A Global Filtration for Satisfying Goals in Mutual Exclusion Networks

Pavel Surynek

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

Abstract. We formulate a problem of goal satisfaction in mutex networks in this paper. The proposed problem is motivated by problems that arise in concurrent planning. For more efficient solving of goal satisfaction problem we design a novel global filtration technique. The filtration technique is based on exploiting graphical structures of the problem - clique decomposition of the problem is used. We evaluated the proposed filtration technique on a set of random goal satisfaction problems as well as a part of GraphPlan based planning algorithm. In both cases we obtained significant improvements in comparison with existing techniques.

Keywords: global filtration, mutual exclusion network, search

1 Introduction

We propose a new global filtration method for satisfying goals in *mutual exclusion networks* in this paper. The mutual exclusion network is an undirected graph where a finite set of symbols is assigned to each vertex. The interpretation of edges is that a pair of vertices connected by an edge cannot be selected together. In other words, edges in the graph represent mutual exclusions of vertices (or conflicts between vertices). Having a goal, which is a finite set of symbols, the task is to select a stable set of vertices in this graph such that the union of their symbols covers the given goal.

This problem may seem artificial at first sight but in fact it is a slight reformulation of problems that appear in *concurrent planning* for artificial intelligence [1] with planning graphs [2] and in *Boolean formula satisfaction* [3]. In addition to these applications the defined problem may be generic enough to worth studying for itself (the more detailed motivation is given in section 2).

Existing techniques that can be used to solve the problem include variety of backtracking based search methods enhanced with consistency techniques which is a typical approach used in *constraint programming* practice [4] and which we are following too. Our experiments expectably showed that local consistency techniques (namely arc-consistency) can bring significant improvements in terms of overall solving time compared to plain backtracking. Nevertheless, this result invoked a question what an

improvement can be obtained by using a certain type of global consistency? We are trying to answer this question in this paper.

As a first step we investigated the possible usage of existing global constraints [4, 12] for modeling the problem. We considered several existing global constraints based on network flows (such as *allDifferent* and similar constraints; our idea was to simulate the problem as a network flow and then model it using these constraints). But it turned out that existing global constraints are not suitable for modeling the problem (the reason we identified as a main obstacle is the quite complicated relation between the impact of selection of a vertex on the rest of the mutual exclusion network and the goal we are trying to satisfy). The option was to develop our own specialized global filtering method for the problem which we eventually did.

Our filtering method exploits the structural information encoded in the problem. Valuable structural information in the mutual exclusion network is a *complete subgraph* (clique). More precisely we can extract this structural information if we have a clique decomposition of the network.

If we know that several vertices in the graph form a clique we also know that at most one of them can be selected into the solution. This simple property allows us to do further relaxed reasoning. The clique of vertices can be treated as single entity with a limited contribution to the solution (since only one vertex can be selected which typically means that not a lot symbols in the goal can be covered by the clique). Then we can check relaxed condition on selection of a vertex into the solution. A vertex can be selected into the solution if the maximum number of symbols that can be obtained from the remaining cliques plus the number of symbols of the vertex is not lower than the number of symbols in the goal. This condition is necessary but not sufficient, however if it does not hold we can filter out the vertex from further consideration.

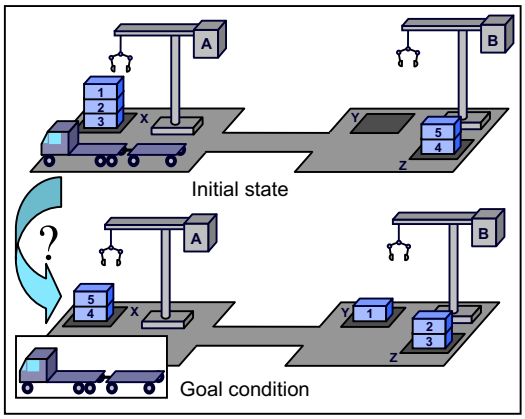
This paper is organized as follows. First we give more details about our motivation to deal with the problem. In the next two sections we introduce some formalism through which we will express the problems and we discuss some complexity issues. The fourth section is devoted to the description of our new global filtration. In the main part, we evaluate our approach using a set of benchmarks. Finally, we put our work into relation with existing works on the similar topic and we sketch out some ideas for future development.

2 Motivation by AI Problems

We would like to give a motivation for studying goal satisfaction problems in mutual exclusion networks in this section. Generally, we have two sources of motivation from artificial intelligence - the first is *concurrent planning* and the second is *Boolean formula satisfaction*.

The main motivation for our work was *concurrent planning* known from artificial intelligence [6,7,9]. To provide better insight into our motivation let us introduce concurrent planning briefly. Although the formalism and theory around concurrent planning is quite complex the basic idea is simple. Let us have a certain planning world (as an example we can consider a planning world shown in the upper part of figure 1). The task we want to fulfill is to transform the given planning world by exe-

cuting actions from a set of allowed actions into a state that satisfies certain goal condition (as an example of the goal condition for a planning world we can take the planning world shown in the lower part of figure 1). An action in this context is an elementary operation that locally changes the planning world (such an elementary operation in figure 1 is for example *take box 1 by crane A*).



