Temporal Planning through Mixed Integer Programming: a Preliminary Report

Yannis Dimopoulos[1] and Alfonso Gerevini[2]

[1] Dep. of Computer Science
University of Cyprus
P.O. Box 20537
CY-1678 Nicosia, Cyprus
yannis@cs.ucy.ac.cy

[2] DEA – Universitá degli Studi di Brescia
Via Branze 38
I-25123 Brescia, Italy
gerevini@ing.unibs.it

Abstract. Temporal planning is an important problem, as in many real world planning domains actions have different durations and the goals should be achieved by a specified deadline, or as soon as possible. This paper presents a novel approach to temporal planning that is based on Mixed Integer Programming. In the new framework, a temporal planning domain is modeled by two sets of linear inequalities. The first set involves integer variables and is a Graphplan-like encoding of a simplification of the original problem where the duration of the actions is ignored. The second set involves both integer and real valued variables, and models the temporal aspects of the problem. The two sets interact through the common integer variables, and their combination can be solved by using available Mixed Integer Programming software. The new method aims at generating good solutions quickly, under different minimization objectives. Preliminary experimental results illustrate the effectiveness of our approach.

1 Introduction

Over the last years there has been a remarkable progress in solving STRIPS planning problems [16]. However, for many interesting applications the STRIPS language is inadequate. We need to solve problems that involve actions that have different durations, consume resources and must be executed by certain deadlines. We need to generate plans that optimize complex combinations of different criteria, including completion time, resource utilization, action costs and others.

Such advanced application domains involve *numeric* variables, constraints, and complex objective functions. Mixed Integer Programming (MIP), and its language of linear inequalities, can easily accommodate these key features and therefore seems to provide a rich representational framework for such applications. However, there are relatively few works that apply IP to planning problems with numeric constraints [17], [11].

In the original STRIPS language actions are instantaneous and time is implicitly represented. Several domain-independent systems have been proposed to handle a richer notion of time (e.g., [1, 14, 12]). However, these approaches scale up poorly, and can deal with only very simple problems. The success of recent approaches to STRIPS planning, such as planning graphs and heuristic search, has motivated the application of these techniques to temporal planning. For instance, TGP [13] uses a generalization of Graphplan mutual exclusion reasoning to handle actions with durations, while TP4 [10] applies heuristic search to solve problems with action durations and resources.

In this paper we apply MIP to temporal planning, by developing models for domains that contain actions with different durations. Our approach decomposes a planning domain into two interacting sets of linear inequalities referred to as the *logical* and the *temporal* part respectively.

The logical part is an encoding of the planning graph of the STRIPS problem that is obtained from the original problem by ignoring action durations. For this encoding we use the method developed by [15] and improved by [3]. This approach formulates the planning graph of a STRIPS planning domain as an Integer Programming problem and then uses branch-and-bound for solution extraction.

The temporal part associates with every action a real-valued variable that represents the start time of the action. The linear inequalities of this part ensure the correctness of the start times that are assigned to the actions taking into account their durations.

The combination of the logical and temporal parts can be solved in a uniform, integrated way by well-known MIP software like CPLEX. Since the two sets of constraints interact, this is much more effective than a naive approach in which the logical part is iteratively solved first, and each potential solution is checked against the constraints of the temporal part (which may include a constraint on the overall duration of the plan), until a valid (i.e., a solution for both the sets of constraints) or optimal temporal plan is found.

In order to increase the efficiency of the representation, the structure of the domain is exploited. In particular, we propose some techniques that use ideas from domain analysis tools [5, 7, 8] to reduce the number of constraints and variables of the temporal part of a planning problem, leading to stronger MIP formulations.

In contrast to TGP and TP4 that generate plans of minimum duration, the new approach does not provide optimality guarantees. However, apart from the overall duration of the plan, the MIP models can easily accommodate different optimization criteria and any constraint that can be expressed as a linear inequality. Furthermore, the new method is capable of generating high quality plans early in the computation.

The rest of the paper is organized as follows. First we briefly give the necessary background; then we present our basic temporal model, i.e., the set of inequalities forming the temporal part of the problem encoding (while for the logical part we will use some known encoding); then we describe how planning problems can be solved in the new approach; then we give some preliminary experimental results; finally, we give our conclusions and mention future work.

2 Preliminaries

The planning language we consider is propositional STRIPS extended with time. Actions have (positive) preconditions, (add and delete) effects and constant duration that can be any real number. Our assumptions for the execution of actions are the same as those used in [13] and [10]:

- The preconditions of an action must hold in the beginning and during the execution of the action.
- Add and delete effects take place at some point during the execution of an action and can only be used at the end of the execution of the action.

The above assumptions require that the preconditions and effects of an actions are protected during their execution. Therefore, the linear inequalities of the MIP models we develop, enforce that actions with contradictory effects or with contradictions between their effects and preconditions do not overlap in time.

A MIP problem [18] comprises of a mixture of real-valued and integer variables, a set of linear inequalities on these variables, and an objective function. The models developed in the paper are 0/1 MIP models i.e., integer variables can only assume the values 0 and 1. We assume that the reader is familiar with the basics of MIP.

Our modeling techniques utilize some ideas developed in the context of the domain analysis tool DISCOPLAN [8]. In particular, they exploit *single-valuedness* (SV) constraints and binary XOR-constraints, which are automatically inferred by DISCOPLAN. An SV-constraint states that the value of a certain predicate argument is unique for any given values of the remaining arguments. An example of an SV-constraint in blocks-world is $on(x, *y)$, stating that any object is on at most one thing ("$*$" indicates the single-valued argument). An example of XOR-constraint is (XOR $on(x, y)$ $clear(y)$) stating that any object is either clear or has something on it.

