Long-Distance Mutual Exclusion for Propositional Planning

Yixin Chen, Zhao Xing, and Weixiong Zhang
Washington University in St.Louis
Department of Computer Science and Engineering
{chen,zx2,zhang}@cse.wustl.edu

Abstract

The use of mutual exclusion (mutex) has led to significant advances in propositional planning. However, previous mutex can only detect pairs of actions or facts that cannot be arranged at the same time step. In this paper, we introduce a new class of constraints that significantly generalizes mutex and can be efficiently computed. The proposed **long-distance mutual exclusion (londex)** can capture constraints over actions and facts not only at the same time step but also across multiple steps. Londex provides a powerful and general approach for improving planning efficiency. As an application, we have integrated londex into SATPLAN04, a leading optimal planner. Experimental results show that londex can effectively prune the search space and reduce the planning time. The resulting planner, MaxPlan, has won the First Place Award in the Optimal Track of the 5th International Planning Competition.

1 Introduction

Propositional planning entails arranging a course of actions in order to achieve certain goals subject to logical conditions and effects of actions. Propositional planning has traditionally been formalized as deduction [Green, 1969]and is known to be PSPACE-complete [Bylander, 1994]. Previous work on this deduction paradigm has achieved limited success and can only handle small cases.

Emerging as one of the most efficient techniques for propositional planning, mutual exclusion (mutex) was introduced in Graphplan [Blum and Furst, 1997] to construct constraints for actions and facts. In Graphplan, two actions at the same level are persistent mutex actions if one action deletes a precondition or an add-effect of the other, or the two actions have mutual exclusive preconditions. During the search, mutex constraints are enforced to prune the search space.

Since its invention, mutex has been extensively studied and led to the development of many efficient planners. For example, Gerevini and Schubert proposed new state constraints from forward propagation of mutex [Gerevini and Schubert, 1998], Smith and Weld [Penberthy and Weld, 1994] and Gerevini, et al. [Gerevini *et al.*, 2003] extended mutex to temporal planning, and Kautz and Selman encoded mutex constraints in the BLACKBOX [Kautz and Selman, 1996] and SATPLAN04 [Kautz, 2004] planners. Note that almost all existing fast planners for different applications are based on mutex. For example, the LPG temporal planner by Gerevini, et al. [Gerevini *et al.*, 2003] won the First Prize of the Suboptimal Track of the 3rd International Planning Competition (IPC), and SATPLAN04 won the First Prize of the Optimal Track of the 4th and 5th IPCs.

Despite its success, mutex has a major limitation that it can only detect conflicts between a pair of actions or facts in the same time step, but it is unable to capture constraints relating actions and facts in different steps. On the other hand, long distance constraints are ubiquitous in almost all planning domains. For example, in a transportation problem, two actions requiring to use the same truck but at different locations cannot be arranged too close to each other in time since the truck has to be relocated after the first action.

In this research, we have developed **long-distance mutual exclusion (londex)**, a novel class of constraints for propositional planning. Unlike mutex that is based on a Boolean fact representation, londex is derived from domain transition graphs (DTGs) on a multi-valued domain formulation (MDF) [Jonsson and Backstrom, 1998] of a planning problem. Table 1 compares the key features of londex and mutex. As a superset of mutex, londex can capture constraints relating actions and facts not only in the same time step, but also across multiple steps. An important utility of londex is for measuring the minimum distance between a pair of actions to prune a search space and improve the planning efficiency. For example, for the truck5 problem from IPC5 [IPC, 2006], SATPLAN04 generates 199,219 mutex constraints, whereas our method using londex produces 12,421,196 londex constraints which result in a 4.3 speedup in time when incorporated in SATPLAN04.

The paper is organized as follows. In Section 2, we define the basic concepts and review the conventional mutex. We then discuss the MDF representation and introduce londex in Section 3. We incorporate londex in the SATPLAN04 planner to develop our new planner, and evaluate its performance on IPC5 benchmark problems in Section 4. We discuss related work in Section 5 and conclude in Section 6.