Concurrent plan for reaching the goal

{take(craneA, box1, pileX); take(craneB, box5, pileZ)}
{load(truck, craneA, box1); put(craneB, box5, pileY)}
{move(truck, siteA, siteB); take(craneB, box4, pileZ); take(craneA, box2, pileX)}
{load(truck, craneB, box4); put(craneA, box2, pileX)}
{take(craneB, box5, pileY); take(craneA, box2, pileX)}
{put(craneB, box5, pileZ)}
{put(craneA, box2, pileX); unload(truck, craneB, box1)}
{move(truck, siteB, siteA); put(craneB, box1, pileY); take(craneA, box2, pileX)}
{load(truck, craneA, box2); take(craneB, box1, pileY)}
{move(truck, siteA, siteB); put(craneB, box1, pileY); take(craneA, box3, pileX)}
{put(craneA, box3, pileX); unload(truck, craneB, box2)}
{move(truck, siteB, siteA); put(craneB, box2, pileY); take(craneA, box3, pileX)}
{load(truck, craneA, box3); take(craneB, box5, pileZ)}
{put(craneB, box5, pileY); unload(truck, craneA, box4)}
{move(truck, siteA, siteB); take(craneB, box5, pileY)}
{load(truck, craneB, box5); put(craneA, box4, pileX)}
{take(craneA, box4, pileX); unload(truck, craneB, box3)}
{move(truck, siteB, siteA); put(craneA, box4, pileX); put(craneB, box3, pileZ)}
{take(craneB, box2, pileY); unload(truck, craneA, box5)}
{put(craneA, box5, pileX); put(craneB, box2, pileZ)}

Fig. 1. An example of planning problem. The task is to transform the initial state of a given planning world into a planning world satisfying the goal condition (in the goal condition we do not care where the truck is located). A concurrent solution plan is in the right part of the figure.

In the basic variant of planning problems we are searching for a sequence of actions that, when executed one by one starting in the given planning world, results into the planning world that satisfies the goal. The concurrent planning itself represents a generalization of this basic variant. Particularly, we allow more than one action to be executed in a single step in concurrent planning. This generalization is motivated by the fact that certain actions do not interfere with each other and they can be executed simultaneously without influencing each other (such non-interfering actions in figure 1 in the upper part are for example *take box 1 by crane A* and *take box 5 by crane B*; the pair of actions *load box 1 by crane A on truck* and *move truck from left location to right location* do interfere). Thus, the task in concurrent planning is to find not just a sequence of actions but a sequence of sets of non-interfering actions that when executed starting in the given planning world results into a goal satisfying state. The execution of a sequence of sets of actions means that we are executing sets of actions one by one where actions from each set are executed simultaneously. This is allowed by the fact that actions in each set of the sequence do not interfere.

But what is the relation between the concurrent planning and the mutual exclusion network mentioned in the introduction? The frequently asked question which arise during solving process of algorithms for concurrent planning is “What are the sets (or is there any) of non-interfering actions that satisfies certain goal?”. To be more concrete, this question often arises during the solving process with the usage of the framework of so called *planning graphs* [2]. This is the case of the pioneering *GraphPlan* algorithm as well as of its modern derived variants [8,9,10]. This question can be directly modeled as a mutual exclusion network (actually we borrowed the

term mutual exclusion from the planning graph terminology), where the vertices of the network are represented by actions (a set of terms that forms the effect of the action is assigned to the corresponding vertex as a set of symbols) and edges of the network are represented by pairs of actions that interfere with each other.

The minor motivation to study the concept of mutual exclusion networks is *Boolean formula satisfiability*. We found that Boolean formula satisfaction problems (*SAT*) [3, 19] can be modeled as mutual exclusion networks. This issue is studied in more details in [17, 18], therefore we mention it as a motivation only. However, let us note that a SAT problem consists in finding of a valuation of Boolean variables that satisfies a given formula in *conjunctive normal form* (*CNF* - conjunction of clauses where clause is a disjunction of literals).

Intuitively, it is possible to observe that such modeling can be done by declaring literals to be vertices of the network where each vertex (literal) has assigned a set of clauses in which it appear as its set of symbols. The goal would be the set of all the clauses of the given formula and edges in the network would connect vertices (literals) which are conflicting (in the most trivial case, literals x and $\neg x$, where x is a variable, are conflicting).

These two areas of application of our concept of global filtering are especially suitable since they often contain properties of objects that behave like functions. That is, a single value can be assigned to the property of an object or of a group of objects at the moment (for example imagine a robot at coordinates $[3,2]$, the robot can move to coordinates in its neighborhood, so the possible actions are: $moveTo([2,2])$, $moveTo([2,3])$, …; the robot can choose only one of these actions at the moment; executing more than one action at once is physically implausible). Such functional property typically induces a complete sub-graph in the mutual exclusion network.

3 Mutual Exclusion Network and Related Problem

We define mutual exclusion network and problems associated with it formally in this section.

The following definitions formalize mutual exclusion network (shortly *mutex network*) and the associated *problem of satisfying goals* in the mutex network. We assume a finite universe of symbols S for the following definitions.

Definition 1 (Mutual exclusion network). Mutual exclusion network is an undirected graph $N = (V, M)$, where a finite set of symbols $\emptyset \neq S(v) \subseteq S$ is assigned to each vertex $v \in V$. □

Definition 2 (Goal satisfaction in mutex network). Given a goal $G \subseteq S$ and a mutex network $N = (V, M)$ the problem of satisfying goal G in the mutex network N is the task of finding a stable set of vertices $U \subseteq V$ such that $G \subseteq \bigcup_{u \in U} S(u)$. □

An example of mutual exclusion network and a problem of goal satisfaction in this network are shown in figure 2.

The problem of goal satisfaction in mutex network is computationally difficult. To show this claim we can use a polynomial time reduction of the Boolean formula satisfaction problem to the problem of goal satisfaction in mutex network. Then it remains

only little to conclude that the problem of goal satisfaction in mutex network is *NP*-complete.

