a) = (s - e^-) \cup e^+$. □

For every atom t we also assume a so called no-op action $noop_t = (t,t,\varnothing)$. Briefly said a no-op action preserves an atom into the next state. For reasoning about complexity we suppose that the number of preconditions, the number of positive effects and the number of negative effects are bounded by a constant. Given a set of actions and a goal the objective is to transform a given initial state into a state satisfying the goal.

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 $\xi = [a_1,a_2,\ldots,a_n]$ 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 $i = 1,2,\ldots,n$. □

Definition 5. A *constraint satisfaction problem* (CSP) is a triple (X,D,C), where X is a finite set of variables, D is a finite domain of values for variables from the set X and C is a finite set of constraints. A *constraint* is an arbitrary relation over the domains of its variables. □

Definition 6. A *solution* of a constraint satisfaction problem (X,D,C) is an assignment of values to the variables $\psi : X \to D$ such that all the constrains are satisfied for ψ, that is $(\forall c \in C)[x_1,x_2,\ldots,x_k] = X_c \Rightarrow [\psi(x_1),\psi(x_2),\ldots$ $\ldots,\psi(x_k)] \in C$ (X_c denotes variables constrained by the constraint c). □

Planning Graphs and GraphPlan Algorithm

The *GraphPlan* 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 propositions 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 7. A pair of actions $\{a,b\}$ is *independent* if $e^-(a) \cap (p(b) \cup e^+(b)) = \emptyset$ and $e^-(b) \cap (p(a) \cup e^+(a)) = \emptyset$. Otherwise $\{a,b\}$ is a pair of *dependent* actions. □

Definition 8. 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 9. 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 $\{p,q\} \subseteq e^+(c)$. □

Projection Consistency

A problem of finding supporting actions for a goal (Surynek, 2007b) is defined for an action layer of the planning graph and for an arbitrary goal. Briefly said, we want to find a set of actions from the action layer that satisfies the given goal and that do not conflict with each other. The formal definition of the problem is following. 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 (that is $(\forall a \in \zeta, \forall b \in \zeta) \mu A \cap \{\{a,b\}\} = \emptyset$ and $g \subseteq \bigcup_{a \in \zeta} e^+(a)$). The problem of finding supports for a sub-goal will be called a *supports problem* in short. The effectiveness of a method for solving supports problem has a major impact on the performance of the planning algorithm as a whole. Unfortunately the supports problem is *NP*-complete. The proof can be found in (Surynek, 2007b). Hence it is unlikely that the supports problem can be solved without search in general.

In (Surynek, 2007a) Surynek studied maintaining arc-consistency for solving the supports problem. Compared to the standard backtracking he obtained reasonable speedups. In (Surynek, 2007b) he proposed another method which he called a *projection consistency*. The projection consistency and associated projection constraint propagation algorithm provide a certain type of global reasoning over the supports problem.

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 a 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 $C_i \cap C_j = \emptyset$ 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. Our objective is to minimize n and $|mA|$. Unfortunately the problem of clique decomposition of the defined property is obviously *NP*-complete on a graph without any restriction (Golumbic, 1980). Surynek suggests using a simple greedy algorithm. We will follow this suggestion too.

For the following description assume an action layer of the planning graph for which a clique cover $A = C_1 \cup C_2 \cup \ldots \cup C_n$ of the set of actions A with respect to the set of mutexes μA was computed. Next let mA be se set of mutexes outside the clique cover. Projection consistency is defined over the above decomposition for a goal p. The goal p is called a *projection goal* in this context.

Definition 10. A *contribution* of a clique $C \in \{C_1, C_2, \ldots, C_n\}$ to the projection goal p is defined as $\max(|e^+(a) \cap p| \mid a \in C)$. It is denoted by $c(C, p)$. □

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.