Property	Mutex	Londex
representation	Boolean facts	MDF
generation time	polynomial	polynomial
relevant level(s)	same level	multiple levels
fact constraints	from actions	from DTG analysis
action constraints	from add/del/pre facts	from fact distances
quantity	moderate	large
pruning	moderate	strong
relationship	persistent mutex $\subset$ londex	

Table 1: Comparison of persistent mutex and londex.

2 Propositional Planning and Mutex

Propositional planning requires to generate a sequence of actions in order to accomplish a set of goals. Propositional planning can be described in the STRIPS formalism, which is represented by a set of Boolean facts. Given a set of facts $F = \{f_1, f_2, \cdots, f_n\}$, a **state** s is a subset of F where every fact in s is True.

Example 1. A simple cargo planning problem concerns moving a piece of cargo C among three locations $L1, L2$ and $L3$ using a truck T. The fact set F includes all the facts about possible locations of the truck and the cargo, such as $at(T, L1)$, $at(C, L2)$, and $in(C, T)$. A state specifies the current status. For example, $s = \{at(T, L1), at(C, L3)\}$.

Definition 1 *An* ***action*** *o is a triplet $o = (pre(o), add(o), del(o))$, where $pre(o) \subseteq F$ is the set of preconditions of action o, and $add(o) \subseteq F$ and $del(o) \subseteq F$ are the sets of add facts and delete facts, respectively.*

The result of applying an action o to a state s is defined as:

$$Result(s, o) = \begin{cases} s \cup add(o) \setminus del(o), & \text{if } pre(o) \subseteq s; \\ undefined, & \text{otherwise.} \end{cases}$$

Consequently, applying a sequence of actions $P = \langle o_1, o_2, \cdots, o_n \rangle$ results in $Result(s, P) = Result(\cdots(Result(Result(s, o_1), o_2)) \cdots o_n)$. Alternatively, a set of actions can be performed in a parallel plan with a number of *time steps*, where multiple actions can be executed in a time step.

Definition 2 *A* ***planning task*** *is a triplet $T = (O, I, G)$, where O is a set of actions, $I \subseteq F$ is the initial state, and $G \subseteq F$ is the goal specification.*

Definition 3 Persistent mutex of actions. *Actions o_1 and o_2 are mutex when any of the following sets is not empty: $pre(o_1) \cap del(o_2)$, $pre(o_2) \cap del(o_1)$, $del(o_1) \cap add(o_2)$, and $del(o_2) \cap add(o_1)$, or when there exist two facts $f_1 \in pre(o_1)$ and $f_2 \in pre(o_2)$ that are mutex.*

Definition 4 Persistent mutex of facts. *Two facts f_1 and f_2 are mutex if there do not exist two non-mutex actions o_1 and o_2 such that $f_1 \in add(o_1)$ and $f_2 \in add(o_2)$.*

The mutex constraints defined above [Blum and Furst, 1997; Gerevini *et al.*, 2003] are *persistent* in the sense that they are always effective, regardless of the initial state or the level of a planning graph. Blum and Furst [Blum and Furst, 1997] also suggested to derive non-persistent mutex dependent of initial states and levels. However, deriving non-persistent mutex usually requires to construct a planning graph, which is expensive and impractical for large problems. As a result, many fast planners only use persistent mutex constraints. In the rest of the paper, when we mention mutex, we refer to persistent mutex.

3 Long-Distance Mutual Exclusion (Londex)

To develop londex, we first consider the multi-valued formulation of propositional planning.

3.1 Domain Transition Graph

The *multi-valued domain formulation* (MDF) [Jonsson and Backstrom, 1998] translates Boolean facts into a more compact representation using multi-valued variables, where each variable represents a group of Boolean facts in which only one of them can be true in any state.

The MDF representation of a propositional planning domain is defined over a set of multi-valued variables: $X = (x_1, x_2, \cdots, x_m)$, where x_i has a finite discrete domain $\mathcal{D}_i$.