Theorem 1 (Complexity of goal satisfaction). *The problem of goal satisfaction in mutex network is NP-complete.* ■

Sketch of proof. If we are given a set of vertices we are able to decide whether it is a solution of the problem or not in polynomial time. Hence, the problem is in *NP* class. *NP*-hardness can be proved by using polynomial time reduction of Boolean formula satisfaction problem (*SAT*) to the problem of goal satisfaction in mutex network. Consider a Boolean formula B over a set of Boolean variables. It is possible to assume that the formula B is in the form of conjunction of disjunctions, that is $B = \wedge_{i=1}^{n} \vee_{j=1}^{m_i} x_j^i$, where x_j^i is a variable or a negation of a variable (literal). For each clause $\vee_{j=1}^{m_i} x_j^i$ where $i = 1,2,\ldots,n$ we introduce a symbol i into the constructed goal G. We introduce vertices v and $\neg v$ into the network for every variable v from the set of variables. A set of symbols $S(x) = \{i \mid x \in \bigcup_{j=1}^{m_i} \{x_j^i\}\}$ is assigned to each vertex x of the network (set of symbols for a vertex corresponds to the set of clauses in which the literal corresponding to the vertex occurs). Finally we add an edge $\{x_j^i, x_l^k\}$ into the mutex network if $x_j^i = v$ and $x_l^k = \neg v$ or $x_j^i = \neg v$ and $x_l^k = v$ for some Boolean variable v (x_j^i and x_l^k are positive and negative literals of the same variable). Now it is sufficient to observe that we can obtain a solution of the original Boolean formula satisfaction problem from the solution of the constructed problem of goal satisfaction in polynomial time. ■

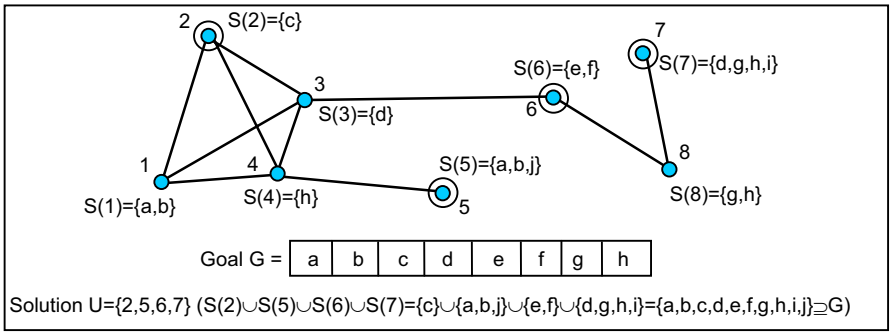


Fig. 2. An instance of the problem of satisfying goal in a mutual exclusion network. The solution is depicted by circles around vertices.

There is little hope to solve the problem of goal satisfaction in mutex network effectively (in polynomial time) in the light of this result. It seems that search is the only option to solve the problem. However, the search may be more or less informed. The more informed search leads to the lower number to steps required for obtaining a solution (the number of steps of the search is usually in tight relation with the overall solving time). One of the most successful techniques how the search can be made more informed is the usage of so called *filtration techniques* (or consistency techniques) which are used intensively in constraint programming [4].

The filtration technique is a specialized algorithm that enforces certain necessary condition for existence of the solution in the problem. Since the basic requirement on

the filtration technique is its high speed and low space requirements the necessary conditions that are used in practice represent relatively big relaxations of the original problem. Enforcing of the consistency is done in most cases by ruling out the values from the variables over which the search makes decisions. Such removal of values reduces the size of the search space. The amount of search space reduction is determined by the strength of the filtration technique (that is by the strength of the enforced necessary condition). On the other hand the stronger filtration technique is often redeemed by its higher time complexity. Therefore a balanced trade-off between strength of filtration technique and resource requirements must be found.

A well known example of filtration technique is *arc-consistency* [11]. This is the representative of the technique that enforces certain type of local necessary condition. Locality of filtering technique means that a small number of decision variables is considered at once (in the case of arc-consistency only two variables are considered at once). Another advantage of local filtering techniques is that they are usually highly generic which allows using them in variety of problems with no or little adaptation.

The stronger filtration can be achieved by so called *global filtering techniques*. These methods take into account more than two decision variables at once (in the extreme case all the decision variables in the problem). The large portion of the problem considered at once inherently implies stronger necessary conditions that can be enforced. However, the drawback of global filtering techniques is that they are associated with particular sub-problems (for example the problem where we have several variables with finite domains of values and we require pair-wise different values to be assigned to these variables respecting variable's domains - *allDifferent* filtering technique [12]) which precludes their usage when we cannot recognize the right sub-problem in the problem of our choice.

4 Global Filtration for Goal Satisfaction in Mutex Network

We have the formal definition of the problem we are about to solve it at this point. The solving approach we develop in this section is a new global filtration technique. The technique will be designed specially for problems of goal satisfaction in mutex network (with regard on applications in concurrent planning).

We visually observed that mutex networks obtained from problems arising in concurrent planning embody high density of edges grouped in relatively small number of clusters (this observation was done using our visualization utility on the series of concurrent planning problems). Let us note that our method works with sparse mutex networks as well. The high density of edges and their structural distribution is caused by various factors. Nevertheless, we regard the functional character of properties of objects encoded in the network as the most important one (in planning generally, the functional character of object's properties is typical). Values that form the domain of such property induces complete sub-graphs (clique) in the mutex network. The described structural characterization of mutex networks we can meet in concurrent planning can be exploited for designing of a filtration technique.

If we know a clique decomposition of the mutex network we can reason about the impact of the vertex selection on possibility of goal satisfaction. To be more concrete,

we know that at most one vertex from each clique of the decomposition can be selected to contribute to the satisfaction of the goal. Hence, for each clique of the decomposition we can calculate the maximum number of symbols of the goal which can be covered by the vertices of the clique. When we select a vertex into the solution the necessary condition on the solution existence is that the number of symbols covered by the remaining cliques of the decomposition together with symbols associated with selected vertex must not be lower than the number of symbols in the goal.