Definition 11. An action $a \in C_i$ for $i \in \{1,2,\ldots,n\}$ is *supported* with respect to *projection consistency* with the projection goal p if $\sum_{j=1, j\neq i}^{n} c(C_j, p) \geq |p - e^+(a)|$ holds. □

Definition 12. The preprocessed instance of the supports problem consisting of actions $A = C_1 \cup C_2 \cup \ldots \cup C_n$, mutexes μA and the goal g is *projection consistent* with respect to a projection goal $p \subseteq g$, $p \neq \varnothing$ if every action $a \in C_i$ for $i = 1,2,\ldots,n$ is supported. □

If cliques of the clique cover are regarded as CSP variables 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 supports problem for some projection goal p we can easily remove values from the domains of variables. The propagation algorithm for the projection consistency is shown in (Surynek, 2007b). It can be implemented to run in $O(|p||A|)$ steps. The projection consistency can be enforced with respect to multiple projection goals. The trouble is that there are too many projection goals $p \subseteq g$ for a goal g (exactly $2^{|g|}$). In (Surynek, 2007b) it is argued that sets of atoms with constant number of supports should be selected as projection goals. Such selection proved to be best in empirical tests. We will select projection goals according to this scheme too.

Tractable Class of the Supports Problem

It is possible to make projection consistency stronger by a slight reformulation of the definition of the supported action. The definition of the consistent problem remains the same. We will need this modified version of the projection consistency to be able to solve certain instances of the supports problem in polynomial time.

Definition 13. 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 $\sum_{j=1, j\neq i}^{n} c(C_j, p - e^+(a)) \geq$ $\geq |p - e^+(a)|$ holds. □

Let us call the projection consistency that uses the definition of strongly supported actions a *strong projection consistency.*

Proposition 1. If the supports problem is strongly projection consistent with respect to a projection goal p then it is projection consistent with respect to the projection goal p. Moreover there exists a supports problem which is projection consistent with respect to a projection goal p but it is not strongly projection consistent with respect to the same projection goal p. □

Proof. To prove the claim it is sufficient to observe that $\sum_{j=1, j\neq i}^{n} c(C_j, p - e^+(a)) \geq |p - e^+(a)| \Rightarrow$ $\Rightarrow \sum_{j=1, j\neq i}^{n} c(C_j, p) \geq |p - e^+(a)|$ for any $a \in C_i$ for $i = \{1,2,\ldots,n\}$ and for any projection goal p. Moreover there exists a supports problem and the projection goal p where for some $a \in C_i$ and $i \in \{1,2,\ldots,n\}$ inequalities $\sum_{j=1, j\neq i}^{n} c(C_j, p) \geq$ $\geq |p - e^+(a)|$ and $\sum_{j=1, j\neq i}^{n} c(C_j, p - e^+(a)) <$ $< |p - e^+(a)|$ hold. ■

The modification of the definition was simple. Unfortunately this is not true for the propagation algorithm. Our modification substantially changed the effect of removal of an unsupported action on the set of strongly supported actions with respect to a single projection goal. The set of supported actions does not change after removal of an unsupported action in projection consistency. This property of projection consistency is called a *monotonicity* in (Surynek, 2007b) and represents main argument for the low complexity of the propagation algorithm. For the strong projection consistency the monotonicity

does not hold (the set of supported actions may change). Fortunately this property does not matter for the (tractable) case we are about to investigate.

Definition 14. 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 15. 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 $E_I = \{\{C_i, C_j\} \mid i \neq j \;\&\; me^+(C_i) \cap me^+(C_j) \neq \emptyset\}$. □

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. □

Proof. We need to show that if the defined problem is strong projection consistent with respect to the certain projection goals then it is necessary to do only little to find solution or conclude that there is no solution. The projection goals are $g \cap (me^+(C_i) - \bigcup_{j=1,\, j \neq i}^{n} me^+(C_j))$ for every $i = \{1, 2, \ldots, n\}$ and $g \cap me^+(C_i) \cap me^+(C_j)$ for every $\{C_i, C_j\} \in E_I$. If $g - \bigcup_{i=1}^{n} me^+(C_i) \neq \emptyset$ holds then there is obviously no solution. This condition can be checked in $O(d\,|g|\,|A|)$ steps where d is the action size bounding constant. If $g \subseteq \bigcup_{i=1}^{n} me^+(C_i)$ holds then arbitrary selection of just one action a_i from the clique C_i for every $i = \{1, 2, \ldots, n\}$ which preserves relation of strong supports over the edges E_I solves the problem. This selection can be carried out by starting in the root clique of G_I and continuing to the leaves in breadth first order. It takes $O(d\,|g|\,|A|)$ steps to select actions in this way.

