# An Optimal Temporally Expressive Planner: Initial Results and Application to P2P Network Optimization

**Ruoyun Huang,Yixin Chen, Weixiong Zhang**
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130, USA
{rh11,chen,zhang}@cse.wustl.edu

## Abstract

Temporally expressive planning, an important class of temporal planning, has attracted much attention lately. Temporally expressive planning is difficult; few existing planners can solve them, as they have highly concurrent actions. We propose an *optimal* approach to temporally expressive planning based on a SAT formulation of the problem, finding solutions with the shortest time spans. Our experiments on several temporally expressive domains showed that our planner is able to optimally solve many instances in a reasonable amount of time, comparing favorably to existing temporally expressive planners.

Our second result is a temporally expressive planning problem formulation of the Peer-to-Peer (P2P) network communications. In addition to demonstrating a better performance of our new method than the only existing temporally expressive planners on several temporally expressive problem domains, we apply our new planner to find optimal communication schedules for P2P networks. Our results will be potentially useful for designing efficient communication protocols in P2P networks.

## Introduction

Many real-world planning problems are inherently temporal and contain more intrinsically concurrent actions than what we have assumed before. The high concurrency of actions has not been sufficiently exploited in most existing planning systems. In particularly, the latest results have shown that a large class of temporal planning problems, the temporally expressive problems, have not been adequately studied (Cushing et al. 2007a; 2007b). This is in part due to the lack of a sufficient number of benchmarks that are truly temporally expressive and representative of real-world problems. The lack of sufficient temporally expressive domains is also a barricade to future advance in temporal planning.

### Temporal planning

Temporal planning is an important class of planning, as temporal constraints are inherent in most real-world planning problems. Temporal planning is also difficult and much more complex than propositional planning. Despite that temporal planning is important and much effort has been devoted to it, most existing temporal planners, including SGPlan (Wah and Chen 2006) and CPT (Vidal and Geffner 2006), do not support temporally expressive planning (Cushing et al. 2007a). Part of the reason is that these existing planners made some assumptions on how actions interact with one another. The only existing PDDL based temporally expressive planner that we are aware of is Crikey (Coles et al. 2008b; 2008a). Crikey combines planning and scheduling for temporal problems, and uses state-based forward heuristic search, which is Enforced Hill Climbing (EHC) followed by Best-First Search if EHC fails. A recent work in (Hu 2007) theoretically studied compilation of temporally expressive problems by a constraint satisfaction formulation.

The lack of optimal temporally expressive planners is in sharp contrast with the reality that many real-world planning problems are highly concurrent. The SAT-based planning approach (Kautz, Selman, and Hoffmann 1999) has been proven to be a very successful paradigm for propositional planning, especially for optimizing parallel actions and minimizing time steps. It has also been applied to several other types of planning problems (Giunchiglia and Maratea 2007; Mattmüller and Rintanen 2007). Inspired by the enormous success of the SAT-based planning paradigm, we adopted its basic idea to formulate temporally expressive planning problems, and developed a new temporally expressive planner. Our work was also in part inspired by the result in (Pham, Thornton, and Sattar 2008), which studied the advantage of applying a SAT-based method to general temporal problems. In particular, our results showed that the SAT-based approach is also a good choice for temporally expressive planning, especially when the problems are highly concurrent.

### Optimizing P2P networks

As pointed out in (Cushing et al. 2007b), all the temporal planning domains in the recent planning competitions are not temporally expressive. The shortage of a sufficient number of temporally expressive planning domains may refrain future development of temporal planning. Aiming in part at expanding our pool of temporal planning domains, we consider a critical problem in networks, which is briefly discussed below.

In Peer-to-Peer (P2P) networks, each computer, called a peer, may upload or download data from one another. In

Copyright © 2009, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

such a network, file transmission is no longer limited by the bandwidth of a single centralized server, thus the overall throughput within the network is significantly increased. A large number of services have been developed for P2P networks and more related distributed systems (Subramanian and Goodman 2005) keep coming out; it may potentially be used in multi-core CPU architecture, a trend in current computer architecture design. One critical issue in P2P networks is that a substantial amount of inter-peer data communication traffic is unnecessarily duplicated.

For those systems having consistent and intensive data sharing between peers, communication latency is a potential bottleneck of the overall network performance. It is highly desirable, when designing a P2P network, to have not only a design plan, but also an optimal one to optimize the potential utility of the network. Mechanisms in network design, particularly proxy caching (a.k.a. gateway caching), have been proposed to reduce duplicated data transmission. Making a good use of the proxy cache is critical for optimizing data transmission.

The problems of duplicated communication traffic and communication latency are intrinsically related. To reduce or remove duplicated communication and to effectively use proxy cache amount to optimizing communication traffic. The main issue is to determine each peer's actions at every time point, with the objective of letting all peers get all data requested within the shortest possible time. Due to its importance, optimizing communication traffic in P2P networks has already attracted some attention (Zhang, Goel, and Govindan 2009). The problem, when casted as a planning problem, is temporally expressive.