The second part of the idea of our filtration technique is that if we restrict ourselves on the proper subset of the goal the set of vertices ruled out by the above counting arguments can be different. Therefore it is possible to perform filtration by the technique with respect to multiple sub-goals of the goal to achieve the maximum pruning power. We call these sub-goals of the goal *projection goals* and according to this designation we call the whole filtration technique *projection consistency*.

4.1 Partitioning the Mutex Network into Cliques

Projection consistency assumes that a partition into cliques of a mutex network is known. Thus we need to perform a preprocessing step in which a partition into cliques of the mutex network is constructed. Let $N = (V, M)$ be a mutex network. The task is to find a partition of the set of vertices $V = C_1 \cup C_2 \cup \ldots \cup C_n$ 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 M for $i = \{1,2,\ldots,n\}$. Cliques of the partitioning do not cover all the edges in the network in general case. For $m = M - (C_1^2 \cup C_2^2 \cup \ldots \cup C_n^2)$, $m \neq \emptyset$ holds in general (where $C^2 = \{\{a,b\} \mid a,b \in C \,\&\, a \neq b\}$). Our requirement is to minimize n and $|m|$ somehow. Unfortunately this problem is too hard for reasonable objective functions of n and $|m|$ to be solved within the preprocessing step (for instance it is *NP*-complete for minimizing just n [5]).

As an exponential amount of time spent in preprocessing step is unacceptable it is necessary to abandon the requirement on optimality of partition into cliques. It is sufficient to find some partition into cliques to be able to introduce projection consistency. Our experiments showed that a simple greedy algorithm provides satisfactory results. Its complexity is polynomial in size of the input graph which is acceptable for the preprocessing step. The greedy algorithm we are using repeatedly finds the largest greedy clique; the clique is extracted from the network in each step; the algorithm continues until the network is non-empty. For detailed description of this process see [14]. We also made some experiments with partition into cliques of higher qualities than that produced by the greedy algorithm. However, we did not observe any subsequent improvement of the filtering strength of projection consistency.

4.2 Formal Definition of Projection Consistency

For the following formal description of projection consistency we assume that a partition into cliques $V = C_1 \cup C_2 \cup \ldots \cup C_n$ of the mutex network $N = (V, M)$ was constructed. Projection consistency is defined over the above clique decomposition for a projection goal $\emptyset \neq P \subseteq G$. The projection goal P enters the definitions as a parame-

ter. Projection goals are used for restricting the consistency on a certain part of the goal satisfaction problem (on certain part of the goal) which may eventually strengthen the necessary condition we are about to check.

The fact that at most one vertex from a clique can be selected into the solution allows us to introduce the following definition.

Definition 3 (Clique contribution). A *contribution of a clique* $C \in \{C_1, C_2, \ldots, C_n\}$ to the projection goal $\emptyset \neq P \subseteq G$ is defined as $\max(|S(v) \cap P| \mid v \in C)$ and it is denoted as $c(C,P)$. □

The concept of clique contribution is helpful when we are trying to decide whether it is possible to satisfy the projection goal by selecting the vertices from the partition into cliques. If for instance $\sum_{i=1}^{n} c(C_i,P) < |P|$ holds then the projection goal P cannot be satisfied. Nevertheless, the projection consistency can handle a more general case as it is described in the following definitions.

Definition 4 (Projection consistency: supported vertex). A vertex $v \in C_i$ for $i \in \{1,2,\ldots,n\}$ is *supported* with respect to a given clique decomposition and the projection goal P if $\sum_{j=1, j\neq i}^{n} c(C_j,P) \geq |P - S(v)|$ holds. □

Definition 5 (Projection consistency: consistent problem). An instance of the problem of satisfaction of a goal G in a mutex network $N = (V,M)$ is consistent with respect to the given clique decomposition and the projection goal $\emptyset \neq P \subseteq G$ if every vertex $v \in C_i$ for $i = 1,2,\ldots,n$ is supported with respect to the given clique decomposition and the given projection goal. □

It is easy to observe that projection consistency is a necessary but not sufficient condition on existence of the solution. This claim is formally proved in [15].

Algorithm 1: Projection consistency propagation algorithm

function *propagateProjectionConsistency* $(\{C_1, C_2, \ldots, C_n\}, P)$: **set**
1: $\gamma \leftarrow 0$
2: **for** $i = 1,2,\ldots,n$ **do**
3: | $c_i \leftarrow$ *calculateCliqueContribution* (C_i, P)
4: | $\gamma \leftarrow \gamma + c_i$
5: **for** $i = 1,2,\ldots,n$ **do**
6: | **for each** $v \in C_i$ **do**
7: | | **if** $\gamma + |S(v) \cap P| < |P - S(v)| + c_i$ **then** $C_i \leftarrow C_i - \{v\}$
8: **return** $\{C_1, C_2, \ldots, C_n\}$

function *calculateCliqueContribution* (C,P) : **integer**
9: $c \leftarrow 0$
10: **for each** $v \in C$ **do**
11: | $c \leftarrow \max(c, |S(v) \cap P|)$
12: **return** c

A propagation algorithm for projection consistency is shown here as algorithm 1. Clique decomposition and projection goal are parameters of the algorithm. The algorithm runs in $O(|V||P|)$ steps which is polynomial in size of the input [15].

To ensure maximum vertex filtration effect we can enforce the consistency with respect to multiple projection goals. However, it is not possible to use all the projection goals since they are too many. Our experiments showed that projection goals P_i where $P_i = \{s \mid s \in G \,\&\, |\{v \mid s \in S(v) \cap G\}| = i\}$ provide satisfactory filtration effect (precisely, it is the best selection rule we found by experimentation). The number of projection goals of this form is linear is size of the goal G.

An example of projection consistency enforcing in the goal satisfaction problem from figure 2 is shown in figure 3.