3 The Basic Temporal Model

Let P be a temporal planning problem, let P^S be its STRIPS simplification (i.e., actions are instantaneous), and assume a planning graph representation of P^S. For each action (instantiated operator) A of the problem and each level l of the graph, there is a corresponding node in the planning graph, denoted by A^l, that can be understood as a binary variable. A plan for P^S is essentially a value assignment to the action variables such that the value 1 is assigned to variable A^l iff action A at level l is included in the plan.

Our goal now is to find a set of linear inequalities for problem P that, given a plan for P^S, assign start times to the actions in the plan. The inequalities that model the temporal part of P involve, apart from the binary action variables A^l, a set of real valued variables as follows. For every action A and level l of the graph, we introduce a variable A^l_{st} (also denoted as $st(A^l)$) that represents the start time of action A at level l, i.e., the time when the execution of the action starts. Similarly, for every fluent f of the domain, and every level l of the graph, we introduce a variable f^l_{st} (also denoted as $st(f^l)$) that represents the time at which fluent f becomes true. In the following, $dur(A)$ denotes the duration of action A, which is a real number.

The first set of inequalities of the temporal model is used to enforce the constraint that actions can not start before their preconditions become true. If f is a precondition of A, the following set of inequalities is included in the model, for each level l of the planning graph

(1) $A^l_{st} \geq f^l_{st}$

The next set of inequalities represents the contraint that a fluent can become true after the execution of an action that adds it. Therefore, if f is an add effect of action A the model includes (A^l is a 0/1 variable)

(2) $f^{l+1}_{st} \geq A^l_{st} + dur(A) \cdot A^l$

Note that if fluent f was true at level l before the execution of action A, the above constraint causes f^{l+1}_{st} to take a value that can not be smaller than the end time of A. In combination with the previous constraint (1) this causes all actions that have f as their precondition, and appear in levels higher than l, to start after the end time of A. Although there is a way to overcome this restriction, in this paper we assume that there is no reachable state S such that a fluent f is true in S, and an action that adds f can be executed in S (the blocks world and rocket are examples of such domains).

The temporal model prevents contradictory actions from overlaping in time. For every pair of actions A and B such that A deletes the preconditions of B, the following constraints are added to the model

(3) $A^{l+1}_{st} \geq B^l_{st} + dur(B) \cdot B^l$

For every pair of actions A and B such that A deletes an add effect of B, the model includes the following inequalities

(4.1) $A^{l+1}_{st} \geq B^l_{st} + dur(B) \cdot B^l$

(4.2) $B^{l+1}_{st} \geq A^l_{st} + dur(A) \cdot A^l$

For every fluent f, the following constraints propagate the start time of f through the levels of the planning graph

(5) $f^{l+1}_{st} \geq f^l_{st}$

Similarly, for each action A, its start time is propagated through the following constraints

(6) $A^{l+1}_{st} \geq A^l_{st}$

Finally, plans start at time 1, which is stated by

(7) $f^1_{st} \geq 1$

The following theorem states that any valid Graphplan-style plan satisfying constraints (1)–(7) is temporally sound.

Theorem 1 (Soundness). *For every action A that is in a plan and satisfies the constraints above, the following holds: If p is precondition of A, then there exists an action A_p such that p is an add effect of A_p, and*

a) $st(A_p) + dur(A_p) \leq st(A)$;
b) *if B is an action that has p as a delete effect, $st(B)+dur(B) \leq st(A_p)$ or $st(A)+dur(A) \leq st(B)$.*

Furthermore, if q is an add effect of A, then

c) *for every action C in the plan that deletes q, $st(C) + dur(C) \leq st(A)$ or $st(A) + dur(A) \leq st(C)$.*

Proof: The correctness of the underlying non-temporal planning algorithm guarantees that if A is an action in the plan and p is a precondition of A, there will be another action A_p in the plan that has p as an add effect. Moreover, if l is the level of A and l' the level of A_p, it must be the case that $l' < l$, and there is no action that deletes p and occurs in any level l'' with $l'' \geq l'$ and $l'' \leq l$.
In order to prove (a), observe that because of inequality (2), $st(p^{l'+1}) \geq st(A_p^{l'}) + dur(A_p)$. Moreover, the set of inequalities (5) will enforce $st(p^l) \geq st(A_p^{l'}) + dur(A_p)$ (because $l' < l$). Finally, because of inequality (1), we have $st(A^l) \geq st(p^l)$, which together with the previous constraint give $st(A^l) \geq st(A_p^{l'}) + dur(A_p)$.
In order to prove (b), let B be any action that has p as delete effect, and let l'' be its level. As noted earlier, the correctness of the non-temporal plan implies that either $l'' < l'$ or $l'' > l$ must hold. Assume $l'' < l'$. Since B deletes an add effect of A_p, inequality (4.2) will enforce the constraint $st(A_p^{l''+1}) \geq st(B^{l''}) + dur(B)$. This constraint, together with the set of inequalities (6), impose $st(A_p^{l'}) \geq st(B^{l''}) + dur(B)$. Assume that $l'' > l$. Since B deletes a precondition of A, inequality (3) will enforce $st(B^{l+1}) \geq st(A^l) + dur(A)$ which together with inequalities (6) ensure that $st(B^{l''}) \geq st(A^l) + dur(A)$. Thus it is indeed the case that $st(B)+dur(B) \leq st(A_p)$ or $st(A)+dur(A) \leq st(B)$ is true.
In order to prove (c), assume that an action $C^{l'}$ in the plan deletes q, and that q is an add effect of A^l. Clearly $l' < l$ or $l' > l$ (because of the correctness of the non-temporal plan). If $l' < l$, since constraints (4) will enforce $st(A^{l'+1}) \geq st(C^{l'}) + dur(C)$, by constraints (6) we have $st(A^l) \geq st(C^{l'}) + dur(C)$. Similarly, if $l' > l$, since constraints (4) ensure that $st(C^{l+1}) \geq st(A^l) + dur(A)$, by constraints (6) we have $st(C^{l'}) \geq st(A^l) + dur(A)$. Therefore, it is indeed the case that $st(C) + dur(C) \leq st(A)$ or $st(A) + dur(A) \leq st(C)$ is true. □