In this work, we approach the problem of optimizing communication traffic in P2P networks from the viewpoint of temporally expressive planning, propose a SAT-based formulation of the problem, and develop a new temporally expressive planner for the problem. The most salient feature of our approach is its optimality in finding a solution with the shortest time span. There are some existing methods for general network planning problems. For example, the one in (Rudenko 2002), is heuristic based and specialized (domain specific) planning algorithm instead of applying automated planning methods. To the best of our knowledge, our approach is the first of this kind for P2P networks and networks in general which are optimal and temporally expressive.

This paper is organized as follows. We first review the temporal planning in the first section and consider the SAT encoding for temporal planning problems. Next, the problem of optimizing P2P networks is discussed and modeled. After that, We present our experimental results on the P2P network domain and several other domains. Finally, we conclude in the last section.

## Temporal Planning

A **fact** $f$ is an atomic proposition that can be either true or false. We denote $f_t$ when $f$ is true at time $t$. An action $o$ is defined by a tuple $(\rho, \pi_{\vdash}, \pi_{\leftrightarrow}, \pi_{\dashv}, \alpha_{\vdash}, \alpha_{\dashv})$, where $\rho$, a constant, is the duration of $o$; $\pi_{\vdash}, \pi_{\leftrightarrow}, \pi_{\dashv}$ are the conditions that must hold at the start, over its lifetime, and at the end of action $o$; $\alpha_{\vdash}, \alpha_{\dashv}$ are the add-effects at the start or the end of $o$, respectively.

Note that the time in this formulation is treated as discrete. Therefore, assuming $o$ is executed at time $t$, which implies that $o$ ends at time $t + \rho - 1$, we need all the conditions to be satisfied as follows: a) $\forall f \in \pi_{\vdash}, f_t$ is true, b) $\forall f \in \pi_{\dashv}, f_{t+\rho-1}$ is true, and c) $\forall f \in \pi_{\leftrightarrow}, t' \in (t, t+\rho-1), f_{t'}$ is true.

**Definition 1** (Temporal planning) *An instance of temporal planning is defined as a tuple $(I, F, O, G, A)$, where, $I$ is the initial state, $F$ is a set of facts, $O$ is a set of durative actions, $G$ is a set of goal facts, and $A$ is a set of axioms.*

This formulation of temporal planning is a subset of PDDL2.1. For simplicity, we assume each action $o \in O$ to be durative and grounded. We use the method proposed in (Helmert 2008) to translate conditional quantifiers into axioms, which are denoted in a way that is similar to simple actions. Given an axiom $a$, we denote $pre(a)$ as its conditions, and $eff(a)$ as its instantaneous effects.

**Temporally expressive** The concept of temporally expressive planning was first proposed in (Cushing et al. 2007a; 2007b), from which we adopt the definition of required concurrency.

**Definition 2** (Required concurrency) *A problem has a required concurrency if there exists a plan for solving the problem and every solution has concurrently executed actions.*

A temporal planning problem is temporally expressive if it has a required concurrency, otherwise, it is *temporally simple*.

## SAT-based Temporally Expressive Planning

In this section, we formulate temporal planning using a SAT-based approach (Kautz, Selman, and Hoffmann 1999). Our overall procedure is in Algorithm 1. Each iteration of the procedure increases the time span by a fixed step size. A set of partial order variables for the goals is used to indicate that the goal state is achieved at time $t$ (Ray and Ginsberg 2008). A modified SAT solver solves the instance by assuming one set of the goal variables to be true. As such, we solve the problem with multiple time spans at each iteration.

Note that in (Ray and Ginsberg 2008), a planning graph is first generated to help estimate the step size. Although there exists earlier research in applying planning graph to temporal planning, all existing works that we are aware of either have limited expressiveness (Smith and Weld 1999) or are unable to optimize time span (Long and Fox 2003). These two shortcomings may lead to an overestimation on step size even for the first iteration of the planning. Therefore, in this work we will use a predefined constant for the step size. To estimate a better step size is an interesting open problem and will be part of our future work.

### Transform durative actions

First of all, each durative action $o$ is converted into two simple actions plus one propositional fact, written as $\Psi(o) =$

**Algorithm 1**: SAT-based Temporally Expressive Planning (STEP)

**Input**: A temporally expressive planning problem
**Output**: A solution plan

transform durative actions into simple ones;
set $\delta$ as the step size;
$N \leftarrow 0$;
$Z \leftarrow$ max number of time spans;
**repeat**
  $N \leftarrow N + \delta$;
  encode the problem with partial order goal variables between $N - \delta + 1$ and $N$;
  solve the encoded SAT instance;
**until** *a solution is found or* $N > Z$ ;
**if** *a solution found* **then**
  decode the solution and return;
**else**
  return with no solution;

$(o_\vdash, o_\dashv, f^o)$. These two simple actions indicate the starting and ending event of $o$. The fact $f^o$, when it is true, indicates that $o$ is executing. We denote the set of all such $f^o$ as $F^o = \{f^o \mid o \in O\}$. We further denote $pre(o)$, $add(o)$ and $del(o)$ as the set of preconditions, the set of add-effects and the set of del-effects of a simple action $o$, respectively.