For a planning problem, the value assignment of a multi-valued variable in MDF corresponds to a Boolean fact. Given an MDF variable assignment $x = v$ and a Boolean fact f, we can denote this correspondence as $f = \text{MDF}(x, v)$.

An ***MDF state*** s is encoded as a complete assignment of all the variables in X: $s = (x_1 = v_1, x_2 = v_2, \cdots, x_m = v_m)$, where x_i is assigned a value $v_i \in \mathcal{D}_i$, for $i = 1, 2, \ldots, m$.

Example 3. To formulate Example 1 in MDF, we use two MDF variables, $Location_T$ and $Location_C$,to denote the locations of the truck and the cargo, respectively. $Location_T$ can take values from $\mathcal{D}_T = \{L1, L2, L3\}$ and $Location_C$ from $\mathcal{D}_C = \{L1, L2, L3, T\}$. An example state is $s = (Location_T = L1, Location_C = T)$.

For every multi-valued variable in MDF, there is a corresponding domain transition graph defined as follows.

Definition 5 (Domain transition graph (DTG)). *Given an MDF variable $x \in X$ with domain $\mathcal{D}_x$, its DTG G_x is a digraph with vertex set $\mathcal{D}_x$ and arc set $\mathcal{A}_x$. There is an arc $(v, v') \in \mathcal{A}_x$ iff there is an action o with $del(o) = v$ and $add(o) = v'$, in which case the arc (v, v') is labeled by o and we say there is a transition from v to v' by action o.*

Definition 6 (Fact distance in a DTG). *Given a DTG G_x, the distance from a fact $f_1 = MDF(x, v_1)$ to another factor $f_2 = MDF(x, v_2)$, written as $\Delta_{G_x}(f_1, f_2)$, is defined as the minimum distance from vertex v_1 to vertex v_2 in G_x.*

Example 4. In the cargo example, the cargo can be loaded into the truck or unloaded at a location, and the truck can move among different locations. The DTGs of the cargo and truck are shown in Figure 1. In the figure, $\Delta_{G_{Location_C}}(\text{MDF}(C, L1), \text{MDF}(C, L2)) = 2$, and $\Delta_{G_{Location_T}}(\text{MDF}(T, L3), \text{MDF}(T, L1)) = \infty$.

Following a method used in the Fast Downward planner [Helmert, 2006], we have implemented our own preprocessing engine to generate DTGs. The main steps of the preprocessing include detecting invariant fact groups, finding MDF variables, and generating the value transitions for each MDF variable. Details of the algorithm can be found in Helmert's paper [Helmert, 2006].

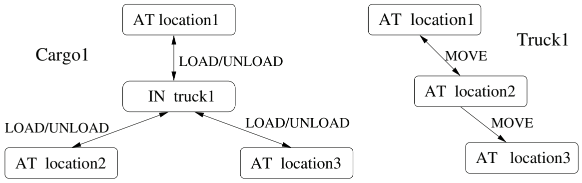

Figure 1: DTGs for MDF variables $Location_C$ and $Location_T$ in Example 4.

3.2 Generation of Londex Constraints

We now define londex based on the MDF representation. We define $t(f)$ as the time step at which the fact f is valid, and $t(o)$ as the time step at which an action o is chosen.

Londex constraints for facts. Given two Boolean facts f_1 and f_2, if f_1 and f_2 correspond to two nodes in a DTG G_x and $\Delta_{G_X}(f_1, f_2) = r$, then there is no valid plan in which $0 \leq t(f_2) - t(f_1) < r$.

Example 5. Continuing from Example 4, if $at(C, L1)$ is true at time step 0, then $at(C, L2)$ cannot be true before step 2.

Londex constraints for actions. For simplicity, we say that an action o is *associated* with a fact f if f appears in $pre(o), add(o)$, or $del(o)$. Given two actions a and b, we consider two classes of londex constraints between them.

Class A. If a and b are associated with a fact f, a and b are mutually exclusive if one of the following holds.