Consider an atom $t \in g$. There are at most two cliques for which the atom t is an element of their merged effect. This is due to the acyclicity of the corresponding clique intersection graph G_I. In the case when there is just one such clique C_i an action $a_i \in C_i$ that satisfies t must be selected. Let $p = g \cap (me^+(C_i) - \bigcup_{j=1,\, j \neq i}^{n} me^+(C_j))$, for such p we have $t \in p$ and $\sum_{j=1,\, j \neq i}^{n} c(C_j, p - e^+(a_i)) \geq |p - e^+(a_i)|$ since the problem is strong projection consistent with respect to the projection goal p. We also have $\sum_{j=1,\, j \neq i}^{n} c(C_j, p - e^+(a_i)) = 0$ since the sum is empty (no other clique intersects the projection goal p by its merged effect). Hence $|p - e^+(a_i)| = 0$ and $t \in e^+(a_i)$. Assume the case when there are two cliques C_i and C_j for which $t \in me^+(C_i)$ and $t \in me^+(C_j)$. Suppose that an action a_i is selected from the clique C_i and an action a_j from the clique C_j. Consider the projection goal $p = g \cap me^+(C_i) \cap me^+(C_j)$, both actions are strongly supported with respect to p. That is $\sum_{k=1, k \neq i}^{n} c(C_k, p - e^+(a_i)) \geq |p - e^+(a_i)|$ and $\sum_{k=1, k \neq j}^{n} c(C_k, p - e^+(a_j)) \geq |p - e^+(a_j)|$. Suppose that action a_i was selected before a_j. Since there are only two cliques interfering over the projection goal p, we specially have $c(C_j, p - e^+(a_i)) \geq |p - e^+(a_i)|$ after selecting a_i. Hence it is possible to select the action a_i such that $|p \cap e^+(a_j)| = c(C_j, p - e^+(a_i))$. Altogether we obtained that $t \in e^+(a_i) \cup e^+(a_j)$. ■

The question arises whether the strong projection consistency with respect to the projection goals mentioned in the proof of the lemma 1 can be enforced over the acyclic problem in polynomial time. The answer is positive and we can conclude that 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 (Dechter, 2003). 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 the strong 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 the 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. The complete algorithm is shown here as algorithm 1.

Proposition 2. The algorithm for enforcing strong projection consistency over a clique decomposition $A = C_1 \cup C_2 \cup \ldots \cup C_n$ and for a goal g can be implemented to run in polynomial time with respect to $|g|$ and $|A|$. □

Proof. The function *propagateProjection* takes $O(d|p|)$ steps for the projection goal $p = g \cap (me^+(C_i) - \bigcup_{j=1,\, j\neq i}^{n} me^+(C_j))$ for $i \in \{1,2,\ldots,n\}$ (line 4), where d is the action size bounding constant. For all such goals the line 4 takes $O(d|g|)$ steps in total. The function *propagateProjection* takes $O(d|p|^2)$ steps for the projection goal $p = g \cap me^+(C_{\pi(i)}) \cap me^+(C_{\pi(j)})$ for $\{C_i, C_j\} \in E_I$ (lines 9 and 13). The time for all such goals is bounded by $O(d|g|^2)$ (however this upper bound may be wasting). The breadth first search performed over the clique intersection graph (line 5) takes $O(n^2)$ steps which is $O(|g|)$. ■

Definition 16. 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. □

Proof. This is a well known result from constraint programming in fact. If each clique of the clique decomposition $A = C_1 \cup C_2 \cup \ldots \cup C_n$ is regarded as a CSP variable and mutexes of the set mA are regarded as constraints then the defined problem of selecting non-mutex actions is an acyclic constraint satisfaction problem. It is sufficient to enforce arc-consistency and label the variables in breadth first order to obtain a solution. More details about this result can be found in (Dechter, 2003). This solving algorithm runs in polynomial time with respect to $|A|$ and $|mA|$. ■