We transform a planning problem $(I, F, O, G, A)$ into $(I, F^s, O^s, G, A)$. Here, $F^s$ is $F \cup F^o$, and $O^s$ is $\{a_\vdash, a_\dashv \mid \forall a \in O\} \cup \{Noop\ Action\ for\ f \mid \forall f \in F^s\}$.

The idea of transforming durative actions was proposed in (Long and Fox 2003). It has several advantages. For example, some techniques from classical planning can be applied without sacrificing the completeness.

Given the above planning problem representation, it is necessary to encode action mutual exclusion (mutex) constraints to ensure the correctness of solutions. Several algorithms were proposed to detect the mutexes between durative actions in temporal planning (Smith and Weld 1999). Here we compute those required action mutexes for all transformed actions $o \in O^s$, and use them in the encoding.

### Encoding in each iteration

We extend the encoding of propositional planning in planning graph to temporal planning using the above transformation. Given a time span $N$ and a problem instance $(I, F^s, O^s, G, A)$, we define the following variables for the encoding.

1. action variables $U_{o,t}$, $0 \le t \le N, o \in O^s$.
2. fact variables $V_{f,t}$, $0 \le t \le N, f \in F^s$.
3. goal set variables $W_t$, $N - \delta + 1 \le t \le N$.

We also need the following clauses for the encoding.

1. Initial state (for all $f \in I$): $V_{f,0}$
2. Partial order goal states (for all $t \in [N - \delta + 1, N]$): $W_t \rightarrow \bigwedge_{\forall f \in G} V_{f,t}$
3. Preconditions (for all $o \in O^s$,$0 \le t \le N$): $U_{o,t} \rightarrow \bigwedge_{\forall f, f \in pre(o)} V_{f,t}$
4. Add-Effects (for all $f \in F^s$,$0 \le t \le N$): $V_{f,t+1} \rightarrow \bigvee_{\forall o, f \in add(o)} U_{o,t}$
5. Delete-Effects (for all $o \in O^s$,$0 \le t \le N$): $U_{o,t} \rightarrow \bigwedge_{\forall f, f \in del(o)} \neg V_{f,t+1}$
6. Durative action information ($\forall o, t, o \in O$,$0 \le t \le N$): $U_{o_\vdash,t} \leftrightarrow U_{o_\dashv,t+\rho-1}$ $U_{o_\vdash,t} \rightarrow \bigwedge_{t<t'<t+\rho-1}(V_{f^o,t'} \wedge \bigwedge_{f \in o_\leftrightarrow} V_{f,t'})$
7. Axioms (for each $a \in A$,$0 \le t \le N$): $\bigwedge_{f \in pre(a)}(V_{f,t}) \rightarrow \bigwedge_{f' \in eff(a)}(V_{f',t})$
8. Action mutex: for all mutex actions $(o_1, o_2)$, $U_{o_1,t} \rightarrow \neg U_{o_2,t}$
9. Fact mutex: for all mutex facts $(f_1, f_2)$, $V_{f_1,t} \rightarrow \neg V_{f_2,t}$

**Theorem 1** (Optimality) *The STEP algorithm in Algorithm 1 always finds the solution with the minimal time span, if such a solution exists.*

Starting with 0, each iteration of Algorithm 1 increases the time span by $\delta$, and then encodes and solves the SAT instance. Therefore, if there exists a solution with the minimal time span, Algorithm 1 will always find it as the first solution it encounters. The partial order of goal variables is enforced in the SAT solver so that the first solution found will give the minimum time span. Therefore, Algorithm 1 is optimal in total time span.

**Expressiveness of parallelism** Our approach is not only powerful enough to handle the temporally expressive semantics, but also capable of handling some other attributes regarding parallelism in temporal planning. According to the analysis in (Rintanen 2007), whether a temporal planning problem can be compiled into a classical planning problem in polynomial time is determined by whether self-overlapping is allowed. Our approach supports some of the self-overlappings.

For a given action $o$ and time $t$, we have variables $U_{o_\vdash,t}$ and $U_{o_\dashv,t}$ representing the starting and ending actions of $o$, respectively. Suppose action $o$ has two instances, starting at time $t$ and time $t'$ ($t < t'$), respectively. For the starting action $o_\vdash$, we have different variables $U_{o_\vdash,t}$ and $U_{o_\vdash,t'}$ to indicate the different starting times of the two instances. Those $f^o$ facts, along with all related conditions, will be enforced to be true from $t$ to $t' + \rho$. Thus, these invariant conditions of the two action instances do not exclude each other's existence.

However, the current encoding cannot handle the case when the starting times of the two instances are the same, or the ending times of the two instances are the same. This is because given a simple action $o$, for each time point $t$, we have only one binary variable to indicate if $o$ is executed at $t$.

## Modeling Communications in P2P Networks

A small number of benchmark domains exist for temporally expressive planning; all the benchmark domains in

IPC3, IPC4 and IPC5 are temporally simple (Cushing et al. 2007b). Introducing a new, sound and scalable temporally expressive domain is nontrivial. A few candidates for temporally expressive domains were proposed in (Cushing et al. 2007b; Coles et al. 2008b). Most of them, however, are not truly representative of real-world problems. We develop a temporally expressive domain based on communication optimization in P2P networks using the PDDL language, and apply our planner to this domain.