1) $f \in add(a),\ f \in del(b)$, and $t(a) = t(b)$;
2) $f \in del(a),\ f \in pre(b)$, and $0 \leq t(b) - t(a) \leq 1$;

Note that A(1) and A(2) are stronger than the original mutex due to the inequalities in A(2). If we replace the inequalities in A(2) by $t(a) = t(b)$, A(1) - A(2) are equivalent to the original mutex.

Class B. If a is associated with fact f_1, b is associated with fact f_2, and it is invalid to have $0 \leq t(f_2) - t(f_1) < r$ according to fact londex, then a and b are mutually exclusive if one of the following holds.

1) $f_1 \in add(a),\ f_2 \in add(b)$, and $0 \leq t(b) - t(a) \leq r - 1$;
2) $f_1 \in add(a),\ f_2 \in pre(b)$, and $0 \leq t(b) - t(a) \leq r$;
3) $f_1 \in pre(a),\ f_2 \in add(b)$, and $0 \leq t(b) - t(a) \leq r - 2$;
4) $f_1 \in pre(a),\ f_2 \in pre(b)$, and $0 \leq t(b) - t(a) \leq r - 1$.

Intuitively, since two facts f_1 and f_2 in the same DTG cannot appear too closely due to the structure of the DTG, two actions associated with these facts should also keep a minimum distance. For example, in case B(2), if a is executed at time $t(a)$, then f_1 is valid at time $t(a) + 1$. Since the fact distance from f_1 to f_2 is larger than r, f_2 (the precondition of b) cannot be true until time $t(a) + 1 + r$, which means b cannot be true until time $t(a) + 1 + r$. Other cases can be proved similarly.

The procedure to generate the londex constraints is in Table 2, in which $EA1(f)$ represents all Class-A action londex related to a fact f, and $EA2(f_1, f_2)$ denotes all Class-B action londex related to a pair of facts f_1 and f_2 in a DTG.

Example 6. In the cargo example, mutex can only detect the constraints that the truck cannot arrive at and leave the same location at the same time. For example, $move(T, L1, L2)$ and $move(T, L3, L1)$ are mutually exclusive because $move(T, L1, L2)$ deletes $at(T, L1)$ while $move(T, L3, L1)$ adds $at(T, L1)$. In contrast, londex is stronger as it specifies that two actions using the same truck, even if they happen at *different* time, may lead to a conflict. For example, if $L1$ and $L4$ are 5 steps apart in a DTG, then the conflict of arranging $move(T, L1, L2)$ at step 0 and $move(T, L4, L3)$ at step 3 is a londex but not a mutex.

```
procedure Generate_Londex (P)
(INPUT: A STRIPS planning problem P)
(OUTPUT: londex constraints for facts and actions)
    generate the DTGs for P;
    generate fact londex based on the DTGs;
    for each fact f do
        generate action londex in EA1(f);
    for each DTG graph G_x do
        for each pair of facts (f1, f2), f1, f2 ∈ G_x do
            generate action londex in EA2(f1, f2);
end_procedure
```

Table 2: Generation of londex constraints.

3.3 Analysis of Londex Constraints

In this section, we study the completeness, computation complexity, and pruning power of londex constraints.

Proposition 1 *For any planning problem P, the set of persistent mutex is a subset of londex.*

This is true by the definitions of mutex and londex. The Class A action londex include all persistent action mutex.

Proposition 2 *Londex constraints can be generated in time polynomial in the number of facts.*

Proof. Due to space limits, we only sketch the time results here. We denote the number of facts as $|F|$ and the plan length as T. We use S to denote the upper bound of the number of preconditions, add effects, or delete effects that any action may have in a planning domain. Usually S is a small constant for most planning domains. The time to generate the DTGs is $O(S|F|^2)$. The time to generate Class A londex is $O(TS^2|F|)$, and the time to generate Class B londex is $O(TS^2|F|^2)$. Therefore, the total time complexity for the algorithm is $O(TS^2|F|^2)$, which is polynomial in $|F|$.

