Planning Graph-based Heuristics for Cost-sensitive Temporal Planning

Minh B. Do & Subbarao Kambhampati [*]
Department of Computer Science and Engineering
Arizona State University, Tempe AZ 85287-5406
{binhminh,rao}@asu.edu

Abstract

Real world planners need to be sensitive to the quality of the plans they generate. Unlike classical planning where quality is often synonymous with plans having least number of actions, in temporal planning plan quality is multi-dimensional. It involves both temporal aspects of the plan (such as makespan, slack, tardiness) and execution cost aspects (such as cumulative action cost, resource consumption). Until now, most domain-independent temporal planners have concentrated solely on the former, ignoring the latter. In this paper, we consider the problem of developing heuristics that are sensitive to both makespan and cost, and develop a planning graph-based approach for this purpose. Our approach involves augmenting a (temporal) planning graph data structure with a mechanism to track the execution cost of the goals and subgoals. Since the cost of achieving a goal is dependent on the amount of available time, we need to track the cost of a literal as a *function* of time. We present a methodology for efficiently tracking the cost functions, and discuss how they can be used as the basis for deriving heuristics to support any objective function based on makespan and execution cost. We demonstrate the effectiveness of this general method for deriving cost- and makespan-sensitive heuristics in the context of *Sapa* a forward chaining planner for metric temporal domains that we have been developing. A version of *Sapa* using a subset of the techniques discussed in this paper was one of the best domain independent planners for domains with metric and temporal constraints in the third International Planning Competition, held at AIPS-02.

Introduction

Of late, there has been increased interest in the planning community to leverage the successes in heuristic control of classical planners to tackle the more realistic metric temporal planning problems. Developing heuristics for metric temporal planning is complicated by the **multi-objective** nature of the problem. In contrast to classical planning, where the heuristics need only be sensitive to the "length" of the plans, in metric temporal planning, the user may be interested in improving either temporal quality of the plan (e.g. makespan) or its cost (e.g. cumulative action cost, cost of resources consumed etc.), or more generally, a combination there of.[1] Consequently, effective plan synthesis requires heuristics that are able to track both these aspects of an evolving plan.

Until now, most domain-independent temporal planners have concentrated solely on optimizing makespan (c.f. (Haslum & Geffner 2001; Smith & Weld 1999; Do & Kambhampati 2001)), a temporal aspect of plan quality. Consequently, most heuristics for temporal planners are only sensitive to temporal aspects (more specifically, just to makespan). We are interested in developing heuristics that are sensitive to both temporal and cost aspects of plan quality, so that we can effectively handle the multi-objective nature of temporal planning. An important challenge here, as illustrated by the example below, is that the cost and temporal aspects of a plan are often inter-dependent:

Example: Suppose we need to go from Tucson to Los Angeles. The two common options are: (1) rent a car and drive from Tucson to Los Angeles in one day for \$100 or (2) take a shuttle to the Phoenix airport and fly to Los Angeles in 3 hours for \$200. The first option takes more time (higher makespan) but less money, while the second one clearly takes less time but is more expensive. Depending on the specific weights the user gives to each criterion, she may prefer the first option over the second or vice versa. Moreover, the user's decision may also be influenced by other constraints on time and cost that are imposed over the final plan. For example, if she needs to be in Los Angeles in six hours, then she may be forced to choose the second option. However, if she has plenty of time but limited budget, then she may choose the first option.

The simple example above shows that makespan and execution cost, while nominally independent of each other, are nevertheless related in terms of the overall objective of the user and the constraints on a given planning problem. More specifically, for a given makespan threshold (such as to be in LA within six hours), there is a certain estimated solution cost tied to it (shuttle fee and ticket price to LA) and vice versa. Thus, in order to find plans that are good with respect

[*]We thank David E. Smith and the AIPS reviewers for useful comments on the earlier drafts of this paper. This research is supported in part by the NSF grant IRI-9801676, and the NASA grants NAG2-1461 and NCC-1225.

[1]Another dimension of optimization involves execution flexibility (e.g. slack, latency etc.). In the current paper, we ignore this dimension and concentrate on cost and make-span tradeoffs.

to both cost and makespan, we need heuristics that track cost of a set of (sub)goals as a function of time.