There are at least two different types of optimization in P2P networks. One of them is approached from the user's point of view. Each individual user wants all the data needed within the shortest possible time (Bhattacharya and Ghosh 2007). The other type is approached from the point of view of a network service provider (such as an internet service provider (ISP)), who owns the network but does not control individual peers. The main concern of a service provider is to reduce the overall communication load.

These two performance metrics are apparently mutually exclusive. In this paper we adopt a third performance metric which lies in the middle ground of the above mentioned two. Under this metric, the network owner knows each peer's needs, and the objective is to minimize the overall time span for all the data delivery for all peers.

The major constraint in this problem is that to satisfy a file request from a peer $p_1$, there must be such a file in another peer $p_2$. $p_1$ can execute the action *download* to get the file, when: 1) $p_1$ and $p_2$ are routed, and 2) $p_2$ is *serving* the file throughout the networks. As such, these two actions require a concurrency for a valid plan.

In addition, the proxy cache will guarantee that, when $p_2$ is *serving* a file, any peer who is routed to $p_2$ can *download* the file very fast. The upload bandwidth of a peer is typically much narrower than its download bandwidth. Therefore, enforced by the optimality, the more peers downloading this particular file, the larger the whole network's throughput will be, which bring about a shorter time span in a plan solution.

Due to the space limitation, below we only specify the PDDL definition for a *serve* action.

```
(:durative-action serve
 :parameters ( ?c - computer ?f - file )
 :duration (= ?duration (file-size ?f) )
 :condition( and
        (at start (free ?c) )
        (over all (not(free ?c )))
        (at start (saved ?c ?f))
        (over all (serving ?c ?f) ))
 :effect(and
        (at start  (not (free ?c )) )
        (at end    (free ?c ) )
        (at start  (serving ?c ?f) )
        (at end(not (serving ?c ?f) )))
```

In the definition of the *serve* action, the serving time of a file is proportional to its file size. We assume that by actively sharing a file, the uploading peer uses up its uploading bandwidth. That is, we assume that it cannot share another file simultaneously. This assumption will not impose a real restriction as we can introduce a time sharing scheme to extend the method we develop. A predicate 'serving' as one of the add-effects at the beginning indicates that the peer is sharing a file.

When sharing a file from a peer, the connected router will guarantee that any other peer can get this file in a constant time (because download speed is much faster), as long as it is routed to the uploading peer.

It is worthwhile to note that a good temporally expressive domain must have, but not limited to, at least the following three properties:

- Scalability in problem size (i.e. in time span);
- Scalability in degree of concurrency;
- Representativeness of a real-world application domain or problem, to be useful and for further development.

Being motivated by a real world problem, the communication optimization in P2P domain is scalable in problem size as an arbitrary number of peers and files may participate. It is also scalable in degree of concurrency as peers can be routed very differently.

**High concurrency** Communication in P2P networks is unique, since many uploading peers may get involved simultaneously in a typical download event. Consequently, this gives rise to a high concurrency in the resulting temporally expressive planning, which is very different from the most temporal planning problems we have seen. To have a profound understanding of this high concurrency in our new temporal planning domain, we examine temporal dependency here.

**Definition 3** (Temporal dependency) *Given two actions $o$ and $o'$, we say $o$ temporally depends on $o'$ when either of the following conditions holds:*

1. $\exists f \in \pi(o)$, *such that* $f \in \alpha_{\vdash}(o')$ *and* $\neg f \in \alpha_{\dashv}(o')$;
2. $\exists \neg f \in \pi(o)$, *such that* $\neg f \in \alpha_{\vdash}(o')$ *and* $f \in \alpha_{\dashv}(o')$.

Two factors can lead to concurrencies in a temporally expressive problem. One is the required concurrent interaction (i.e., concurrent execution) among actions, and the other is enforced deadline (Coles et al. 2008b). Note that a temporal dependency among actions is just a necessary condition for required concurrencies. A problem is temporally expressive only when there exists no action that can achieve the goals if the temporal dependency is ignored. When solution optimality is enforced in the P2P domain, required concurrencies are naturally introduced in order to reduce the overall time span.

Figure 1 illustrates the temporal dependencies in several instances from different domains. All these instances have comparable problem sizes. The instance of P2P domain has 90 facts and 252 actions, and the instance of Matchlift domain (Coles et al. 2008b) has 216 facts and 558 actions. Figure 1(I) is an instance of Trucks domain, which is temporally simple, thus all actions are isolated. In Figure 1(II), each action has up to two actions temporally depending on it. In Figure 1(III), each action has up to five actions temporally depending on it.

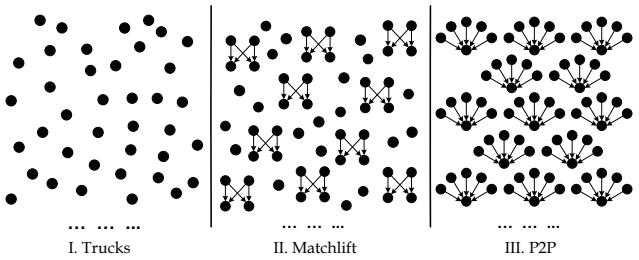


Figure 1: This figure partially illustrates temporal dependencies of actions for instances in three domains, Trucks, Matchlift and P2P. Each node represents an action. Each edge represents a temporal dependency between two actions.