Algorithm 1: Strong projection consistency propagation algorithm for acyclic clique intersection graph

function *enforceProjectionConsistency*$(g, \{C_1, C_2, \ldots, C_n\})$: **set**
1: **let** $G_I = (\{C_1, C_2, \ldots, C_n\}, E_I)$ be the clique intersection graph
2: **for** $i = 1,2,\ldots,n$ **do**
3: $\quad p \leftarrow g \cap (me^+(C_i) - \bigcup_{j=1,\, j\neq i}^{n} me^+(C_j))$
4: $\quad \{C_1, C_2, \ldots, C_n\} \leftarrow$
$\quad \leftarrow propagateProjection(p, \{C_1, C_2, \ldots, C_n\})$
5: $\pi \leftarrow breadthFirstSearch(G_I)$
6: **for** $i = 1,2,\ldots,n$ **do**
7: $\quad$ **for each** $\{C_{\pi(i)}, C_{\pi(j)}\} \in E_I$
$\quad\quad$ such that $\pi(i) < \pi(j)$ **do**
8: $\quad\quad p \leftarrow g \cap me^+(C_{\pi(i)}) \cap me^+(C_{\pi(j)})$
9: $\quad\quad \{C_1, C_2, \ldots, C_n\} \leftarrow$
$\quad\quad \leftarrow propagateProjection(p, \{C_1, C_2, \ldots, C_n\})$
10: **for** $i = n, n-1, \ldots, 1$ **do**
11: $\quad$ **for each** $\{C_{\pi(i)}, C_{\pi(j)}\} \in E_I$
$\quad\quad$ such that $\pi(i) > \pi(j)$ **do**
12: $\quad\quad p \leftarrow g \cap me^+(C_{\pi(i)}) \cap me^+(C_{\pi(j)})$
13: $\quad\quad \{C_1, C_2, \ldots, C_n\} \leftarrow$
$\quad\quad \leftarrow propagateProjection(p, \{C_1, C_2, \ldots, C_n\})$
14: **return** $\{C_1, C_2, \ldots, C_n\}$

function *propagateProjection*$(p, \{C_1, C_2, \ldots, C_n\})$: **set**
15: **for** $i = 1,2,\ldots,n$ **do**
16: $\quad$ **for each** $a \in C_i$ **do**
17: $\quad\quad$ **if** $\sum_{j=1,\, j\neq i}^{n} c(C_j, p - e^+(a)) < |p - e^+(a)|$ **then**
18: $\quad\quad\quad C_i \leftarrow C_i - \{a\}$
19: **return** $\{C_1, C_2, \ldots, C_n\}$

function *breadthFirstSearch*$((\{v_1, v_2, \ldots, v_m\}, E))$: **permutation**
20: $\pi \leftarrow$ numbering of vertexes by breadth first search
21: **return** π

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 supports problem can be solved in polynomial time. □

Proof. To prove the theorem we use a combination of results from lemma 1, lemma 2 and proposition 2. The first step consists of enforcing strong projection consistency and arc-consistency in the supports problem. Since it is quite easy using the above results we describe the process briefly. If the interference of cliques is through an edge from E_I then strong projection consistency is enforced over the intersection of corresponding merged effects. If the interference of cliques is through an edge from E_m then arc-consistency with respect to mA is enforced. Again this combined consistency can be enforced in polynomial time by proceeding from the root to the leaves of the graph G and conversely. The extraction of a solution from the consistent problem can be also done in polynomial time. The extraction procedure starts by selecting action from the root clique and proceeds to the leaves of the graph G while strong projection consistency and arc-consistency relations are preserved over the edges of G. ■

We described the tractable class of the supports problem in order to utilize the theoretical results in solving real problems. The obstacle is that not every instance of the supports problem belongs to the described class. The figure 1 show a real example of the clique decomposition of the action layer of the planning graph of an instance of the Dock Worker Robots planning domain (for simplicity we do not depict mutexes outside the decomposition).

The problem obviously does not belong to our class. However it is very close to the class. The corresponding clique intersection graph (figure 2) can be made acyclic by removing a single vertex (vertex removal corresponds to the selection of an action from the clique). Unfortunately determining the smallest set of vertexes (cycle-cut-set) which removal makes the graph acyclic is *NP*-complete (Dechter, 2003). For our purposes we do not need optimal cycle-cut-set. Nevertheless the better the cycle-cut-set is the larger portion of the problem remains tractable. 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 (merged with corresponding mutex network) is selected first. Our experiments showed that this heuristics sufficiently prefers the tractable case.