Since the cost of achieving a goal is dependent on the amount of available time (see the example above), we introduce an approach to track the cost of a literal as a *function* of time. (Figure 3, to be discussed later, shows the cost functions for subgoals in an extended version of the travel example). Specifically, the cost incurred to achieve facts and to execute actions are estimated by cost propagation while building the temporal planning graph. These cost functions can in turn be used to estimate the achievement cost of a set of goals for a given makespan bound and *vice versa*. We present the methodology for efficiently maintaining the cost functions, and discuss how these time-sensitive cost functions can be used as the basis for deriving heuristics to support any objective function based on makespan and execution cost. Finally, we empirically demonstrate the effectiveness of our heuristics in generating plans that offer a variety of cost-makespan tradeoffs. Our experiments are done with *Sapa*, a forward chaining planner for metric temporal domains that we have been developing (Do & Kambhampati 2001).

A version of *Sapa* using a subset of the techniques discussed in this paper was one of the best domain independent planners for domains with metric and temporal constraints in the third International Planning Competition, held at AIPS-02. In fact, it is the best planner in terms of solution quality and number of problems solved in the highest level of PDDL2.1 setting used in the competition for the two domains *Satellite* and *Rovers*, which are inspired by real-world applications being investigated by NASA.

The paper is organized as follows: first we describe the temporal planning problem and the action representation that we assume. Next, we discuss the problem of how to build a temporal planning graph and use it to propagate the cost information. The next two sections show how the propagated information can be used to estimate the cost of achieving the goals from a given state. Then, we discuss how the mutual exclusion relations can help to improve the heuristic estimation. We continue with sections on empirical results of using our heuristics in *Sapa*. We conclude the paper with a discussion on related work, the conclusion and the future work.

Action Representation

This section provides the background on the action representation and different types of constraints in the the temporal planning problems. Unlike actions in classical planning, in planning problems with temporal and resource constraints, actions are not instantaneous but have durations. Their preconditions may either be instantaneous or durative and their effects may occur at any time point during their execution. Each action *A* has a duration D_A, starting time S_A, and end time ($E_A = S_A + D_A$). The value of D_A can be statically defined for a domain, statically defined for a particular planning problem, or can be dynamically decided at the time of execution. Action *A* has preconditions $Pre(A)$ that may be required either to be instantaneously true at the time point S_A, or required to be true starting at S_A and remain true for some duration $d \leq D_A$. The logical effects *Eff(A)* of A are instantaneous and occur at time points $S_A + d$ ($0 \leq d \leq D_A$). If $d > 0$ then they are called delayed effects as their onset is delayed with respect to action start time. Actions can also consume or produce metric resources and their preconditions may also well depend on the value of the corresponding resource. Each action is associated with the cost value, which represents the total money we need to spend to execute that action.

We shall now illustrate the action representation in a simple temporal planning problem. This problem, which is an extended version of the example we introduced in the introduction, will be used as the running example through out the rest of the paper. Figure 1 shows graphically the problem description. In this problem, a group of students in Tucson need to go to Los Angeles (LA). There are two car rental companies in Tucson. If the students rent a car from the first company, which has faster but more expensive cars (*Car1*) , they can only go to Phoenix (PHX) or Las Vegas (LV). However, if they decide to rent a car from the second company (*Car2*), which is slower but cheaper, then they can use it to drive to Phoenix or directly to LA. Moreover, to reach LA, the students can also take a *train* from LV or a flight from PHX. In total, there are 6 actions in the domain: *drive-car1-tucson-phoenix* ($D^{c_1}_{t \rightarrow p}$), *drive-car1-tucson-lv* ($D^{c_1}_{t \rightarrow lv}$), *drive-car2-tucson-phoenix* ($D^{c_2}_{t \rightarrow p}$), *drive-car2-tucson-la* ($D^{c_2}_{t \rightarrow la}$), *fly-airplane-phoenix-la* ($F_{p \rightarrow la}$), and *use-train-lv-la* ($T_{lv \rightarrow la}$). Each move (by car/airplane/train) action A between two cities X and Y requires the precondition that the students should be at X ($at(X)$)at the beginning of A. There are also two temporal effects: $\neg at(X)$ occurs at the starting time point of A and $at(Y)$ at the end time point of A. The durations and execution cost values for six actions described above are shown in the right side of Figure 1. In this case, the costs of moving by train or airplane are the respective ticket prices, and the costs of moving by rental cars include the rental fees and gas (resource) costs.