4 Improved Temporal Model

When a planning problem is modeled as a set of linear inequalities, the number of constraints and variables that are included in the model can significantly affect the performance of the solver that is used for generating plans. The previously described model of temporal planning problems can generate a large number of constraints and variables, but it can be substantially improved if certain features of the domain structure are taken into account. The improvements that we will discuss aim at reducing the number of temporal constraints, as well as the number of temporal variables that are required to correctly model a planning domain. The reduction of the number of temporal constraints is based on the notions of *persistent fluent*, *persistent pair of fluents* and *strong* *interference* that we give for binary fluents, and that can be computed by exploiting existing domain analysis tools like DISCOPLAN [8].[3]

Definition 1 (Persistent fluent). *Let f be a binary fluent such that $f(x, *y)$ holds, and every action (instantiated operator) that has an instance of f in its add effects has another instance of f with the same first argument but different second argument in its preconditions. We say that f is* persistent *on its first argument.*

Persistence on the second argument of a fluent is defined similarly. We now define argument persistence on the first argument for a pair of binary fluents. Persistence on other arguments, or between a binary and a unary fluent can be defined in a similar way. In the following, X, Y, Z and W indicate any constant, x, y and z universally quantified variables, $\mathtt{x}$, $\mathtt{y}$ and $\mathtt{z}$ operator parameters. Moreover, we assume that no action has any literal as both precondition and add effect.

Definition 2 (Persistent pair of fluents). *Let $f1$ and $f2$ be two binary fluents such that $f1(x, *y)$, $f2(x, *z)$ and $(\mathtt{XOR}\ f1(x, y),\ f2(x, z))$ hold. We say that $f1$ and $f2$ is a* persistent pair *of fluents on their first argument, if every action that has $f1(X, Y)$ or $f2(X, Z)$ as an add effect, also has $f1(X, W)$ or $f2(X, W')$ as a precondition, where $W \neq Y, Z$ and $W' \neq Y, Z$.*

Now we can define the notion of *strong interference* between actions that will be used to improve the temporal model.[4]

Definition 3 (Strong interference). *A pair of actions A and B* strongly interfere *if*

- *f is persistent on its first argument, A has a precondition $f(X, Y)$, B has a precondition $f(X, Z)$, and either $Y \neq Z$, or $Y = Z$ and A and B have an instance of f as add effect with X as the same first argument and different second argument;* or
- *$f1, f2$ is a persistent pair of fluents on their first argument, and A has a precondition $f1(X, Y)$ and B a precondition $f2(X, Z)$, or (a) they have a common precondition $f1(X, Y)$ or $f2(X, Y)$ which they both delete and (b) they have an instance of $f1$ or $f2$ as add effect with X as the first argument and different second argument (if they are instances of the same fluent).*

For instance, under the assumptions of the previous definition, A and B strongly interfere when

- $f1(a, b)$ is a precondition of A and $f2(a, c)$ is a precondition of B, or
- when $f1(a, b)$ is a precondition of A and B, $\neg f1(a, b)$ is an effect of A and B, $f2(a, b)$ is an effect of A, and $f1(a, c)$ an effect of B.

[3] We restrict our analysis to binary fluents, which are the most common in many existing domain formalizations. Work on an extension to fluents of higher arity is in progress.

[4] For the sake of clarity, the definition is given for the case of persistence on the first argument of binary fluents, but it can be easily generalized to the cases where persistence is on the second argument, and f, $f1$, $f2$ are unary fluents.

It turns out that for pairs of actions that strongly interfere, all constraints of the form (3) and (4) can be omitted from the temporal model, because actions that strongly interfere can not overlap in time. We call such models *reduced temporal models*. An example of a reduced model is given at the end of this section. The following theorem states the soundness of the reduced temporal models.

Theorem 2. *For any two actions A and B that strongly interfere and are both included in a plan, a reduced temporal model satisfies either $st(A) \geq st(B) + dur(B)$ or $st(B) \geq st(A) + dur(A)$.*