Proposition 2 gives the worst case complexity guarantee to generate all londex constraints. This is comparable to the polynomial time complexity to generate mutex constraints [Blum and Furst, 1997]. Empirically, our preprocessor takes less than 30 seconds to generate all londex constraints for each of the problems in IPC5, which is negligible comparing to the planning time that londex can help reduce, often in thousands of seconds for large problems.

Table 3 illustrates the use of londex in reducing planning time. We can compare the size of mutex constraints derived by Blackbox and SATPLAN04 (two SAT-based planners using mutex) and the londex constraints. The amount of londex constraints is 2 to 100 times larger than that of mutex, depending on planning domains. It is evident from Table 3 that incorporating londex in a SAT planner, although largely increases the size of the SAT instance, can significantly reduce the speed of SAT solving because of the much stronger constraint propagation and pruning.

method	size	U. P.	decisions	time
SATPLAN04	2.90e5	3.51e7	83398	109.5
Blackbox	2.87e5	4.06e7	496254	267.8
SATPLAN04+londex	1.96e7	3.13e7	22015	34.5

Table 3: Comparison of the number of constraints, unit propagations (UP), decisions, and solution time of SATPLAN04, Blackbox, and SATPLAN04 with londex, for solving the truck3 problem from IPC5 at time step 15.

```
procedure SATPLAN04+londex
 (INPUT: A propositional planning problem P)
   generate the DTGs and londex constraints for P;
   set k to be an lower bound of the optimal plan length;
   repeat
       convert P to a planning graph with k time steps;
       translate the planning graph into a SAT instance;
       add the londex constraints to the SAT;
       solve the SAT instance by a generic SAT solver;
       if (the SAT instance is satisfiable)
           convert the SAT solution to a propositional plan;
       else increase plan length by one, k ⟵ k + 1;
   until a solution is found.
end_procedure
```

Table 4: The framework of SATPLAN04+londex.

4 Experimental Results

As an application of londex, we incorporated londex constraints in SATPLAN04 [Kautz, 2004], a leading SAT-based optimal propositional planner which attempts to minimize the number of time steps. SATPLAN04 converts a planning problem to a Boolean satisfiability (SAT) problem and solves it using a SAT solver. SATPLAN04 uses an action-based encoding that converts all fact variables to action variables. Besides some goal constraints and action-precondition constraints, most of the constraints in the SAT are binary constraints modeling the mutex between actions.

We replaced the mutex constraints with londex constraints, and kept other components of SATPLAN04 unchanged. Following the action-based encoding of SATPLAN04, we used a Boolean variable $v_{i,j}$ to represent the execution of an action o_i in time step j. If an action o_{i_1} in time step j_1 and an action o_{i_2} in time step j_2 were mutually exclusive due to a londex constraint, the clause $\neg v_{i_1,j_1} \vee \neg v_{i_2,j_2}$ was generated.

The resulting algorithm SATPLAN04+londex is shown in Table 4. Starting from a lower bound k of the optimal plan length, SATPLAN04+londex iteratively solves, for each targeted plan length k, a converted SAT problem augmented by londex constraints. The method increases k until the optimal plan length is reached. We only need to generate all londex constraints once and save them for use by all steps.

The current default SAT solver in SATPLAN04 is Siege [Ryan, 2003]. In our experiments, we found that running Siege in SATPLAN04+londex caused memory-related errors.Since the source code of Siege is not available, it is impossible for us to modify its parameters to accommodate a large memory requirement. We thus chose to use a different SAT solver Minisat [Eén and Biere, 2005] in SATPLAN04+londex. We ran all our experiments on a PC workstation with Intel Xeon 2.4 GHZ CPU and 2 GB memory.

Domains	N	londex+minisat	minisat	Siege
openstacks	30	3	0	0
TPP	30	21	21	22
truck	30	6	3	3
storage	30	16	14	12
pathways	30	9	8	8
pipeworld	50	7	7	9
rovers	40	29	16	16

Table 5: The number of instances each method can solve for the IPC5 domains. N is the total number of problem instances in each domain. We highlight in box the better one between londex+minisat and minisat if there is a difference.