Propagating cost information

To measure the heuristic distance of a given state to the goals, we need to estimate how costly it is to achieve the goals from that state. All we know is that facts in the initial states have *zero* costs and that each action has some execution cost. Thus, to evaluate the cost of a set of goals from a given state, we need to propagate the costs from the initial state to the goals using the mutual dependencies between facts and actions. Specifically, the cost to achieve a fact depends on the cost to execute the actions supporting it, which in turn depends on the costs to achieve facts that are their preconditions. Given that the planning graph is an excellent structure to represent the relation between facts and actions, we will use the temporal planning graph structure (TGP(Smith & Weld 1999)) as a substrate for propagating the costs information.

In this section, we start with a brief discussion of the data structures used for the cost propagation process. We then continue with the details of the propagation process and the

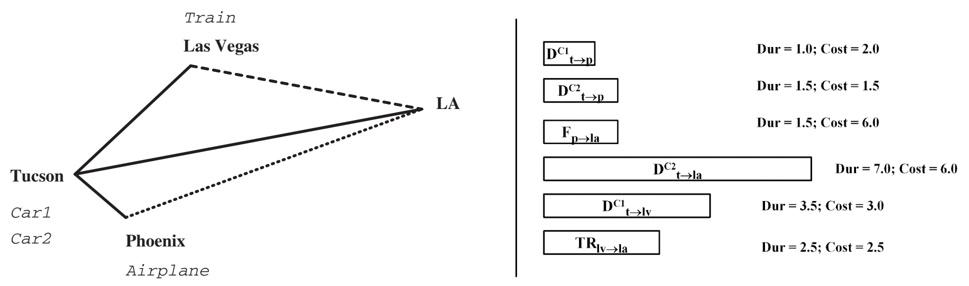

Figure 1: The travel example

criteria used to terminate the propagation.

The Temporal Planning Graph Structure

The temporal planning graph G is a bi-level graph, with one level containing all *facts*, and the other containing all *actions* in the planning problem. Each fact links to all actions supporting it, and each action links to all facts that belong to its precondition and effect lists.[2] As we will see in more detail in the later part of this section, we build the temporal planning graph by incrementally increasing the time (makespan value) of the graph. At a given time point t, an action A is activated if all preconditions of A can be achieved at t. To support the *delayed effects* of the activated actions (i.e., effects that occur at the *future* time points beyond t), we also maintain an event queue $Q = \{e_1, e_2, ...e_n\}$ sorted in the increasing order of event time. Each event is a 4-tuple $e = \langle f, t, c, A\rangle$ in which: (1) f is the fact that e will add; (2) t is the time point at which the event will occur; and (3) c is the cost incurred to enable the execution of action A which causes e. For each action A, we introduce a cost function $C(A, t) = v$ to specify the estimated cost v that we incur to enable A's execution at time point t. In other words, $C(A, t)$ is the estimate of the cost incurred to achieve all of A's preconditions at time point t. Moreover, each action will also have an *execution cost* ($C_{exec}(A)$), which is the cost incurred in executing A (e.g ticket price for *fly* action, gas cost for driving a car). For each fact f, a similar cost function $C(f, t) = v$ specifies the estimated cost v incurred to achieve f at time point t.

In the original Graphplan algorithm(Blum & Furst 1995), there is a process of propagation of mutex information, which captures the negative interactions between different propositions and actions occur at the same time (level). To simplify the discussion, in this paper we will neglect the mutex propagation and will discuss the propagation process in the context of the relaxed problem in which the *delete effects* of actions, which cause the mutex relations, are ignored. Later, we will discuss how the temporal mutex relation such as the ones discussed in TGP(Smith & Weld 1999) can be used to improve the cost propagation and heuristic estimation processes.

[2]The bi-level representation has been used in the classical planning to save time and space, but as Smith & Weld(Smith & Weld 1999) showed, it makes even more sense in the temporal planning domains because there is actually no notion of level. All we have are a set of facts/action nodes, each one encoding information such as the *earliest time point* at which the fact/action can be achieved/executed, and the *lowest cost* incurred to achieve them etc.

```
Function Propagate Cost
  Current time: $t_c = 0$;
  $Apply(A_{init}, 0)$;
  while Termination-Criteria $\neq$ true
    Get earliest event $e = \langle f_e, t_e, c_e, A_e\rangle$ from $Q$;
    $t_c = t_e$;
    if $c_e < C(f, t_c)$ then
    Update: $C(f, t) = c_e$
      for all action $A$: $f \in Precondition(A)$
        $NewCost_A = CostAggregate(A, t_c)$;
        if $NewCost_A < C(A, t_c)$ then
          Update: $C(A, t) = NewCost(A), t_c \leq t < \infty$;
          $Apply(A, t_c)$;
End Propagate Cost;

Function $Apply(A, t)$
    For all A's effect that add $f$ at $S_A + d$ do
          $Q = Q \bigcup \{e = \langle f, t+d, C(A, t) + C_{exec}(A), A\rangle\}$;
End $Apply(A, t)$;
```