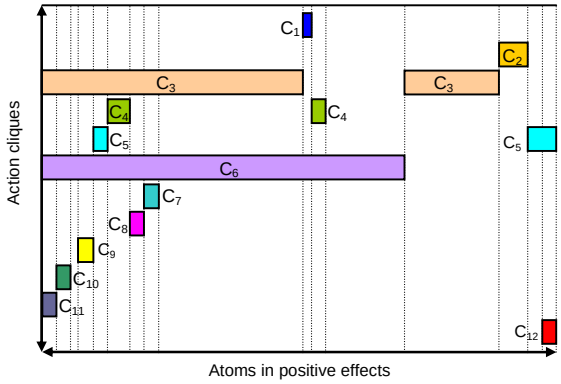

Figure 1: A diagram of merged positive effects of cliques of an action layer clique decomposition. Each line of the diagram represents a clique. The scope of the merged positive effect of the clique is depicted as one or more horizontal bars. The width of bars is proportional to the number of actions in the individual action cliques. The diagram was constructed according to an action layer of the planning graph for an instance of the Dock Worker Robots domain.

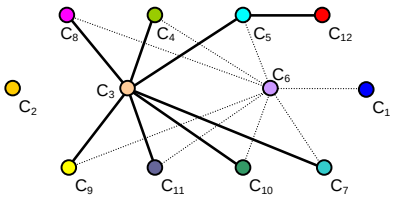

Figure 2: An intersection graph of merged positive effects of cliques of the action layer clique decomposition form figure 1. The effect of removal of the cycle cut-set consisting of the only one vertex C_6 is denoted by doted edges.

Experimental Results

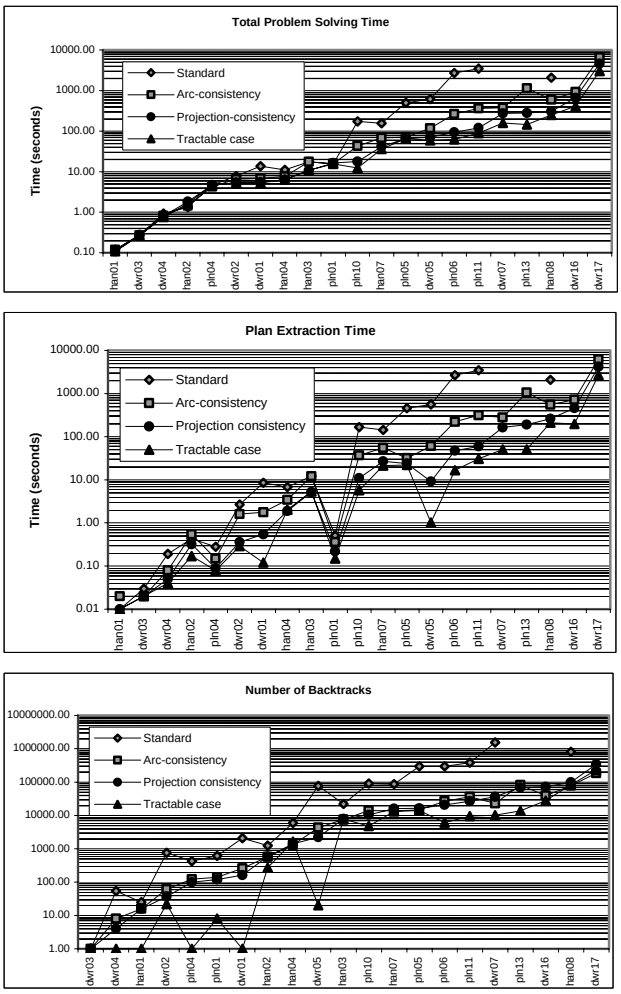

Figure 3: Experimental results over several planning problems of various difficulties. Total problem solving time, plan extraction time and number of backtracks are shown. Value ranges use logarithmic scale.

We evaluated the proposed approach in our experimental planning system written in C++. Our planning algorithm follows the standard GraphPlan algorithm except the part for solving the supports problem. For this we use maintaining (weak) projection consistency with the heuristic for preferring tractable case (action from a clique of the highest degree is preferably selected) and when the tractable case is reached we switch to the strong projection consistency as it is described in above paragraphs. Specifically, the tractable case preferring heuristic is used for value selection ordering. For variable ordering we use the standard first fail (smallest domain) heuristic. We also use an unrestricted nogood recording to improve the search.