To evaluate the effect of londex constraints, we compared SATPLAN04 with Minisat (denoted as *minisat* in Tables 5 and 6) and SATPLAN04+londex with Minisat (denoted as *londex+minisat* in the tables). As a reference, we also ran the original SATPLAN04 with Siege (denoted as *Siege* in the tables). All these algorithms have the same solution quality (in terms of number of time steps) since they are optimal, so we focused on comparing their running time.

Table 5 shows the performance of the three algorithms on the seven planning domains used in the 5^{th} IPC (IPC5) [IPC, 2006]. These domains were selected from a variety of real-world application domains, such as transportation scheduling (truck, TPP), production management (openstacks), crate storage (storage), molecular biology (pathways), petroleum transportation (pipeworld), and rover motion planning (rover). Each domain contains a number of problems, and the table includes the number of problems in each domain which each algorithm can solve under a thirty-minute time limit.

As shown Table 5, Minisat with londex can solve more problem instances in all the domains than Minisat with mutex. For example, SATPLAN04 with Minisat can only solve 16 problems in rovers, whereas SATPLAN04+londex with Minisat can solve 29 problems. Furthermore, although not directly comparable due to the use of different solvers, SATPLAN04+londex with Minisat outperforms SATPLAN04 with Siege in 5 out of the 7 domains. The problems only solvable by SATPLAN04+londex but not others are typically very large, with tens of millions of variables and clauses.

Since most methods can easily solve small problem instances in IPC5 and the problem difficulty in each domain grows exponentially with problem size, we are particularly interested in the performance of the three methods on large, difficult instances. Table 6 shows the solution times for the three largest problems that at least one of the three methods can solve in each domain. The result shows that except for the TPP and pipesworld domains, using londex constraints with Minisat can help SATPLAN04 solve more large, difficult instances in substantially less time. We note that most of DTGs generated from the TPP and pipesworld domains contain only few fact nodes. As londex reasoning can only generate constraints between facts (or associated actions) within the same DTG, fewer fact nodes in the same DTG usually means fewer

Problem	londex+minisat	minisat	Siege
openstacks1	1303	-	-
openstacks3	1526	-	-
openstacks4	1120	-	-
TPP20	557	934	464
TPP22	1085	723	408
TPP24	-	-	779
truck4	1410	-	-
truck5	1233	-	-
truck7	1528	-	-
storage14	96.6	330	
storage15	108	1153	102
storage16	1533	-	
pathways7	27.3	83.6	39.4
pathways10	34.1	5.2	4.9
pathways15	643	-	-
pipesworld8	393	225	559
pipesworld11	-	-	724
pipesworld41	-	-	210
rovers27	505	-	-
rovers28	399	-	-
rovers29	494	-	-

Table 6: The solution time in seconds for some largest problem instances from IPC5. For each domain, we show the three largest problems that at least one method can solve. "-" means timeout after 1,800 seconds. We highlight in box the better one between londex+minisat and minisat if there is a difference.

(and thus weaker) londex constraints. Therefore, the relatively unsatisfactory performance of SATPLAN04+londex in the two domains is likely due to this special property of DTGs on these two domains.

We also notice that SATPLAN04+londex with Minisat is extremely efficient on the openstacks, trucks, and rovers domains. We are studying the underlying structures of these three domains to further understand the reasons for the superior performance.

5 Related Work and Discussion

The notions of multi-value formulations and domain transition graph can be traced back to the work of Jonsson and Backstrom on planning complexity [Jonsson and Backstrom, 1998], and even to some earlier work of Backstrom and Nebel on SAS+ system [Backstrom and Nebel, 1995]. The use of such representations can result in more compact encodings than Boolean-fact representations. Example planners employing these representations include Fast Downward [Helmert, 2006] and the IPP planner [Briel *et al.*, 2005]. Fast Downward is a best-first search planner that utilizes the information from domain transition graph as the heuristic to guide the search. IPP is a planner based on integer programming.