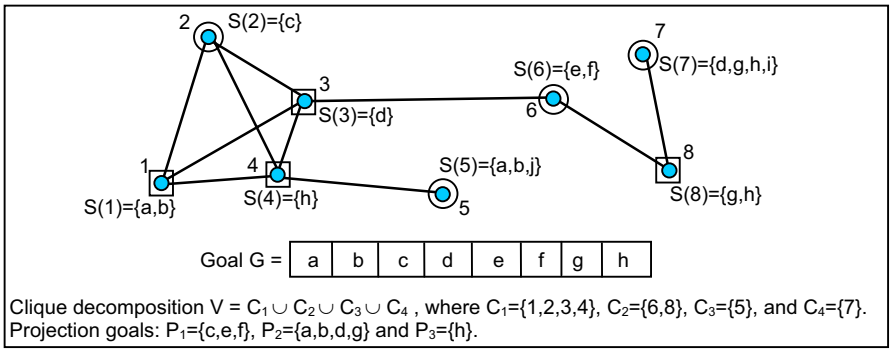


Fig. 3. Example of projection consistency enforcing. The goal satisfaction problem is same as in figure 2. Unsupported vertices are surrounded by squares. For example vertex 3 is unsupported for the projection goal P_1={c,e,f} since vertex 3 contributes by 0, C_2 contributes by 2, C_3 contributes by 0, and C_4 contributes by 0 which is together less than the size of P_1.

5 Experimental Evaluation

This section is devoted to experimental evaluation of the projection consistency. Our experimental evaluation is concentrated on two aspects of the proposed global consistency. Firstly, we would like to evaluate the consistency itself by using a set of randomly generated goal satisfaction problems. Secondly, we would like to evaluate the benefit of the new consistency when it is applied in concurrent planning. We carried out this evaluation by integrating the consistency into the GraphPlan based algorithm for generating concurrent solutions of planning problems.

5.1 Random Goal Satisfaction Problems

When we visually observed how do the problems arising in concurrent planning look like the distribution of structures was clearly evident. The mutex network associated with the problems typically consists of small number of relatively large cliques accompanied with small number of edges not belonging to any clique. The example of such mutex network is shown in figure 4.

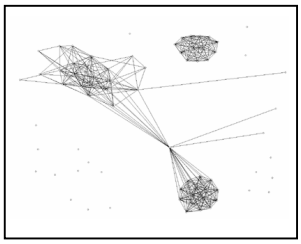


Fig. 4. Mutex network arising as a sub-problem during concurrent solution construction of a planning problem.

The most variable part of the problem as it was evidenced by our observation is the number of edges not belonging to any clique. Therefore we decided to have this parameter as the main variable parameter in our set of randomly generated problems.

As a competitive technique we chose arc-consistency since it is similar to our new technique in several aspects. First, arc-consistency is easy to implement. This is also true for projection consistency. Second, both filtration techniques remove values from the decision variables (not tuples of values etc.).

We integrated both techniques into a backtracking based algorithm for solving the goal satisfaction problem in mutex network. The algorithm performs filtration after each decision - so we are maintaining arc-consistency [11] or projection consistency respectively in our solving approach.

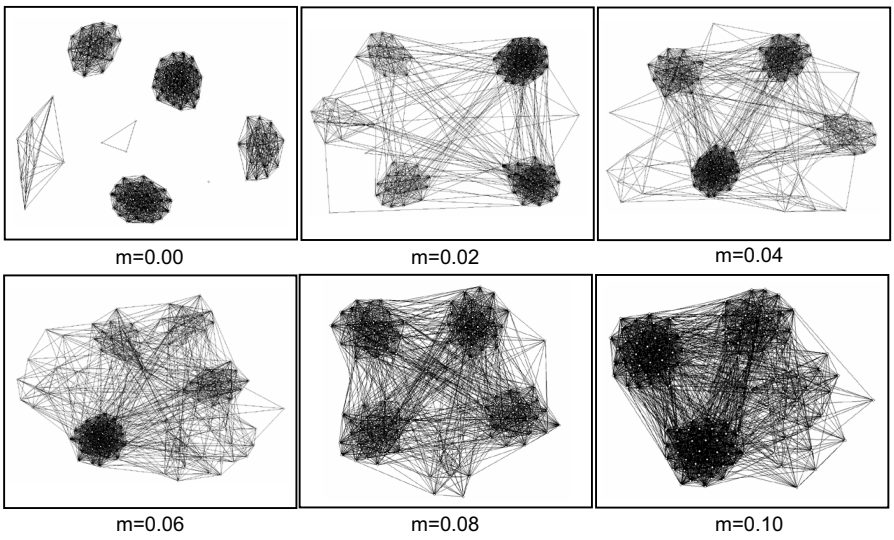


Fig. 5. Random mutex networks with 120 vertices and with fixed structured part (several complete sub-graphs) and with increasing portion of randomly added edges. The parameter *m* is the probability of presence of an edge between a pair of vertices. Mutex networks shown in this figure were used for experimental evaluation.

We evaluated our global filtration technique in comparison with arc-consistency on a set of random problems of goal satisfaction of the following setup motivated by the visual observation of problems arising in concurrent planning. In a mutex network consisting of 120 vertices we constructed several complete sub-graphs using uniform distribution with the mean value of 20.0. The size of the goal was 60 and each vertex

has assigned a random set of symbols from the goal of the size generated by the normal distribution with the mean value of 8.0 and the standard deviation of 6.0. Finally we added random edges into the mutex network. More precisely, we add each possible edge into the mutex network with the probability of m where m was a variable parameter which ranged from 0.0 to 0.1. The illustration of randomly generated mutex networks used in our evaluation is shown in figure 5.