Proof (sketch): Assume that both actions A and B are included in a plan generated by the underlying non-temporal planning algorithm, and let l be the level of A and l' the level of B. Since by Definition 3 the two actions have mutually exclusive preconditions or mutually exclusive effects, the soundness of the non-temporal algorithm implies $l' \neq l$.
We first consider the case in which $l < l'$, and $f(X, a)$ is a precondition of A and $f(X, b)$ a precondition of B, where f is a persistent fluent on its first argument, and a, b any pair of different constants.[5] Assume that A has $f(X, b)$ as an add effect. Consequently, because of constraint (2), $st(f(X, b)^{l+1}) \geq st(A^l) + dur(A)$ will hold. Then the set of constraints (5) will enforce that $st(f(X, b)^{l'}) \geq A^l_{st} + dur(A)$. Since $f(X, b)$ is a precondition of B, constraints (1) will enforce $st(B^{l'}) \geq st(f(X, b)^{l'})$ and therefore $st(B^{l'}) \geq st(A^l) + dur(A)$.
Assume now that either A does not have any add effect of the form $f(X, Y)$, or it has an add effect $f(X, Y)$ with $Y \neq b$. Then there will be a sequence of actions $A_1, ..., A_n$ in the plan, at levels $l_1, ..., l_n$ respectively, with $n > 1$, $l_i > l$ and $l_i < l'$, such that $f(X, b_i)$ is an add effect of A_i and a precondition of A_{i+1} for some constant b_i, $b_n = b$ and $l_{i+1} > l_i$. Since f persists on its first argument, if $f(X, b_i)$ is an add effect of A_i, then, for some d, $f(X, d)$ must be a precondition of A_i. Hence, it must be the case that $d = b_{i-1}$, where b_{i-1} is an add effect of action A_{i-1}. Therefore, the combination of constraints (1), (2) and (5) implies, along the sequence of actions $A_1, .., A_n$, a set of constraints $st(A^{l_i+1}_{i+1}) \geq st(A^{l_i}_i) + dur(A_i)$, for every $i \geq 1$, which implies that $st(A^{l_n}_n) \geq st(A^{l_1}_1) + \sum_{1 \leq i \leq n-1} dur(A_i)$. Then, by constraints (6) $st(B^{l'}) \geq st(A^{l_1}_1) + \sum_{1 \leq i \leq n-1} dur(A_i)$.

We now consider the temporal relation between actions A and A_1. Assume first that action A has an add effect $f(X, c)$ for some constant $c \neq b$. Then, $f(X, c)$ must be a precondition of A_1. Therefore $st(A^{l_1}_1) \geq st(A^l) + dur(A)$. Assume now that action A does not have any instance of fluent f in its add effects with X as its first argument. Then, by Definition 1 and the assumed soundness of the non-temporal plan, $f(X, a)$ must be a precondition of A_1, and A, A_1 do not strongly interfere, provided that they do not have other preconditions or effects that could cause strong interference. Moreover, note that $f(X, a)$ must be a delete effect of A_1. Since A_1 deletes a precondition of A, constraint (4) applies, and $st(A^{l_1}_1) \geq st(A^l) + dur(A)$ must hold. Hence, it is again the case that $st(A^{l_1}_1) \geq st(A^l) + dur(A)$. This constraint, together with $st(B^{l'}) \geq st(A^{l_1}_1) + \sum_{1 \leq i \leq n-1} dur(A_i)$, give us $st(B^{l'}) \geq st(A^l) + dur(A)$.

[5] For clarity the proof is given considering only persistence on first arguments; generalization to persistence on different arguments is straightforward.

Assume now that actions A and B, have both precondition $f(X,c)$ and that A adds $f(X,a)$ and B adds $f(X,b)$, for some different constants a, b and c. Then again there will be a sequence of actions $A_1,...,A_n$ in the plan, such that A_n adds $f(X,c)$, and by an analogous argument the constraints $st(A_1) \geq st(A^l) + dur(A)$ and $st(B^{l'}) \geq st(A_1^{l_1}) + \sum_{1 \leq i \leq n-1} dur(A_i)$ will hold, enforcing $st(B^{l'}) \geq st(A^l) + dur(A)$.

Now let again $l < l'$, and assume that A has a precondition $f1(X,a)$ and B has a precondition $f2(X,b)$, where $f1$ and $f2$ is a persistent pair on the first argument, and a and b is any pair of different constants. If A has $f2(X,b)$ as an add effect, then using arguments similar to the one given above we can prove that $st(B^{l'}) \geq st(A^l)+dur(A)$. Otherwise, there will be a sequence of actions $A_1,...,A_n$ in the plan, at levels $l_1,...,l_n$ respectively, with $n > 1$, $l_i > l$ and $l_i < l'$, such that $f1(X,b_i)$ or $f2(X,b_i)$ is an add effect of A_i for some constant b_i, and $f_2(X,b)$ is an add effect of A_n. By arguments similar to those used above, we can prove that $st(B^{l'}) \geq st(A_1^{l_1})+\sum_{1 \leq i \leq n-1} dur(A_i)$ and $st(A_1^{l_1}) \geq st(A^l) + dur(A)$ will hold. Thus $st(B^{l'}) \geq st(A^l) + dur(A)$ will also hold.

Finally, if $l > l'$, we can use symmetric arguments to those above to prove $st(A^l) \geq st(B^{l'}) + dur(B)$. $\Box$

We now discuss some improvements that reduce the number of temporal variables in the model of a problem. More importantly, they achieve more effective propagation of the start times of actions and fluents. We note that some of these improvements also reduce the number of constraints that are included in the model of a planning domain.

Let f be a binary fluent for which the sv-constraint $f(x,*y)$ holds. We can replace in the model all temporal variables $st(f(X,Y)^l)$ that refer to the different values of Y and same value of X with one new variable $st(fn(X)^l)$, for each level l. Similarly, if $f(*x,y)$ holds. Moreover, if $f1(x,y)$ and $f2(x,z)$ are two fluents related with a XOR constraint ($\texttt{XOR}\ f1(x,y)\ f2(x,z)$), we can replace their corresponding temporal variables referring to the different values of y and z by, but same value X for x, with a single variable $st(f12(X)^l)$. Similarly, if ($\texttt{XOR}\ f1(y,x)\ f2(z,x)$) or ($\texttt{XOR}\ f1(x,y)$ $f2(x)$) holds. We call this new set of variables *abstract temporal fluent variables*. Note that it can be the case that several different sv or XOR constraints hold for the same fluent, giving rise to different models depending on the particular abstract variables that are choosen. We handle such cases in an ad-hoc manner, but here we do not discuss this issue further.