We call these problems such as the P2P domain highly concurrent. The reason is that, with similar sizes, they have more temporal dependencies than their counterparts of temporally simple problems. The overall required time span for a highly concurrent temporally expressive problem is expected to be short.

## Experimental Results

In our current implementation of SAT-based temporally expressive planner (STEP), we used MiniSAT2 (Een and Sörensson 2003) as its SAT solver. We studied the performance of STEP and compared it with Crikey2 (Coles et al. 2008a)(runnable java JAR) and Crikey3 (Coles et al. 2008b)(static statically-linked binary for x86 Linux). These are two different versions of the only practical temporally expressive planner we are aware of. It is important to emphasize that this is essentially a comparison of "apple vs. orange", since Crikey is fast but incomplete whereas STEP is relatively slow but optimal in total time span. Other well-known temporal planners were not considered because they only support temporally simple problems.

Our experiments were done on the P2P domain and several other temporally expressive domains (Coles et al. 2008b). The original domain definitions in (Coles et al. 2008b) do not have enough problem instances. Therefore, we generated most of the instances for our study. Note that we did not use all the domains in (Coles et al. 2008b) because some of them are not scalable to large problems (e.g. the Match domain), and a few of them have variable-duration actions (e.g. the Café domain), which we do not consider now.

We ran all experiments on a PC workstation with a 2.0 GHZ Xeon CPU and 2GB memory. The time limit, for each instance, was set to 3600 seconds. The step size $\delta$ of each new iteration in STEP was set to 5.

### The P2P domain

We first experimented on a collection of instances in the P2P domain, as shown in Table 1. The instances were generated randomly with different parameter settings, and the size of each file object was randomly chosen from four to eight units. The goal state for each instance was that each peer gets all requested files.

| P | C | F | STEP | | Crikey3 | | Crikey2 | |
|---|---|---|---|---|---|---|---|---|
| | | | Time | QAL | Time | QAL | Time | QAL |
| 1 | 4 | 4 | 6.21 | 27 | 1.21 | 72 | - | - |
| 2 | 5 | 5 | 25.05 | 34 | 17.42 | 100 | - | - |
| 3 | 6 | 6 | 131.62 | 44 | 300.6 | 150 | - | - |
| 4 | 7 | 6 | 334.19 | 54 | T | T | - | - |
| 5 | 6 | 7 | 369.96 | 49 | 773.75 | 200 | - | - |
| 6 | 7 | 7 | 3168.74 | 60 | T | T | - | - |
| 7 | 5 | 25 | 387.19 | 32 | T | T | - | - |
| 8 | 6 | 18 | 92.06 | 20 | T | T | - | - |
| 9 | 6 | 24 | 283.62 | 23 | T | T | - | - |
| 10 | 6 | 30 | 845.53 | 31 | T | T | - | - |
| 11 | 6 | 36 | 2713.81 | 36 | T | T | - | - |
| 12 | 7 | 35 | 1738.29 | 29 | T | T | - | - |

Table 1: Experimental results on the P2P domain. Column 'P' is the instance ID. Columns 'C' and 'F' are the numbers of peers and files, respectively, in the network. Columns 'Time' and 'QAL' are the running time and time span of solutions, respectively. 'T' means the solver ran out of the time limit of 3600s and '-' means no solution found.

There are two types of problems settings with different styles of network topology. One is loosely connected while the other is more highly connected.

Instances 1 to 6 all have simpler topologies. Each peer is connected to no more than two other peers. Also, in the initial state, only leaf peers (those that are only connected to one other peer) have files to share. There are less concurrencies in this setting. Crikey3 solved four out of the six these instances. It was faster on two simpler instances but slower than STEP on two other larger instances. Overall, the time spans found by Crikey3 were about three to five times longer than those found by STEP. Crikey2 failed to solve any instance in this category.

Instances 7 to 12 have more complicated network topology. Nearly all nodes are connected to one another. Every peer has some files needed by all others. In this setting, much higher concurrencies are required to derive a plan. Both Crikey2 and Crikey3 failed to solve any instance within the resource limit. Crikey3 timed out and Crikey2 reported no solution was found. It may be due to their incompleteness.

### The Matchlift domain

The results on the Matchlift domain (Coles et al. 2008b) are in Table 2. The original Matchlift domain had some flaws, in which an electrician's position was not updated until the ending of the durative action. This deficiency made both Crikey2 and Crikey3 blow up in the number of electricians, and eventually the electricians would exist everywhere. STEP is immune to this flaw, because it has the mutually exclusive actions encoded. As a result, STEP forbids two actions that delete the same fact to be executed at the same time. To make Crikey2 and Crikey3 work properly on this domain, we fixed the flaws and used the fixed version of Matchlift domain for this set of experiments.