Figure 2: Main cost propagation algorithm

Cost propagation procedure

As mentioned above, our general approach is to propagate the estimated costs incurred to achieve facts and actions from the initial state. As a first step, we need to initialize the cost functions $C(A, t)$ and $C(f, t)$ for all facts and actions. For a given initial state S_{init}, let $F = \{f_1, f_2...f_n\}$ be the set of facts that are true at time point t_{init} and $\{(f'_1, t_1), ...(f'_m, t_m)\}$, be a set of outstanding positive events which specify the addition of facts f'_i at time points $t_i > t_{init}$. We introduce a dummy action A_{init} to represent S_{init} where A_{init} (1) requires no preconditions; (2) has cost $C_{exec}(A_{init}) = 0$ and (3) causes the events of adding all f_i at t_{init} and f'_i at time points t_i. At the beginning ($t = 0$), the event queue Q is empty, the cost functions for all facts and actions are initialized as: $C(A, t) = \infty, C(f, t) = \infty$, $\forall 0 \leq t < \infty$, and A_{init} is the only action that is applicable.

Figure 2 summarizes the steps in the cost propagation algorithm. The main algorithm contains two interleaving parts: one for applying an action and the other for activating an event representing the action's effect.

Applying an action: When an action A is applied, we (1) augment the event queue Q with events corresponding to all of A's effects, and (2) update the cost function $C(A, t)$ of A.

Activating an event: When an event $e = \langle f_e, t_e, C_e, A_e \rangle$, which represents an effect of A_e occurring at time point t_e and add fact f_e with cost C_e is activated, the cost function of the fact f_e is updated if $C_e < C(f_e, t_e)$. Moreover, if the newly improved cost of f_e leads to a reduction in the cost functions of an action A that f_e supports (decided by function $CostAggregate(A, t)$ in line 11 of Figure 2) then we will *(re)apply* A to propagate f_e's new cost of achievement to the cost functions of A and its effects.

At any given time point t, $C(A, t)$ is an aggregated cost (returned by function $CostAggregate(A, t)$) to achieve all of its preconditions. The aggregation can be done in different ways:

1. **Max-propagation:**
   $C(A, t) = Max\{C(f, t) : f \in Precond(A)\}$ or
2. **Sum-propagation:**
   $C(A, t) = \sum\{C(f, t) : f \in Precond(A)\}$ or
3. **Combo:**
   $C(A, t) = 0.5 * (Max\{C(f, t) : f \in Precond(A)\})$
   $+ 0.5 * (\sum\{C(f, t) : f \in Precond(A)\})$

The first method assumes that all preconditions of an action depend on each other and the cost to achieve all of them is equal to the cost to achieve the costliest one. This rule leads to the underestimation of $C(A, t)$ and the value of $C(A, t)$ is admissible. The second method (*sum-propagation*) assumes that all facts are independent. Although clearly inadmissible, it has been shown (c.f.(Nguyen, Kambhampati, & Nigenda 2001; Bonet, Loerincs, & Geffner 1997)) to be more effective than the *max-propagation*. The last method combines the two and basically tries to account for the dependency between different facts in the *sum-propagation*.

When the cost function of one of the preconditions of a given action is updated, the $CostAggregate(A, t)$ function is called and it uses one of the methods described above to calculate if the cost required to execute an action has improved (reduced).[3] If $C(A, t)$ has improved, then we will *re-apply* A (line 12-14 in Figure 2) to propagate the improved cost to the cost functions of its effects.[4]

Finally, the only remaining issue in the main algorithm illustrated in Figure 2 is the *termination criteria* for the propagation, which will be discussed in detail in the next section.

[3]Propagation rule (2) and (3) will guarantee a lower cost of $C(A, t)$ when the cost function one of A's precondition is improved. However, it's not true for rule (1).

[4]Notice that because we increase time in *step* jump by going through events in the event queue, the cost functions for all facts and actions will appear as *step-functions* (even though times are measured continuously.)