For each value of the parameter m we generated 10 goal satisfaction problems and we solved them using backtracking with maintaining arc-consistency and maintaining projection consistency respectively. Along the solving process we collected data such as number of backtracks, runtime etc. The variable and value ordering heuristics are the following. A variable with the smallest domain (smallest clique) is selected preferably. Values (vertices) within the variable's domain are not ordered.

The tested algorithms were implemented in C++ and were run on a machine with AMD Opteron 242 processor (1.6 GHz) and 1 GB of memory under Mandriva Linux 10.2. The code was compiled by gcc compiler version 3.4.3.

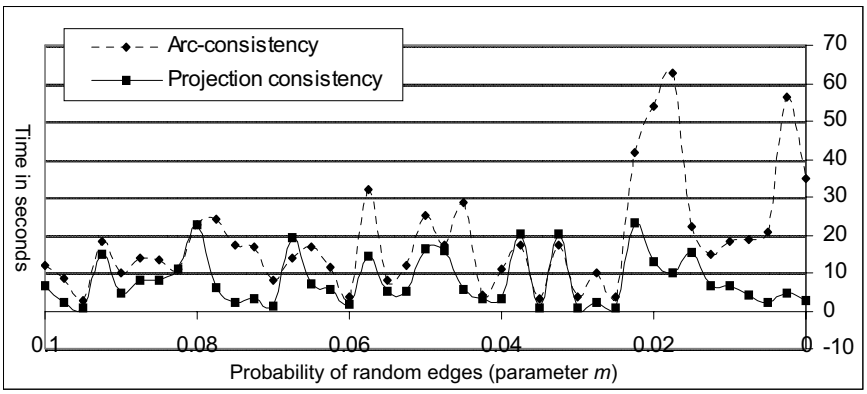


Fig. 6. Runtime for random goal satisfaction problems (average of 10 problems for each value of random edge probability m).

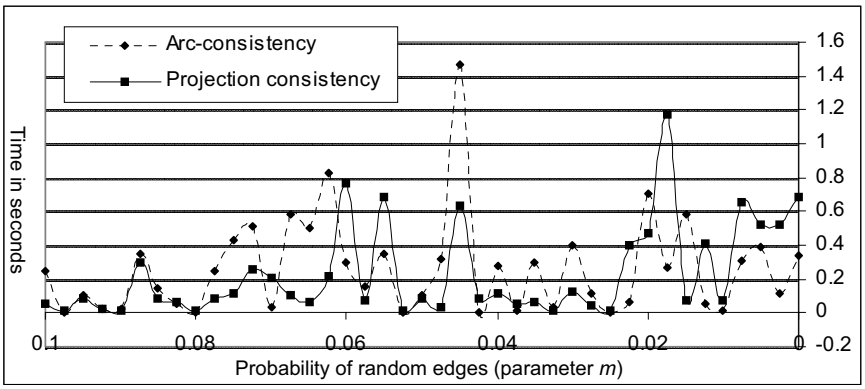


Fig. 7. Runtime for random goal satisfaction problems (easiest problem of 10 for each value of random edge probability m).

For each value of parameter m we calculated average runtime of both techniques, runtime of the easiest problem (the problem with the fewest number of backtracks) and the runtime of the hardest problem (the problem with the highest number of backtracks). The results we obtained are shown in figures 6, 7, and 8.

The results show that backtracking with maintaining projection consistency is generally faster than backtracking with maintaining arc-consistency on a set of tested problems. In some cases, version with maintaining projection consistency is several times faster (figure 6). The version with maintaining projection consistency achieves better improvement compared to the version with maintaining arc-consistency on harder problems (figure 8). On the other hand, on easy problems projection consistency provides little or no advantage (figure 7). We may also observe that harder problems tends to occur more for lower values of m. On these problems projection consistency represents clearly the better option.

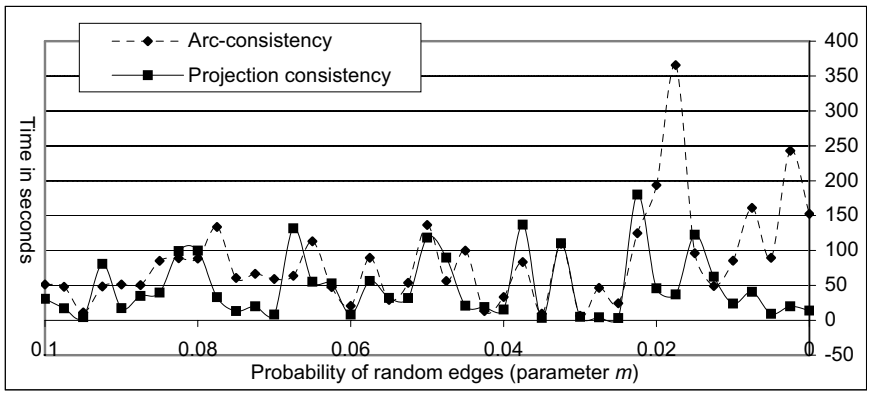


Fig. 8. Runtime for random goal satisfaction problems (hardest problem of 10 for each value of random edge probability m).

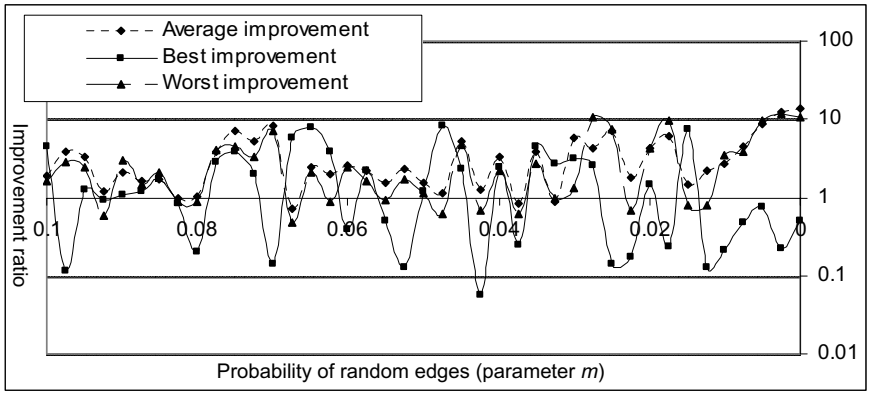


Fig. 9. Improvement ratio of backtracking with maintaining projection consistency with respect to backtracking with maintaining arc-consistency on random goal satisfaction problems.