We generated all instances randomly using different parameters for the numbers of floors, rooms, matches and

| P | L,M,R,U | STEP | | Crikey3 | | Crikey2 | |
|---|---|---|---|---|---|---|---|
| | | Time | QAL | Time | QAL | Time | QAL |
| 1 | 2,3,4,3 | 2.10 | 13 | 0.10 | 18 | 1.27 | 13 |
| 2 | 3,2,9,2 | 1.54 | 9 | 0.25 | 14 | 2.33 | 11 |
| 3 | 2,3,4,3 | 8.09 | 20 | 0.01 | 28 | 5.01 | 23 |
| 4 | 3,3,9,3 | 8.35 | 18 | 0.02 | 34 | 3.73 | 19 |
| 5 | 3,4,9,4 | 24.77 | 24 | 0.03 | 43 | 9.36 | 35 |
| 6 | 3,5,9,5 | 63.18 | 25 | 2.27 | 47 | 10.92 | 39 |
| 7 | 3,6,9,6 | 511.12 | 31 | 0.91 | 58 | 15.4 | 37 |
| 8 | 3,7,9,7 | 934.29 | 30 | 5.55 | 58 | 10.91 | 42 |
| 9 | 4,4,16,4 | 63.12 | 26 | 0.05 | 43 | 9.98 | 39 |
| 10 | 4,5,16,5 | 193.07 | 28 | 0.11 | 58 | 19.82 | 28 |
| 11 | 4,6,16,6 | 685.70 | 31 | 1.80 | 58 | 28.71 | 47 |

Table 2: Experimental results on the Matchlift domain. The numbers in Columns L, M, R, U represent the numbers of floors, matches, rooms and fuses, respectively, which were used in generating the instances.

| P | E,M,U | STEP | | Crikey3 | | Crikey2 | |
|---|---|---|---|---|---|---|---|
| | | Time | QAL | Time | QAL | Time | QAL |
| 1 | 2,2,4 | 2.95 | 13 | 11.65 | 19 | 2.5 | 14 |
| 2 | 2,1,4 | 1.59 | 13 | 20.17 | 18 | 16.23 | 13 |
| 3 | 2,3,5 | 7.45 | 18 | 0.02 | 23 | 1.87 | 19 |
| 4 | 2,2,5 | 30.67 | 21 | 108.85 | 27 | 13.70 | 25 |
| 5 | 2,4,6 | 22.70 | 22 | 0.04 | 33 | 2.91 | 29 |
| 6 | 2,2,6 | 34.66 | 21 | 105.16 | 27 | 15.43 | 25 |
| 7 | 3,2,7 | 140.60 | 16 | T | T | - | - |
| 8 | 3,2,7 | 441.41 | 16 | T | T | 16.52 | 17 |
| 9 | 3,4,8 | 110.37 | 22 | T | T | T | T |
| 10 | 3,2,8 | 699.33 | 20 | T | T | T | T |
| 11 | 4,3,8 | 1886.94 | 16 | T | T | 1052.56 | 24 |
| 12 | 4,1,8 | 3.86 | 13 | T | T | T | T |

Table 3: Experimental results on the Matchlift-Variant domain. The numbers in columns 'E, M, U' represent the numbers of electricians, matches, and fuses, respectively. 'T' means the solver ran out of the time limit and '-' means no solution found.

fuses. Each instance has the same number of fuses and matches. In other words, these instances are easier because we can always find a valid plan, such that there is exactly one fixing action concurrent with a lighting match action. In the future, we will have a variant domain of Matchlift for another set of experiment, where the number of match resources is less than the number of fuses to be fixed. As a result, it will require more concurrencies than needed in these simple Matchlift problem instances.

On all instances, Crikey3 is the fastest to find solutions, but with the poorest quality. STEP spent more time than Crikey3 and Crikey2 did, since the former found the optimal solutions. Among the three algorithms, Crikey2 is the second in terms of both solving time and solution quality (except the first instance, in which its quality is the same as that of STEP).

## The Matchlift-Variant domain

We also tested on a set of instances in a revised Matchlift domain (called Matchlift-Variant domain), which allow more concurrencies in two aspects. First, the number of matches is less than the number of fuses, resulting in that multiple electricians need to share one match. Second, we change the duration of a 'MEND_FUSE' action so that an electrician is able to conduct more actions during one match's lighting, which also results in higher concurrencies.

The results are shown in Table 3. All instances were generated with increasing numbers of fuses and electricians. All other settings were random. Instances with the same number of fuses and electricians might still have different degrees of concurrency, due to different numbers of matches and other resources available. For example, although Instances 7 and 8 have the same parameters, Instance 8 is more difficult than Instance 7 due to different ways how the fuses were distributed over the rooms.

As shown by our experimental results, STEP found optimal solutions on all instances tested, whereas Crikey2 and Crikey3 ran out of time on most instances and generated suboptimal plans on the few instances they finished. For the instances they solved, Crikey3 had the worst solution quality. It was very efficient in finding a solution in two instances, but in other four instances, it was even slower than STEP, which was able to find optimal solutions.

The results on Instances 11 and 12 are special and interesting. These two instances were generated under the same parameter setting, except for the number of matches. Instance 12 has only one match, which means the four electricians need to cooperate with each other perfectly to get all the fuses fixed. Comparing to Instance 11, Instance 12 turned out to be more difficult for Crikey2, because it required more concurrencies. As a result, Crikey2 solved Instance 11 but failed on Instance 12. As a comparison, Instance 12 was much easier than Instance 11 for STEP. STEP solved Instance 12 in just about three seconds, while spent more than 1800 seconds on Instance 11.