Coming back to our running example, the left side of Figure 3 shows graphically the time points at which each action can be applied ($C(A, t) < \infty$) and the right side shows how the cost function of facts/actions change as the time increases. Here is an outline of the update process in this example: at time point $t = 0$, four actions can be applied. They are $D^{c_1}_{t\to p}$, $D^{c_2}_{t\to p}$, $D^{c_1}_{t\to lv}$, $D^{c_2}_{t\to la}$. These actions add 4 events into the event queue $Q = \{e_1 = \langle at\_phx, t = 1.0, c = 2.0, D^{c_1}_{t\to p}\rangle$, $e_2 = \langle at\_phx, 1.5, 1.5, D^{c_2}_{t\to p}\rangle$, $e_3 = \langle at\_lv, 3.5, 3.0, D^{c_1}_{t\to lv}\rangle$, $e_4 = \langle at\_la, 7.0, 6.0, D^{c_2}_{t\to la}\rangle\}$. After we advance the time to $t = 1.0$, the first event e_1 is activated and $C(at\_phx, t)$ is updated. Moreover, because at_phx is a precondition of $F_{p\to la}$, we also update $C(F_{p\to la}, t)$ at $t_e = 1.0$ from ∞ to 2.0 and put an event $e = \langle at\_la, 2.5, 8.0, F_{p\to la}\rangle$, which represents $F_{p\to la}$'s effect, into Q. We then go on with the second event $\langle at\_phx, 1.5, 1.5, D^{c_2}_{t\to p}\rangle$ and lower the cost of the fact at_phx and action $F_{p\to la}$. Event $e = \langle at\_la, 3.0, 7.5, F_{p\to la}\rangle$ is added as a result of the newly improved cost of $F_{p\to la}$. Continuing the process, we update the cost function of at_la once at time point $t = 2.5$, and again at $t = 3.0$ as the delayed effects of actions $F_{p\to la}$ occur. At time point $t = 3.5$, we update the cost value of at_lv and action $T_{lv\to la}$ and introduce the event $e = \langle at\_la, 6.0, 5.5, T_{lv\to la}\rangle$. Notice that the final event $e' = \langle at\_la, 7.0, 6.0, D^{c_2}_{t\to la}\rangle$ representing a delayed effect of action $D^{c_2}_{t\to la}$ applied at $t = 0$ will not cause any update. This is because the cost function of at_la has been updated to value $c = 5.5 < c_{e'}$ at time $t = 6.0 < t_{e'} = 7.0$.

Beside the values of the cost functions, Figure 3 also shows the supporting actions ($SA(f, t)$) for the fact (goal) at_la. We can see that action $T_{lv\to la}$ gives the best cost of $C(at\_la, t) = 5.5$ for $t \geq 6.0$ and action $F_{p\to la}$ gives best cost $C(at\_la, t) = 7.5$ for $3.0 \leq t < 5.5$ and $C(at\_la, t) = 8.0$ for $2.5 \leq t < 3.0$. The right most graph in Figure 3 shows similar cost functions for the actions in this example. We only show the cost functions of actions $T_{lv\to la}$ and $F_{p\to la}$ because the other four actions are already applicable at time point $t_{init} = 0$ and thus their cost functions stabilize at 0.

Termination criteria for the cost propagation process

In this section, we discuss the issue of when we should terminate the cost propagation process. The first intuition is that we should not stop the propagation when there exists top level goals for which the cost of achievement is still ∞ (unreached goal). On the other hand, given our objective function of finding the cheapest way to achieve the goals, we need not continue the propagation when there is no chance that we can improve the cost of achieving the goals. From those intuitions, following are several rules that constraint the termination:

Deadline termination: *The propagation should stop at time point t if: (1) $\forall$ goal G : $Deadline(G) \leq t$, or (2) $\exists$ goal G : $(Deadline(G) < t) \wedge (C(G, t) = \infty)$.*

The first rule governs the hard constraints on the goal deadlines, it implies that we should not propagate beyond

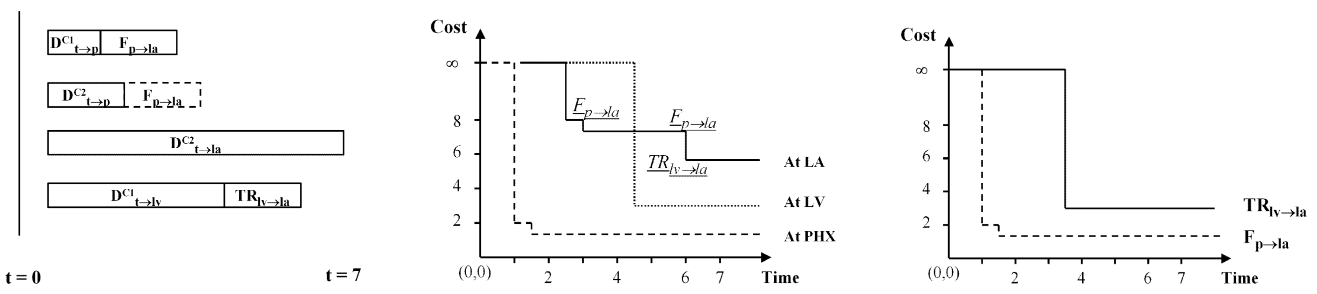

Figure 3: Cost functions for facts and actions in the travel example.

the latest goal deadline (because any cost estimation beyond that point is useless), or we can not achieve some goal by its deadline.