The improvement ratio of solving algorithm using projection consistency with respect to the version with maintaining arc-consistency is shown in figure 9. On the tested problems we reached the improvement up to the order of magnitude.

The solvability ratio of the tested problems is shown in figure 10. For different values of the parameter m we had a different numbers of problems that had a solution and problems for which the solution does not exist.

We also performed comparison of number of backtracks that occurred during solving the random problems by the tested algorithms. In addition to backtracking with maintaining arc-consistency and projection consistency we also made the calculation of backtracks made by simple uninformed backtracking. The comparison of number of backtracks is shown in figure 11. According to figure 6 and figure 11 we can observe that the runtime and the number of backtracks correspond well.

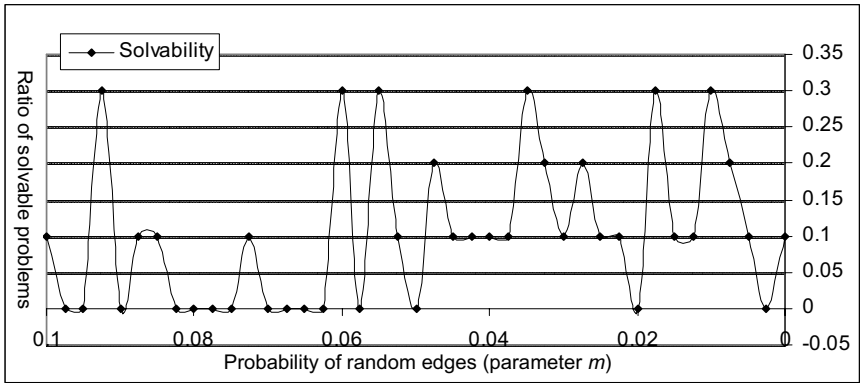


Fig. 10. Ratio of solvable random goal satisfaction problems. For each value of parameter m the number of solvable problems divided by the total number of problems (=10) is shown.

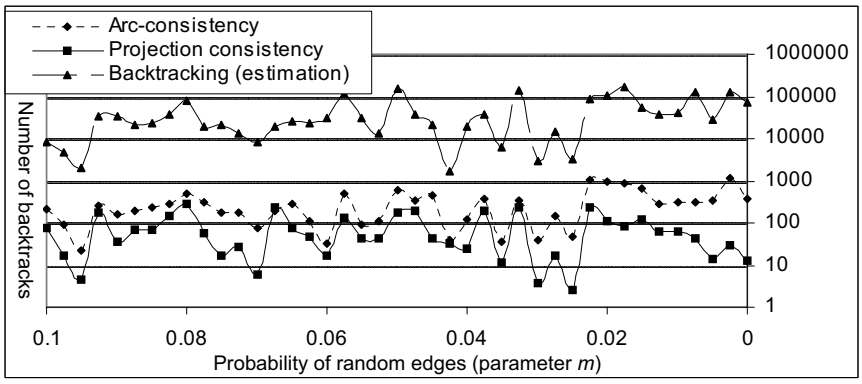


Fig. 11. Number of backtracks of tested algorithms on random goal satisfaction problems.

5.2 Problems Arising in Concurrent Planning

We also evaluated the proposed projection consistency in solving problems that arise in concurrent planning (that is in the area for which the filtering technique was designed). We used GraphPlan planning algorithm [2] for this evaluation. This algorithm often solves a sub-problem that can be reformulated to a goal satisfaction problem in mutex network.

In our evaluation we used maintaining arc-consistency and projection consistency respectively to improve solving process of this sub-problem within the planning algorithm. We used a set of planning problems of three domains - dock worker robots domain, towers of Hanoi domain, and refueling planes domain. The tested problems were of various difficulties. The length of solution plans ranged from 4 to 44 actions. The comparison of runtime of standard GraphPlan and versions enhanced with maintaining arc-consistency and projection consistency is shown in figure 12. All the planning problems used in this evaluation are available at the web site: http://ktiml.mff.cuni.cz/~surynek/research/rac2007/ (we use our own format of planning problems since we use non-standard representation with explicit state variables).

The improvement obtained by using projection consistency is up to 1000% with respect to both the standard GraphPlan as well as with respect to the version enhanced by arc-consistency. Additionally, we found that goal satisfaction problems arising in these planning problems are very similar to random goal satisfaction problems with the parameter ranging from 0.07 to 0.02 where the improvement obtained by projection consistency is promising.

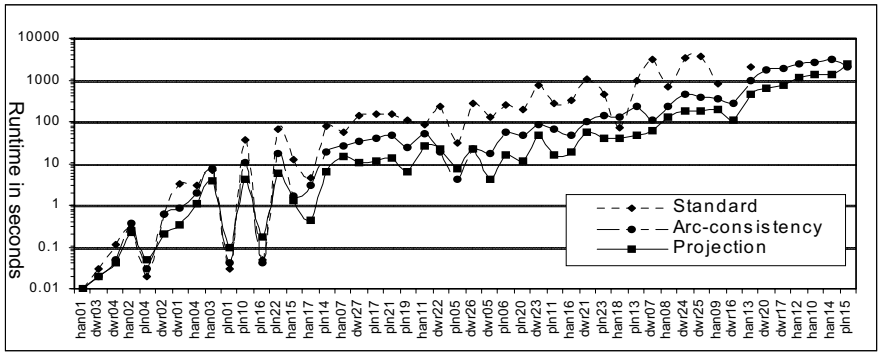


Fig. 12. Runtime comparison of GraphPlan based planning algorithm with versions of this algorithm enhanced by maintaining arc-consistency and projection consistency for solving goal satisfaction problems on a set of planning problems of various difficulties.