For experimental tests we used the same set of planning problems as it was used in (Surynek, 2007b). The set of planning problems consists of several instances of various difficulties of Dock Worker Robots, Towers of Hanoi and Refueling Planes planning domain. We compared the proposed method with the standard GraphPlan, with the version which maintains arc-consistency when solving the supports problem (Surynek, 2007a) and with the version which maintains projection consistency (Surynek, 2007b). Results are shown in figure 3. The tests were run on a machine with two Opteron 242 processors (1600 MHz), with 1GB of memory under Mandriva Linux 10.2. The code was compiled by the gcc compiler 3.4.3 with maximum optimization for the machine (-O9 -mtune=opteron). The whole set of testing problems can be found at http://ktiml.mff.cuni.cz/~surynek/research/aaai2007.

The improvement in overall problem solving time (planning graph building time + clique decomposition time + plan extraction time) is up to 200% compared to the version which uses pure projection consistency. The improvement in plan extraction time is up to 1000%. The improvement in number of backtracks is also substantive. Even some problems were solved without backtracking. The improvements are better for problems with more interacting objects and higher action parallelism (for example dwr05 and pln13). On the other hand there is almost no improvement on problems with no action parallelism (for example han03), which is expectable.

Related Works and Conclusion

The main difference of our approach from other approaches exploiting another formalism (CSP, SAT) for solving planning problems (Kambhampati, 2000; Lopez and Bacchus, 2003) is that we

do not formulate the planning problem in another formalism as a whole. We use constraint programming approach only to solve a sub-problem arising during search.

Kambhampati's idea to formulate plan extraction from planning graph as CSP is presented in (Kambhampati, 2000). He evaluates the use of various constraint programming techniques and its impact on the effectiveness of plan extraction. Another approach is presented in (Lopez and Bacchus, 2003) by Lopez and Bacchus. Again they model the planning problem in planning graph representation as CSP. The originality of their technique consists in making transformations of the obtained CSP which uncovers additional structural information about the problem.

Our contribution can be summarized as follows. We described the tractable class of the supports problem using the recently proposed projection global consistency. Our experiments showed that this class is also useful for practical solving of planning problems since problems of this class arise (with some help) frequently.

However there is a lot of work for future. Using the projection consistency the time spent by solving the supports problem is no more a limiting factor of the planning algorithm. The limiting factor is rather the time spent by building planning graphs and by search across the layers of the planning graph. We account this to the size of the planning graph and namely to the high numbers of no operation actions in the planning graph. We consider that it would be interesting to reformulate planning graphs in order to be friendlier to the backward search. The expectable question is also how to extend the presented ideas for planning graphs with time and resources (Smith and Weld, 1999).

References

Ai-Chang, M., et al., 2004. *MAPGEN: Mixed-Initiative Planning and Scheduling for the Mars Exploration Rover Mission.* IEEE Intelligent Systems 19(1), 8-12, IEEE Press.

Allen, J., Hendler, J., Tate, A. (editors), 1990. *Readings in Planning.* Morgan Kaufmann.

Bernard, D. et al., 1998. *Remote Agent Experiment. Deep Space 1 Technology Validation Report.* NASA Ames and JPL report.

Blum, A. L., Furst, M. L., 1997. *Fast Planning through planning graph analysis.* Artificial Intelligence 90, 281-300, AAAI Press.

Dechter, R., 2003. *Constraint Processing.* Morgan Kaufmann.

Ghallab, M., Nau, D. S., Traverso, P., 2004. *Automated Planning: theory and practice.* Morgan Kaufmann.

Golumbic, M. C., 1980. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press.

Kambhampati, S., 2000. *Planning Graph as a (Dynamic) CSP: Exploiting EBL, DDB and other CSP Search Techniques in GraphPlan.* JAIR 12, 1-34, AAAI Press.

Lopez, A., Bacchus, F., 2003. *Generalizing GraphPlan by Formulating Planning as a CSP.* In Proceedings of IJCAI 2003, 954-960, Morgan Kaufmann.

Smith, D. E., Weld, D. S., 1999. *Temporal Planning with Mutual Exclusion Reasoning.* In Proceedings IJCAI 99, 326-337, Morgan Kaufmann.

Surynek, P., 2007a. *Maintaining Arc-consistency over Mutex Relations in Planning Graphs during Search.* Accepted to FLAIRS 2007, USA. Technical report available in ITI Series, 2007-328, http://iti.mff.cuni.cz/series, Charles University, Czech Republic.

Surynek, P., 2007b. *Projection Global Consistency: An Application in AI Planning.* Technical report, ITI Series, 2007-333, http://iti.mff.cuni.cz/series, Charles University, Czech Republic.