With the observation that the propagated costs can change only if we still have some events left in the queue that can possibly change the cost functions of a specific propositions, we have the second general rule regarding the propagation:

Fix-point termination: *The propagation should stop when there are no more event that can decrease the cost of any proposition.*

The second rule is a qualification for reaching the fix-point in which there is no gain on the cost function of any fact or action. It can be thought of as analogous to the idea of growing the planning graph until it *levels-off* in the classical planning.

Stopping the propagation according to the two general rules above leads us to the best (lowest value) achievable cost estimation for all propositions given a specific initial state. However, there are situations in which we may want to stop the propagation process earlier due to several reasons. First, propagation until the fix-point where there is no gain on the cost function of any fact or action may be costly.[5] Second, the cost functions of the goals may reach the fix-point long before the full propagation process is terminated according to the general rules discussed above, where the costs of *all* propositions and actions stabilize.

Given the above motivations, we introduce several different criteria to stop the propagation earlier than is entailed by the fix-point computation:

Zero-lookahead approximation: *Stop the propagation at the earliest time point t where all the goals are reachable ($C(G, t) < \infty$).*

One-lookahead approximation: *At the earliest time point t where all the goals are reachable, execute all the remaining events in the event queue and stop the propagation.*

One-lookahead approximation looks ahead one step in the (future) event queues when one path to achieve all the goals under the relaxed assumption is guaranteed and hopes that executing all those events would explicate some cheaper path to achieve all goals.[6]

[5]It has been pointed out in AltAlt(Nguyen, Kambhampati, & Nigenda 2001) that growing the classical planning graph until it *levels-off* is very costly in many problems.

[6]Note that even if none of those events is directly related to the goals, their executions can still lead to better (cheaper) path to reach all the goals.

Zero and one-lookahead are examples of a more general k-lookahead approximation in which we start extracting the heuristic value as soon as all the goals are reachable corresponds to *zero-lookahead* and continuing to propagate until the fix-point corresponds to the *infinite (full) lookahead*. The rationale behind the k-lookahead approximation is that when all the goals appear, which is an indication that there exists at least one (relaxed) solution, then we will look ahead one or more steps to see if we can achieve some extra improvement to the cost of achieving the goals (and thus lead to lower cost solution). For backward planners where we only need to run the propagation one time, infinite-lookahead or higher levels of lookahead may pay off, while in forward planners where we need to evaluate the cost of goals for each single search state, lower values of k may be more appropriate.

Coming back to our travel example, zero-lookahead stops the propagation process at the time point $t = 2.5$ and the goal cost is $C(in\_la, 2.5) = 8.0$. The action chain giving that cost is $\{(D^{c1}_{t \to p}, F_{p \to la}\}$. With one-lookahead, we find the lowest cost for achieving the goal in_la is $C(in\_la, 7.0) = 6.0$ and it is given by the action $(D^{c2}_{t \to la})$. With two-lookahead approximation, the lowest cost for in_la is $C(in\_la, 6.0) = 5.5$ and it is achieved by cost propagation through the action set $\{(D^{c1}_{t \to lv}, T_{lv \to la})\}$. In this example, two-lookahead has the same effect as the fix-point propagation (infinite lookahead) if the deadline to achieve in_la is later than $t = 6.0$. If it is earlier, say $Deadline(in\_la) = 5.5$, then the one-lookahead will have the same effect as the infinite-lookahead option and gives the cost of $C(in\_la, 3.0) = 7.5$ for the action chain $\{D^{c2}_{t \to phx}, F_{phx \to la}\}$.

Heuristics based on propagated cost functions

Once the propagation process terminates, the time-sensitive cost functions contain sufficient information to estimate the heuristic value of any given state. Specifically, suppose the planning graph is grown from a state S. Then the cost functions for the set of goals $G = \{(g_1, t_1), (g_2, t_2)...(g_n, t_n)\}$, $t_i = Deadline(g_i)$ can be used to derive the following estimates:

- The minimum makespan estimate for a plan starting from S, $T(P_S)$ is given by the earliest time point τ_0 (where τ_0

is the earliest time point at which all goals are reached with finite cost $C(g, t) < \infty$.)