6 Note on Additional Related Works and Conclusion

Originally, we proposed projection consistency in [15]. This paper is dedicated to theoretical properties of the technique (theoretical comparison with arc-consistency is given in the paper). A study of using similar technique to projection consistency in

solving difficult Boolean formula satisfaction problems is given in [17]. The application of arc-consistency in planning using planning graphs is proposed in [16]. We also investigated the possible strengthening of the projection consistency by replacing the expression $\sum_{j=1, j\neq i}^{n} c(C_j, P) \geq |P - S(v)|$ in the definition 4 by $\sum_{j=1, j\neq i}^{n} c(C_j, P - S(v)) \geq |P - S(v)|$. Details are given in [13]; but let us note that this change complicates effect of vertex removal too much in a general network (the so called *monotonicity* [15] does not hold).

The ideas of using constraint programming techniques in concurrent planning are presented in [6, 7]. However, only local propagation techniques are studied there (contrary to our approach which is global). Concurrent planning itself is studied in [20]. In this work we were primarily inspired by global constraints such as that presented in [12].

We proposed a novel global filtration technique for a problem of goal satisfaction in mutual exclusion networks. The problem for which the filtration was designed was inspired by concurrent planning. However, the technique is more general, currently we know that it is also effectively applicable in solving SAT problems.

We evaluated our technique in comparison with arc-consistency on a set of randomly generated problems. For this evaluation we used our own implementation in C++. The improvement gained by using projection consistency is up to the 10 times shorter runtime; a similar improvement was achieved in number of backtracks. Finally, we integrated our technique into the GraphPlan planning algorithm to evaluate it in some area of application. Again we obtained significant improvements compared to the standard version.

For future work we plan to investigate more precise computation of supported vertices (definition 4) using network flows. We believe that a more precise computation of this would lead to a stronger necessary condition we are checking.

Acknowledgements

This work is supported by the Czech Science Foundation under the contracts number 201/07/0205 and 201/05/H014. I would like to thank anonymous reviewers for many rigorous comments.

References

1. Allen, J., Hendler, J., Tate, A.: Readings in Planning. Morgan Kaufmann Publishers, 1990.
2. Blum, A. L., Furst, M. L.: Fast Planning through planning graph analysis. Artificial Intelligence 90, 281-300, AAAI Press, 1997.
3. Cook, S. A.: The Complexity of Theorem Proving Procedures. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 151-158, USA, ACM Press, 1971.
4. Dechter, R.: Constraint Processing. Morgan Kaufmann Publishers, 2003.
5. Golumbic, M. C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, 1980.
6. Kambhampati, S.: Planning Graph as a (Dynamic) CSP: Exploiting EBL, DDB and other CSP Search Techniques in Graphplan. JAIR 12, 1-34, AAAI Press, 2000.

7. Kambhampati, S., Parker, E., Lambrecht, E.: Understanding and Extending GraphPlan. In Proceedings of 4th European Conference on Planning, 260-272, LNCS 1348, Springer-Verlag, 1997.
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. Little, I., Thiébaux, S.: Concurrent Probabilistic Planning in the Graphplan Framework. In Proceedings of the 16th International Conference on Automated Planning and Scheduling, Cumbria, UK, 263-272, AAAI Press, 2006.
10. Long, D., Fox, M.: Efficient Implementation of the Plan Graph in STAN. JAIR 10, 87-115, AAAI Press, 1999.
11. Mackworth, A. K.: Consistency in Networks of Relations. Artificial Intelligence 8, 99-118, AAAI Press, 1977.
12. Régin, J. C.: A Filtering Algorithm for Constraints of Difference. In Proceedings of the 12th National Conference on Artificial Intelligence, 362-367, AAAI Press, 1994.
13. Surynek, P.: Tractable Class of a Problem of Goal Satisfaction in Mutual Exclusion Network. Proceedings of the 21st International FLAIRS Conference, Miami, FL, USA, to appear, AAAI Press, 2008.
14. Surynek, P.: Projection Global Consistency: An Application in AI Planning. Technical Report, ITI Series, 2007-333, http://iti.mff.cuni.cz/series, Charles University, Prague, 2007.
15. Surynek, P.: Projection Global Consistency: An Application in AI Planning. Proceedings of CSCLP Workshop 2007, 61-75, France, INRIA, 2007.
16. Surynek, P., Barták, R.: Maintaining Arc-consistency over Mutex Relations in Planning Graphs during Search. Proceedings of the 20th International FLAIRS Conference, Key West, FL, USA, 134-139, AAAI Press, 2007.
17. Surynek, P.: Solving Difficult SAT Instances Using Greedy Clique Decomposition. Proceedings of the 7th SARA Symposium, Whistler, Canada. 359-374, LNCS 4612, Springer Verlag, 2007.
18. Surynek, P., Chrpa, L., Vyskočil, J.: Solving Difficult Problems by Viewing Them as Structured Dense Graphs. Proceedings of the 3rd IICAI Conference, Pune, India, 2007.
19. Urquhart, A.: Hard Examples for Resolution. Journal of the ACM, 34/209-219, ACM Press, 1987.
20. Zimmerman, T., Kambhampati, S., Using Memory to Transform Search on the Planning Graph. JAIR 23, 533-585, AAAI Press, 2005.