In domains that involve operators with more than two arguments that can be instantiated by many objects, the technique of splitting action start time variables can be used. Let $A(\texttt{x},\texttt{y},\texttt{z})$ be an operator such that, for every possible value of parameter $\texttt{x}$, all actions that have different values for the pair of parameters $\texttt{y}$ and $\texttt{z}$ are mutually exclusive. We denote such an operator by $A(\texttt{x},*\texttt{y},*\texttt{z})$. Moreover, assume that all preconditions and effects of $A(\texttt{x},*\texttt{y},*\texttt{z})$ are unary or binary fluents, none of which has the pair $\texttt{y},\texttt{z}$ in its parameters. If this is the case, we can split each variable $st(A(X,Y,Z)^l)$ into two variables $st(A_1(X,Y)^l)$ and $st(A_2(X,Z)^l)$, and add the constraint $st(A_1(X,Y)^l) = st(A_2(X,Z)^l)$ to the model. In the constraints (1) and (2) of the temporal model, in which $st(A(X,Y,Z)^l)$ occurs along with a start time variable that refers to a fluent of the form $f(X)$, $f(Y)$ or $f(X,Y)$, variable $st(A(X,Y,Z)^l)$

is replaced by $st(A_1(X,Y)^l)$. Similarly, if the start time fluent variables that occurs in such a constraint is on a fluent of the form $f(X,Z)$, variable $st(A(X,Y,Z)^l)$ is replaced by $st(A_2(X,Z)^l)$. In constraints (3) and (4) one of $st(A_1(X,Y)^l)$ and $st(A_2(X,Z)^l)$ replaces variable $st(A(X,Y,Z)^l)$, depending on the fluent that gives rise to the conflict.

Let again $A(\mathtt{x}, *\mathtt{y}, *\mathtt{z})$ be an operator as defined above. It may be the case that the duration of the instances of $A(\mathtt{x}, *\mathtt{y}, *\mathtt{z})$ does not depend on all its parameters, but only on a subset of them. In the blocks world for example, it is possible that the duration of the move actions depends on the block that moves and the destination of the block, but not the origin of the block. Assume that the duration of a given operator $A(\mathtt{x}, *\mathtt{y}, *\mathtt{z})$ does not depend on the values of parameter $*\mathtt{z}$. Then, we can replace all occurences of $dur(A(X,Y,Z)) \cdot A(X,Y,Z)^l$ in constraints (1)–(4) of the temporal model by $dur(A(X,Y,Z)) \cdot \sum_z A(X,Y,z)^l$, where $\sum_z$ denoted the sum over all possible values of parameter z. If the duration of $A(\mathtt{x}, *\mathtt{y}, *\mathtt{z})$ depends only on the values of parameter x we can replace $dur(A(X,Y,Z)) \cdot A(X,Y,Z)^l$ by $dur(A(X,Y,Z)) \cdot \sum_y \sum_z A(X,y,z)^l$. We call this technique, *compact duration modeling*. In some domains this technique combined with start time variable splitting may lead to tight MIP formulations.

For instance, suppose that z can assume two values, say $Z1$ and $Z2$, and that $A(X,Y,Z1)^l$ and $A(X,Y,Z2)^l$ are mutually exclusive. When the branch and bound algorithm solves the relaxed MIP problem using the Simplex algorithm, it is very likely that some real values different from zero are assigned to $A(X,Y,Z1)^l$ and $A(X,Y,Z2)^l$ (e.g., $A(X,Y,Z1)^l = 0.5$ and $A(X,Y,Z2)^l = 0.5$). Suppose in addition that the action $B(X,Y)$ deletes an effect of both $A(X,Y,Z1)$ and $A(X,Y,Z2)$. The basic temporal model would then contain the following inequality

$$B(X,Y)^{l+1}_{st} \geq A(X,Y,Z1)^l_{st} + d \cdot A(X,Y,Z1)^l$$

where d is the duration of $A(X,Y,Z1)$. In the improved model this inequality is replaced by the following stronger constraint

$$B(X,Y)^{l+1}_{st} \geq A(X,Y,Z1)^l_{st} + d \cdot (A(X,Y,Z1)^l + A(X,Y,Z2)^l).$$

Example. Consider the `Rocket` domain with the usual ld (load) and ul (unload) actions for packages and fl (fly) for airplanes. Assume that these actions have different durations. The basic temporal model for this domain includes all constraints that have been described in the previous section. Consider for example $ld(p1, pl1, l1)$, which represents the action of loading package $p1$ to plane $pl1$ at location $l1$. The constraints of type (1) for this action are $st(ld(p1, pl1, l1)^l) \geq st(at(p1, l1)^l)$ and $st(ld(p1, pl1, l1))^l \geq st(at(pl1, l1)^l)$ for each level l. There is one constraint of type (2) for each level l, namely $st(at(p1, l1)^{l+1}) \geq st(ld(p1, pl1, l1))^l + dur(ld(p1, pl1, l1)) \cdot x^l_{ld}$, where x^l_{ld} is a 0/1 variable that takes the value 1 if the action $ld(p1, pl1, loc1)$ is included in the plan, and 0 otherwise.