- The minimum cost estimate of a plan starting from S and achieving a set of goals G, $C(P_S, \tau_\infty)$, can be computed by aggregating the cost estimates for achieving each of the individual goals at their respective deadlines ($C(g, deadline(g))$).[7] Notice that we use τ_∞ to denote the time point at which the cost propagation process stops. Thus, τ_∞ is the time point at which the cost functions for all individual goals $C(f, \tau_\infty)$ have lowest value.
- For each value $t : \tau_0 < t < \tau_\infty$, the cost estimate of a plan $C(P_S, t)$, which can achieve goals within a given makespan limit of t, is the aggregation of the values $C(g_i, t)$.

The makespan and the cost estimates of a state can be used as the basis for deriving heuristics. The specific way these estimates are combined to compute the heuristic values does of course depend on what the user's ultimate objective function is. In the general case, the objective function would be a function $f(C(P_S), T(P_S))$ involving both the cost and makespan value of the plan. Suppose that the objective function is a linear combination of cost and makespan:
$h(S) = f(C(P_S), T(P_S)) = \alpha.C(P_S) + (1 - \alpha).T(P_S)$
If the user only cares about the makespan value ($\alpha = 0$), then $h(S) = T(P_S) = \tau_0$. Similarly, if the user only cares about the plan cost ($\alpha = 1$), then $h(S) = C(P_S, \tau_\infty)$. In the more general case, where $0 < \alpha < 1$, then we have to find the time point t, $\tau_0 \leq t \leq \tau_\infty$, such that $h_t(S) = f(C(P_S, t), t) = \alpha.C(P_S, t) + (1 - \alpha).t$ has minimum value.

In our ongoing example, given our goal of being in Los Angeles (*at_la*), if $\alpha = 0$, the heuristic value is $h(S) = \tau_0 = 2.5$ which is the earliest time point at which $C(at\_la, t) < \infty$. The heuristic value corresponds to the propagation through action chain $(D^{c1}_{t \to p}, F_{p \to la})$. If $\alpha = 1$ and $Deadline(At_{LA}) \geq 6.0$, then $h(S) = 5.5$, which is the cheapest cost we can get at time point $\tau_\infty = 6.0$. This heuristic value represents another solution $(D^{c1}_{t \to lv}, T_{lv \to la})$. Finally, if $0 < \alpha < 1$, say $\alpha = 0.55$, then the lowest heuristic value $h(S) = \alpha.C(P_S, t) + (1 - \alpha).t$ is $h(S) = 0.55 * 7.5 + 0.45 * 3.0 = 5.47$ at time point $2.5 < t = 3.0 < 6.0$. For $\alpha = 0.55$, this heuristic value $h(S) = 5.47$ corresponds to yet another solution $(D^{c2}_{t \to p}, F_{p \to la})$.

Notice that in the general case where $0 < \alpha < 1$, even though time is measured continuously, we do not need to check every time point t: $\tau_0 < t < \tau_\infty$ to find the value where $h(S) = f(C(P_S, t), t)$ is minimal. This is due to the fact that the cost functions for all facts (including goals) are *step functions*. Thus, we only need to compute $h(S)$ at the time points where one of the cost functions $C(g_i, t)$ changes value. In our example above, we only need to calculate values of $h(S)$ at $\tau_0 = 2.5$, $t = 3.0$ and $\tau_\infty = 6.0$ to realize that $h(S)$ has minimum value at time point $t = 3.0$ for $\alpha = 0.55$.

[7]If we consider G as the set of preconditions for a dummy action that represents the goal state, then we can use any of the propagation rules (max/sum/combo) to directly estimate the total cost of achieving the goals from the given initial state. Among all the different combinations between the propagation rules and the aggregation rules to compute the total cost of the set of goals G, only the *max-max* (max-propagation to update $C(g_i, t)$, and cost of G is the maximum of the values of $C(g_i, Deadline(g_i))$ is admissible.

```
Goals: $G = \{(g_1, t_1), (g_2, t_2)...(g_n, t_n)\}$
Actions in the relaxed-plan: $RP = \{\}$
Supported facts: $SF = \{f : f \in InitialStateS\}$
**While** $G \neq \emptyset$
    Select the best action $A$ that support $g_1$
    $RP = RP + A$
    $t_A = t_1 - Dur(A)$
    Update makespan value $T(RP)$ if $t_A < T(RP)$
    **For all** $f \in Effect(A)$ added by $A$ after
        duration $t_f$ from starting point of $A$ **do**
        $SF = SF \bigcup \{(f, t_A + t_f)\}$
    **For all** $f \in Precondition(A)$ s.t $C(f, t_A) > 0$ **do**
        $G = G \bigcup \{(f, t_A)\}$
    **If** $\exists (g_i, t_i) \in G, (g_i, t_j) \in SF : t_j < t_i$ **Then**
        $G = G \setminus \{(g_i, t_i)\}$
**End while;**
```