## The Driverlogshift domain

The problem instances in the Driverlogshift domain (Coles et al. 2008b) have much longer time spans than those in the Matchlift and P2P domains. The actions with duration of two were changed to three to distinguish $\vdash$, $\dashv$ and $\leftrightarrow$ conditions and effects. This change was made to accommodate STEP for solving discrete problems.

This domain is different from P2P and Matchlift. It has long durative actions, which give rise to longer time spans. Therefore, it is relatively difficult to optimally solve instances in this domain. These observations are reflected by our experimental result in Table 4.

We now compare STEP and Crikey3. As shown, the optimal time spans of the instances tested, provided by STEP, are typically much shorter than those by Crikey3. For example, the optimal time span for Instance 4 in Table 4 is about one third of the time span reported by Crikey3. As a tradeoff, STEP needs longer time for finding optimal solutions.

Now consider Crikey2, a suboptimal solver. Surprisingly, it was able to generate solutions of the same quality as what STEP found on most instances in this domain. However, a close examination led us to believe that some of the solutions produced by Crikey2 are flawed. For example, it may

| P | D,P,T | STEP | | Crikey3 | | Crikey2 | |
|---|---|---|---|---|---|---|---|
| | | Time | QAL | Time | QAL | Time | QAL |
| 1 | 2,2,2 | 249.72 | 102 | 0.21 | 224 | *3.6 | *122 |
| 2 | 2,2,2 | 585.36 | 122 | 0.2 | 122 | 2.37 | 122 |
| 3 | 2,3,2 | 734.99 | 122 | 0.4 | 125 | 3.7 | 122 |
| 4 | 2,3,2 | 763.48 | 122 | 0.52 | 323 | 4.62 | 122 |
| 5 | 2,4,2 | 240.76 | 102 | 0.55 | 238 | 16.12 | 102 |
| 6 | 2,4,3 | 366.75 | 118 | 0.91 | 326 | *1.76 | *122 |

Table 4: Experimental results on the DriverlogShift domain. The numbers in columns 'D, P, T' represent the numbers of drivers, packages, and trucks, respectively. The result marked with a '*' means the solution is invalid.

generate solutions with fragments of invalid action sequence as follows:

```
...
102:(REST driver2) [20.00]
102:(DRIVE-TRUCK truck1 s1 s0 driver2)[10.00]
...
```

Such a plan requires the same driver to perform two actions, *REST* and *DRIVE*, at the same time. It apparently violates the domain specification, which defines that *DRIVE* needs to be concurrent with a *WORK* action and the *WORK* action is mutual exclusive with the *REST* action.

### Number of variables and clauses

One may concern about the size of SAT encoding and the time complexity of STEP, which are issues any optimal planner faces. In Table 5, we list the numbers of variables and clauses of each instance (the last iteration encoding). Due to space limitation, we only show data on the P2P domain and the Matchlift-Variant domain. The solving time is presented to show the difficulty of the instances.

Similar to other SAT problems, it is obvious that the size of the encoding does not necessarily reflect the complexity of a problem. For example, the numbers of variables or clauses of Instance 6 in the P2P domain are only half of that of Instance 7. However, Instance 6 required computation ten times more than that required by Instance 7.

In general, the encoding sizes of STEP on current problem instances may be up to hundreds of thousands of variables, which are within the reach of current SAT solvers. Various boosting methods for SAT-based planning could also be applied to STEP. We plan to further study techniques such as encoding in new formulations (Robinson et al. 2008) and to derive constraint-based pruning clauses (Chen et al. 2009).

## Conclusions and Discussions

In this paper, we proposed and developed a novel SAT-based temporally expressive planning. A critical piece of our work was a SAT-based formulation of temporally expressive problems. Our work was motivated by the observation that high action concurrency is a main characteristic of temporally expressive planning problems. Such high concurrency is exemplified by the new P2P network communications domain

| P | P2P | | | Matchlift-Variant | | |
|---|---|---|---|---|---|---|
| | #VAR | #Clause | Time | #VAR | #Clause | Time |
| 1 | 498 | 1032 | 6.21 | 3632 | 13502 | 2.95 |
| 2 | 1538 | 4161 | 25.05 | 2583 | 8892 | 1.59 |
| 3 | 5799 | 17378 | 131.62 | 7122 | 33414 | 7.45 |
| 4 | 7076 | 19961 | 334.19 | 6431 | 29363 | 30.67 |
| 5 | 19230 | 58521 | 369.96 | 11608 | 66468 | 22.70 |
| 6 | 47046 | 152042 | 3168.74 | 6798 | 31826 | 34.66 |
| 7 | 95080 | 410983 | 387.19 | 7524 | 34137 | 140.60 |
| 8 | 59526 | 198216 | 92.06 | 7524 | 37663 | 441.41 |
| 9 | 91482 | 344504 | 283.62 | 18513 | 128674 | 110.37 |
| 10 | 95080 | 410983 | 845.53 | 10043 | 52352 | 699.33 |
| 11 | 59526 | 198216 | 2713.81 | 13601 | 75843 | 1886.94 |
| 12 | 91482 | 344504 | 1738.29 | 5449 | 21113 | 3.86 |

Table 5: The numbers of variables and clauses on P2P domain and Matchlift-Variant domain. Column '#VAR' is the number of variables, column '#Clause' is the number of clauses and column 'Time' is the overall solving time.

that we introduced. This type of problems makes the SAT-based approach an excellent choice as it is effective in handling parallel executions. Our experiments on several temporally expressive domains, including our new P2P network domain, showed that our new planner is able to *optimally* solve the instances in these domains in a reasonable amount of time, comparing favorably against the existing nonoptimal temporally expressive planners.