Several previous works have exploited various kinds of mutex reasoning or constraint inference among state variables. For example, forward and backward reachability analysis has been studied in order to generate more mutex than Graphplan [Do *et al.*, 1999; Brafman, 2001]. Fox and Long used a finite state machine graph to enhance the mutex reasoning in the STAN planner [Fox and Long, 1998]. Bonet and Geffner in HSP proposed another definition of persistent mutex pairs by recursively finding the conflicts between one action and the precondition of the other action [Bonet and Geffner, 2001]. However, all these previous mutexes relate actions and facts at the same level, and do not provide constraints for actions and facts across multiple leves like londex does. Kautz and Selman [Kautz and Selman, 1998] as well as McCluskey and Porteous [McCluskey and Porteous, 1997] studied hand-coded domain-dependent constraints. Other earlier works that explore constraint inference include [Kelleher and Cohn, 1992] and [Morris and Feldman, 1989].

Our londex constraints are derived from the fact (or action) distances. An earlier attempt along this line of research (i.e., to derive constraints from distance) was made by Chen and van Beek [van Beek and Chen, 1999] who have used hand-coded distance constraints in their CSP formulation for planning. Vidal and Geffner also used distance constraints in their CPT planner [Vidal and Geffner, 2004]. However, these distance constraints require either domain knowledge or an expensive reachability analysis that depends on the initial states. Comparing to them, londex constraints based on DTGs are not only domain-independent, but also much richer and stronger to capture constraints across different time steps, as it provides a systematic representation of state-independent distance constraints.

A SAT-based planner falls into the broad category of transformation-based planning [Selman and Kautz, 1992; Kautz *et al.*, 1996; Kautz and Walser, 2000; Wolfman and Weld, 2000; Wah and Chen, 2006; Chen *et al.*, 2006], which transforms a planning problem into a different formulation before solving it. However, no consensus has yet been reached regarding which formulation is the best among all that have been considered. Furthermore, all transformation-based planners used generic constraint solvers to solve encoded problems. However, by treating constraint solving in a black-box manner, intrinsic structural properties of planning problems are largely ignored.

6 Conclusions and Future Work

We have proposed a new class of mutual exclusion constraints that are capable of capturing long distance mutual exclusions across multiple steps of a planning problem. We abbreviated such long distance mutual exclusions as *londex*. Londex constraints are constructed by the MDF and DTG representations of a propositional planning problem. Londex contains as a subset the persistent mutex which includes only constraints within a same time step, and provides a much stronger pruning power. We have incorporated londex in SATPLAN04 and experimentally evaluated the impact of londex to planning efficiency on a large number of planning domains. Our experimental results show that londex significantly reduces the planning time in most domains.

Even though we have studied londex under the context of optimal planning, it can be directly applied to other planning paradigms, such as stochastic search in LPG, mathematical programming in SGPlan [Wah and Chen, 2006; Chen *et al.*, 2006], and integer programming in the IP planner [Briel *et al.*, 2005]. We are also studying how to extend londex to temporal planning.

7 Acknowledgments

The research was supported in part by NSF grants ITR/EIA-0113618 and IIS-0535257, and an Early Career Principal Investigator grant from the Department of Energy. We thank the anonymous referees for their very helpful comments.

References

[Backstrom and Nebel, 1995] C. Backstrom and B. Nebel. Complexity results for SAS+ planning. *Computational Intelligence*, 11:17–29, 1995.

[Blum and Furst, 1997] A. Blum and M. L. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300, 1997.

[Bonet and Geffner, 2001] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

[Brafman, 2001] R. I. Brafman. On reachability, relevance, and resolution in the planning as satisfiabilty approach. *Journal of Artificial Intelligence Research*, 14:1–28, 2001.

[Briel *et al.*, 2005] M. Briel, T. Vossen, and S. Kambhampati. Reviving integer programming approaches for ai planning: A branch-and-cut framework. In *Proc. ICAPS*, pages 310–319, 2005.