The temporal overlap of actions deleting preconditions of $ld(p1, pl1, l1)$ is prohibited by constraints (3), namely, $st(ld(p1, X, l1)^{l+1}) \geq st(ld(p1, pl1, l1)^l) + dur(ld(p1, pl1, l1)) \cdot x^l_{ld}$ and $st(fly(pl1, l1, Y)^{l+1}) \geq st(ld(p1, pl1, l1)^l) + dur(ld(p1, pl1, l1)) \cdot$

x^l_{ld}, where X stands for any plane different from $pl1$ and Y for any location different from $l1$. Since $at(p1, l1)$ and $in(p1, Z)$ is a persistent pair for any plane Z, and the two load-actions of the first inequality strongly interfere (they share the precondition $at(p1, l1)$, but add different instances of in with $p1$ as first argument), by Theorem 2 the first constraint is not included in the reduced model. However, the second constraint is included, since the load and fly actions that are involved do not strongly interfere. The only action that deletes the add effect of $ld(p1, pl1, l1)$ is "blocked" through a pair of constraints of type (4), which are $st(ul(p1, pl1, l1)^{l+1}) \geq st(ld(p1, pl1, l1))^l) + dur(ld(p1, pl1, l1)) \cdot x^l_{ld}$ and its symmetric. Note that by Theorem 2 these constraints are not included in the reduced model, because of the persistent pair $in(p1, pl1)$, $at(p1, l1)$ appearing in the preconditions of the two interfering actions.

Since $at(x, *y)$ holds for any plane x and location y, all occurrences of $st(at(p1, l1)^l)$ can be replaced by the variable $st(at1(p1)^l)$. Moreover, since (XOR $at(x, y)$ $in(x, y)$) holds, $st(at(p1, X)^l)$ and $st(in(p1, Y)^l)$ can be replaced by $st(atin(p1)^l)$.

5 Solving Planning Problems

When considered alone, the temporal model that we have described could find feasible start times for the actions of a plan that is produced by any algorithm solving planning graphs. This STRIPS planner would ignore completely the duration of the actions, and the temporal part would not need to know how the planner generates the plans. The two parts would be "glued" together through the 0/1 action variables that are shared by the two parts.

This separation of the logical and the temporal part of a planning problem facilitates the use of a different algorithm for each of these parts, e.g., propositional satisfiability for the first and linear programming for the second, in an architecture similar to LPSAT [17]. However, in the approach taken here we represent *both* parts by a set of linear inequalities and use standard branch-and-bound on the *union* of the two parts. The potential benefit of such a unified algorithmic framework is the possibility of exploiting the strong interaction between the two parts, which may lead to extensive value propagation.

For the formulation of a logical part of a temporal planning problem as a set of linear inequalities, we use the method developed by [15] and improved by [3]. This approach essentially translates the planning graph of a STRIPS problem into an Integer Programming model, and then uses branch-and-bound for solution extraction.
The role of the logical part is to order actions in such a way that dependencies and interactions between them are respected, and consistent states are generated. If an action appears at some level i of the planning graph, this means that it can be executed only after the completion of the execution of some other actions at lower levels of the graph.

The overall duration of the plan is represented by the variable mks (for makespan) and a set of constraints of the form $(a^l_{st} + dur(a) \cdot a^l \leq mks)$, for every action a and level l. The objective function that is used in the problem formulation depends on the optimization objective. If the objective is the minimization of the makespan of the plan the objective function is $min(mks)$. If there is a deadline for the overall execution of the plan, mks is constrained to be less than or equal to this value, and in the objective function any other quantity can be used. For instance, if each action a_i has an associated cost c_i the overall cost of the plan is minimized through the objective function $min(\sum_{a_i^l} c_i \cdot a_i^l)$.

The algorithm starts with the encoding of the planning graph of length 1 (i.e, with one level), and then it extends the encoding by increasing the number of levels in the underlying graph, until a feasible solution for both the logical and the temporal parts of the encoding is found. Let l be the level of the first solution, and let opt^l denote the value of the optimal solution for that level, under the optimization objective. After the problem for the l levels is solved to optimality, the encoding is extended by considering an extra level of the underlying graph, and a new search starts for a solution with an objective function value less than opt^l.[6] If a new, improved, solution is found, the procedure repeats by extending again the encoding (i.e, by considering an extra level for the underlying planning graph). If at some level no better solution is found, the algorithm terminates. Of course, the solutions found are not guaranteed to be optimal, as it is possible that better solutions can be extracted if the underlying planning graph is extended further.

An improvement of the incremental process is to estimate a lower bound κ on the size of the underlying planning graph, that is likely to be higher than one. κ can then be used to generate the initial MIP encoding, and this could same a significant amount of time because there is no need to run the MIP solver for determining that the encodings of size less than κ do not admit a feasible solution. An obvious possibility for estimating the value of κ is to construct the planning graph using a Graphplan-style algorithm and set κ to the level (if any) were all goals of the problem are reached and are not mutually exclusive. However, in the experimental results that we present in the next section we have not exploited this possibility (the initial size of the encoding in terms of underlying graph levels is one).

6 Experimental Results

We ran some initial experiments with the new temporal planning approach. The models were generated by hand, using the algebraic modeling system `PLAM` (ProLog and Algebraic Modeling) [2], and solved by `CPLEX` 7.1 with the following parameters. Dual Simplex was used with steepest-edge pricing, the variable selection strategy was set to pseudo reduced costs, while probing was set to 1 or 2 depending on the domain. In order to gain some insight about the difficulty of the problems, and the quality of the solution that are generated by the new method, the makespan minimization problem were also solved with `TP4` and `TGP`. `TP4` and `CPLEX` were run on a Sun Ultra-250 with 1 GB RAM, and an UltraSparcII 400MHz processor. `TGP` was run on a Pentium 500MHz machine running under Linux. Table 1 presents some of the experimental results. The `bw` rows refer to blocks world problems, and the `r` rows refer to `rocket` problems. All blocks world problems are instances with 8 blocks and the duration of each action depends on which block is moved and where. The `rocket` problems involve 4

[6] When we extend the encoding, both the logical and the temporal parts of encoding are extended.