Our current work is an effort to advance the state-of-the-art of temporal planning. While we have reported some significant initial results, including a novel optimal temporally expressive planner and some encouraging experimental results on several problem domains, our work can be further improved. One limitation of our current SAT-based method is that it is not expressive enough for numerical constraints and other complex PDDL2.1 properties. For example, comparing to Crikey, the major limitation of our current implementation is that it does not support variable action durations. To fully support PDDL2.1 and take advantage of SAT-based planning technique, we are extending our new planner; we are implementing a new version by adding an outer layer to the planner to handle complex temporal constraints and to support richer semantics. Furthermore, in our current formulation of the P2P network domain, we ignored some technically involved issues, such as the cost for establishing a connection, the potential of data compression which might take additional processing time but shorten communication delay, file segmentation to further increase concurrency, and stochastic nature of communication channels. We are currently extending our modeling effort to incorporate these important features. Our goal is to develop a practical temporal planning system for this important network communication optimization problem.

## Acknowledgement

The research was supported by NSF grants IIS-0535257, DBI-0743797, IIS-0713109, a DOE ECPI award, and a Microsoft Research New Faculty Fellowship.

## References


Bhattacharya, A., and Ghosh, S. 2007. Self-optimizing peer-to-peer networks with selfish processes. In *Proc. of Int'l Conf. on Self-Adaptive and Self-Organizing Systems*.

Chen, Y.; Huang, R.; Xing, Z.; and Zhang, W. 2009. Long-distance mutual exclusion for planning. *Artificial Intelligence* 173:197–412.

Coles, A.; Fox, M.; Halsey, K.; Long, D.; and Smith, A. 2008a. Managing concurrency in temporal planning using planner-scheduler interaction. *Artificial Intelligence* 173:1–44.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008b. Planning with problems requiring temporal coordination. In *Proc. of AAAI-08*.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. S. 2007a. When is temporal planning really temporal? In *Proc. of Int'l Joint Conf. on AI*.

Cushing, W.; Kambhampati, S.; Talamadupula, K.; Weld, D. S.; and Mausam. 2007b. Evaluating temporal planning domains. In *Proc. of ICAPS-07*.

Een, N., and Sörensson, N. 2003. An Extensible SAT-solver.

Giunchiglia, E., and Maratea, M. 2007. Planning as satisfiability with preferences. In *Proc. of AAAI-07*.

Helmert, M. 2008. Concise finite-domain representations for pddl planning tasks. *Artificial Intelligence* 503–535.

Hu, Y. 2007. Temporally-expressive planning as constraint satisfaction problems. In *Proc. of ICAPS-07*.

Kautz, H.; Selman, B.; and Hoffmann, J. 1999. Unifying sat-based and graph-based planning. In *Proc. of Int'l Joint Conf. on AI*.

Long, D., and Fox, M. 2003. Exploiting a graphplan framework in temporal planning. In *Proc. of Int'l Conf. on Automated Planning and Scheduling*.

Mattmüller, R., and Rintanen, J. 2007. Planning for temporally extended goals as propositional satisfiability. In *Proc. of Int'l Joint Conf. on AI*.

Pham, D.; Thornton, J.; and Sattar, A. 2008. Modelling and solving temporal reasoning as propositional satisfiability. *Artificial Intelligence* 172:1752–1782.

Ray, K., and Ginsberg, M. L. 2008. The complexity of optimal planning and a more efficient method for finding solutions. In *Proc. of Int'l Conf. on Automated Planning and Scheduling*.

Rintanen, J. 2007. Complexity of concurrent temporal planning. In *Proc. of AAAI-07*.

Robinson, N.; Gretton, C.; Pham, D.; and Sattar, A. 2008. A compact and efficient sat encoding for planning. In *Proc. of ICAPS-08*.

Rudenko, A. 2002. *Automated Planning for Open Network Architectures*. Ph.D. Dissertation, UCLA.

Smith, D., and Weld, D. 1999. Temporal planning with mutual exclusion reasoning. In *Proc. of Int'l Joint Conf. on AI*.

Subramanian, R., and Goodman, B. D. 2005. *Peer to Peer Computing: The Evolution of a Disruptive Technology*. IGI Publishing.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence* 170:98–335.

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

Zhang, H.; Goel, A.; and Govindan, R. 2009. An empirical evaluation of internet latency expansion. *ACM SIGCOMM Computer Comms. Review* to appear.