[Bylander, 1994] T. Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

[Chen *et al.*, 2006] Y. Chen, B. W. Wah, and C. Hsu. Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal Artificial Intelligence Research*, 26:323–369, 2006.

[Do *et al.*, 1999] M. Binh Do, B. Srivastava, and S. Kambhampati. Investigating the effect of relevance and reachability constraints on SAT encoding of planning. In *Proc. AAAI*, page 982, 1999.

[Eén and Biere, 2005] N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proc. SAT*, 2005.

[Fox and Long, 1998] M. Fox and D. Long. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research*, 9:367–421, 1998.

[Gerevini and Schubert, 1998] A. Gerevini and L. K. Schubert. Inferring state constraints for domain-independent planning. In *Proceedings of AAAI-98*, pages 905–912, 1998.

[Gerevini *et al.*, 2003] A. Gerevini, A. Saetti, and I. Serina. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research*, 20:239–290, 2003.

[Green, 1969] C. Green. Applications of theorem proving to problem solving. In *Proceedings of IJCAI-69*, pages 219–240, 1969.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 2006. Accepted for publication.

[IPC, 2006] The fifth international planning competition. *http://zeus.ing.unibs.it/ipc-5/*, 2006.

[Jonsson and Backstrom, 1998] P. Jonsson and C. Backstrom. State-variable planning under structural restrictions: Algorithms and complexity. *Artificial Intelligence*, 100(1-2):125–176, 1998.

[Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of AAAI-96*, pages 1194–1201, 1996.

[Kautz and Selman, 1998] H. Kautz and B. Selman. The role of domain-specific knowledge in the planning as satisfiability framework. In *Artificial Intelligence Planning Systems*, pages 181–189, 1998.

[Kautz and Walser, 2000] H. Kautz and J. P. Walser. Integer optimization models of AI planning problems. *The Knowledge Engineering Review*, 15(1):101–117, 2000.

[Kautz *et al.*, 1996] H. Kautz, D. McAllester, and B. Selman. Encoding plans in propositional logic. In *Proceedings of KR-97*, pages 374–384, 1996.

[Kautz, 2004] H. Kautz. SATPLAN04: Planning as satisfiability. In *Proceedings of IPC4, ICAPS*, 2004.

[Kelleher and Cohn, 1992] G. Kelleher and A. G. Cohn. Automatically synthesising domain constraints from operator descriptions. In *Proceedings of ECAI-92*, 1992.

[McCluskey and Porteous, 1997] T. L. McCluskey and J. M. Porteous. Engineering and compiling planning domain models to promote validity and efficiency. *Artificial Intelligence*, 95:1–65, 1997.

[Morris and Feldman, 1989] P. Morris and R. Feldman. Automatically derived heuristics for planning search. In *Proceedings of the Second Irish Conference on Artificial Intelligence and Cognitive Science*, 1989.

[Penberthy and Weld, 1994] J. Penberthy and D. Weld. Temporal planning with continuous change. In *Proc. 12th National Conf. on AI*, pages 1010–1015. AAAI, 1994.

[Ryan, 2003] L. Ryan. Efficient algorithms for clause-learning SAT solvers. Master's thesis, Simon Fraser University, 2003.

[Selman and Kautz, 1992] B. Selman and H. Kautz. Planning as satisfiability. In *Proceedings of ECAI-92*, pages 359–363, 1992.

[van Beek and Chen, 1999] P. van Beek and X. Chen. CPlan: A constraint programming approach to planning. In *Proceedings of AAAI-99*, pages 585–590, 1999.

[Vidal and Geffner, 2004] V. Vidal and H. Geffner. CPT: An optimal temporal POCL planner based on constraint programming. In *Proc. IPC4, ICAPS*, pages 59–60, 2004.

[Wah and Chen, 2006] B. W. Wah and Y. Chen. Constraint partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence*, 170(3):187–231, 2006.

[Wolfman and Weld, 2000] S. Wolfman and D. Weld. Combining linear programming and satisfiability solving for resource planning. *The Knowledge Engineering Review*, 15(1), 2000.