	TP4	TGP	MIP-full		MIP-reduced	
Problem	t/d/a	t/d/a	t/d/a/l	tt	tf	tt
bw1	3/11/11	-/10/-	37/11/11/5	817	1563	–
bw2	51/11/14	-/9/-	32/11/12/5	653	2045	–
bw3	423/8/10	1231/8/9	9/8/9/4	276	1412	–
bw4	-/13/-	-/12/-	65/14/13/6	–	–	–
r1	288/14/30	140/14/24	87/14/24/7	2064	218	–
r2	-/14/-	3306/15/28	94/15/28/7	2840	212	–
r3	6252/9/36	5692/9/29	969/10/29/6	–	2540	–
r4	-/9/-	-/9/-	415/11/34/6	–	2530	–

Table 1. TP4, TGP and MIP on makespan minimization problems.

locations, 2 or 3 planes, and 9 to 11 packages. The duration of a fly-action depends on which airplane is used and on which are the cities connected by the flight; the duration of a load/unload-action depends on which airplane is used by the action. All run times reported are in seconds. A time limit of 7200 seconds was used for all systems.

For TP4 and TGP, the entries t/d/a in the table are respectively the run time (t), plan duration (d) and number of actions (a) of the generated plan. A -/d/- entry indicates that the system was searching for a solution of duration d, when the time limit was reached, and execution was aborted.

The data for the MIP method are presented under the MIP-full column in the t/d/a/l format, with the following meaning. The number in position l is the first planning graph level at which a feasible solution is found. The data t/d/a refer to the optimal solution of the problem that corresponds to graph length l. The duration of this optimal solution is presented in d, and the number of actions in the solution in a. The number in position t is the run time, and includes both the time needed for proving the infeasibility of the problems associated with planning graphs of length less than l, as well as finding the optimal solution for the graph of length l. The last column in the table, labeled with tt, presents the overall run time needed for solving the problem on all different levels, up to the level where the solution does not improve further. A dash "–" in this column denotes that CPLEX reached the time limit before completing the search of the last level.

We note that in all problems, except bw4, the best solution that was found by the method, was at the same planning graph level with the first feasible solution. In problem bw4, the first solution was found at level 6. The optimal solution for this level is 14, the graph was expanded, and a better solution with duration 13 was found after 3032 seconds. Regarding the rocket problems, the first MIP solutions for r1 and r2 are optimal solutions, while the first solutions for the other problems are good quality solution but are not the best ones. However, in all cases except the easiest problem the first MIP solutions are computed more efficiently than the optimal solutions of TP4 and TGP (MIP was up to two orders of magnitude faster).

While column MIP-full refers to the fully optimized MIP models, column MIP-reduced presents experimental results with the reduced temporal fluent models that do not incorporate any of the additional improvements described in section 4, such as abstract fluent variables, start time variable splitting and compact duration modeling. The `tf` entries are the run times for solving the makespan minimization problem for the first level of the planning graph where a feasible solution exists (they do not include the time needed for proving the infeasibility of planning graphs with fewer levels). The `tt` entries have the same meaning as in `MIP-full`. Note that the additional improvements lead to substantial speed-up in the computation, especially in the case of the blocks world problems. In this domain, these improvements reduce the number of temporal variables, but also cause a dramatic reduction of the number of constraints. Indeed, it seems that in the blocks world domain, the inefficiency of the reduced model is mainly due to the large number of constraints that make extremely hard the problem of solving the linear relaxation at each node of the branch and bound tree. The basic model contains even more constraints that further degrade the performance of the method, making it impractical for domains such as the blocks world. Therefore, we do not present experimental results for the basic model.

It is important to note that the improvements to the reduced temporal models also lead to tighter MIP formulations. For instance, the solution of the initial relaxation (at the root node of the branch and bound tree) of problem `bw1` (`r1`) with the reduced model is 5.399 (1.898), while the solution of the same problem with the additional improvements is 6.647 (2.535).

Table 2 presents experimental results on planning problems with deadlines and different minimization objectives. The columns under "action min." correspond to planning problems where the number of actions is minimized. The deadline for each of these problems was set to the minimum makespan as it is computed in the experiments that are reported in Table 1. Data are given for the first and the best solution found by MIP, in the `t/a/l` format. The run time is presented by `t`, the number of actions in the plan by `a`, and the planning graph length by `l`. A dash "—" in the optimal solution column denotes that the first solution that was found was also the best.

It appears that the problem of generating plans with minimum number of actions (even with tight deadlines as those used in our experiments) can be solved more effectively than that of finding plans that minimize makespan. Moreover, as one might expect, with the new objective function, MIP generates plans that contain fewer actions than those generated when minimizing makespan.

The second set of experiments that are reported in Table 2 appear under "action and makespan min.", and corresponds to optimization problems where the objective function is $min(\sum_{a_i^l} a_i^l + 2 \cdot mks)$. This is a simple example of optimization with multiple criteria that can be competing, since makespan minimization may introduce more actions in the generated plans. The particular objective function that has been used in these experiments, assigns slightly higher preference to plan duration minimization through the higher weight of variable mks. A typical example that shows the effects of competing optimization criteria is `bw4` in Table 2, where the decrease in plan duration from 14 to 13, comes with an increase in the number of actions from 12 to 13.
As in the previous experiments, a deadline is used for each problem, with the differ-

	action min.			action and makespan min.		
	First	Best		First	Best	
Problem	`t/a/l`	`t/a/l`	`tt`	`t/d/a/l`	`t/d/a/l`	`tt`
`bw1`	15/10/5	—	103	18/11/10/5	—	388
`bw2`	14/10/5	—	87	18/11/10/5	—	277
`bw3`	5/8/4	—	25	12/8/8/4	—	222
`bw4`	310/13/7	740/12/8	1555	31/14/12/6	974/13/13/7	-
`r1`	35/24/7	—	136	49/14/24/7	—	877
`r2`	53/28/7	—	215	81/15/28/7	—	1027
`r3`	138/29/6	1736/27/7	-	375/10/29/6	2475/10/27/7	-
`r4`	215/32/6	1903/29/7	-	330/11/32/6	—	-

Table 2. MIP on action minimization and combined action and makespan minimization problems with deadlines.

ence that this deadline is now two time units higher[7] than the makespan computed in the makespan minimization experiments. The experimental results are presented in the `t/d/a/l` format, with the same interpretation as in Table 1. Note that CPLEX performs better on these combined action and makespan optimization problems than on optimizing makespan alone. Therefore, it seems that action minimization provides strong guidance in the search for good solutions.

It appears that the new MIP temporal planning method performs well in providing good solutions early in the computation, regardless of the optimization objective that is used. Of course there can be cases, such as problem `r3` in Table 1, where the method fails to find the optimal solution within a reasonable time. Nevertheless, in all problems considered, it quickly found high quality solutions.

7 Conclusions and Current Work

We have presented a novel approach to temporal planning that relies on an explicit representation of the temporal constraints present in a planning problem. We have showed how the structure of a domain can be exploited in the MIP formulations, and presented some encouraging preliminary results from an experimental analysis of the new method.

Currently we are extending our experiments to consider several additional domains, and compare our method to other recent planners that may also generate non-optimal solution such as SAPA [4] and LPG [9]. At the same time we are investigating ways of improving our method. One promising direction is to exploit further the interaction between the logical and temporal part of a planning problem by relaxing the former and tightening the latter. Another direction concerns modeling more expressive planning languages capable, for instance, of dealing with level 3 of PDDL 2.1, the language of the AIPS-2002 planning competition [6]. Features of such languages include actions

[7] The value 2 that has been selected, seems on the one hand to provide a fairly tight deadline for the generated plans, and on the other hand to allow for further improvement of the makespan of the plans. However, it is only indicative, and other values could have been used.

with preconditions/effects involving numerical quantities, resources, and temporal conditions that are required to hold at some point during the execution of an action, at its beginning or at its end. It appears that all such features can be accommodated by a simple extension of our model.

Finally, we are investigating the use of domain knowledge in the form of valid linear inequalities, which in our approach can be easily incorporated into the representation and used by the MIP solver.

References

1. J. Allen. Temporal reasoning and planning. In J. Allen, H. Kautz, R. Pelavin, and J. Tenenberg, editors, *Reasoning about Plans*, San Mateo, CA, 1991. Morgan Kaufmann.
2. P. Barth and A. Bockmayr. Modelling discrete optimisation problems in constraint logic programming. *Annals of Operations Research*, 81, 1998.
3. Y. Dimopoulos. Improved integer programming models and heuristic search for AI planning. In *Proceedings of ECP-01*. Springer Verlag, 2001.
4. M. B. Do and S. Kambhampati. Sapa: A domain-independent heuristic metric temporal planner. In *Proceedings of ECP-01*. Springer Verlag, 2001.
5. M. Fox and D. Long. The automatic inference of state invariants in TIM. *JAIR*, 9:367–421, 1998.
6. M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Technical report, University of Durham, UK, 2001.
7. A. Gerevini and L. Schubert. Inferring state constraints for domain-independent planning. In *Proceedings of AAAI-98*, 1998.
8. A. Gerevini and L. Schubert. Discovering state constraints in DISCOPLAN: Some new results. In *Proceedings of AAAI-00*, 2000.
9. A. Gerevini and I. Serina. LPG: a planner based on planning graphs with action costs. In *Proceedings of AIPS'02*, 2002.
10. P. Haslum and H. Geffner. Heuristic planning with time and resources. In *Proceedings of ECP-01*. Springer Verlag, 2001.
11. H. Kautz and J. Walser. State-space planning by integer optimization. In *Proceedings of AAAI-99*, 1999.
12. J. Penberthy and D. Weld. Temporal planning with continuous change. In *Proceedings of AAAI-94*, pages 1010–1015, Seattle, WA, 1994. Morgan Kaufmann.
13. D. Smith and D. Weld. Temporal planning with mutual exclusive reasoning. In *Proceedings of IJCAI-99*, 1999.
14. E. Tsang. Plan generation in a temporal framework. In *Proceedings of ECAI-96*, pages 479–493, 1986.
15. T. Vossen, M. Ball, A. Lotem, and D. Nau. On the use of integer programming models in AI planning. In *Proceedings of IJCAI-99*, 1999.
16. D. Weld. Recent advances in AI planning. *AI Magazine*, 20(2), 1999.
17. S. Wolfman and D. Weld. The LPSAT engine and its application to resource planning. In *Proceedings of IJCAI-99*, 1999.
18. L. Wolsey. *Integer Programming*. John Wiley and Sons, 1998.