Figure 4: Procedure to extract the relaxed plan

Computing Cost from the relaxed plan

To take into account the positive interactions between facts in the planning problems, we can also do a non-backtrack search from the goals to find a relaxed plan. Then, the total execution cost of actions in the relaxed plan and its makespan of the relaxed plan can be used to calculate the heuristic estimation. Besides a possibly better heuristic estimation, works on FF(Hoffmann 2000) points out that actions in the relaxed plan can also be used effectively to focus the search on the branches surrounding the relaxed solution. Moreover, extracting the relaxed solution allows us to use the resource adjustment techniques discussed in (Do & Kambhampati 2001) to improve the heuristic estimations. The challenge here is how to use the cost functions to develop the best relaxed plan. For the rest of this section, we will discuss the problem of how to extract the relaxed temporal plan using the propagated cost functions.

For a given initial state S and the objective function $h(S) = f(C(P_S), T(P_S))$, a greedy procedure to find a relaxed plan with the least heuristic value given the temporal planning graph is described in Figure 4. First, let RP be the set of actions in the relaxed plan, SF be the set of time-stamped facts (f_i, t_i) that are currently supported and G be the set of current goals. Thus, SF is the collection of facts supported by the initial state S and the effects of actions in RP, and G is the conjunction of top level goals and the set of preconditions of actions in RP that are not currently supported by facts in SF. The estimated heuristic value for the current (partial) relaxed plan and the current goal set is computed as follows: $h(S) = h(RP) + h(G)$ in which $h(RP) = f(C(RP), T(RP))$. For the given set of goals G, $h(G) = min\ f(C(G, t), t) : \tau_0 < t < \tau_\infty$ is calculated according to the approach discussed in the previous section. Finally, the cost and makespan estimation of the (partial) relaxed plan RP is calculated as follows:

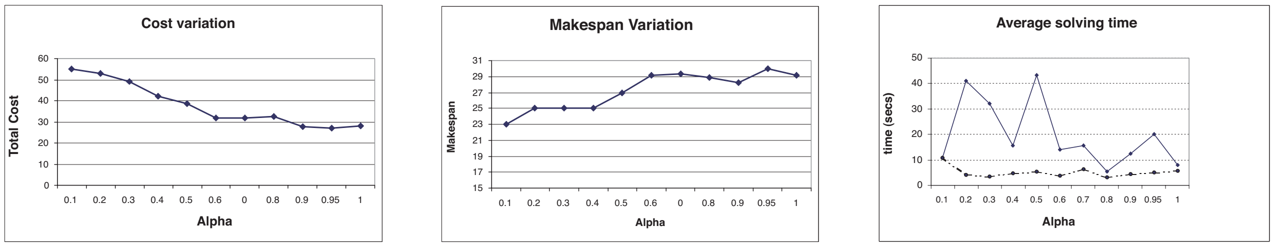

Figure 5: Cost, makespan, and solving time variations according to different weights given to them in the objective function. Each point in the graph corresponds to an average value over 20 problems.

- $C(RP) = \sum_{A \in RP} C_{exec}(A)$. Thus, $C(RP)$ is the summation of the execution costs of all actions in the relaxed plan.
- $T(RP)$ is the makespan of RP where actions in RP are aligned according to their causal relationship. We will elaborate on this in the example at the later part of this section.

In the beginning, G is the set of top level goals, RP is empty and SF contains facts in the initial state. Thus $C(RP) = 0$, $T(RP) = 0$ and $h(S) = h(G)$. We start the extraction process by going backward searching for the *least expensive* action A supporting the first goal g_1. By least expensive, we mean that A contributes the smallest amount to the objective function $h(S) = h(RP) + h(G)$ if A is added to the current relaxed plan. Specifically, for each action A that supports g_1, we calculate the value $h_A(S) = h(RP + A) + h((G \setminus g_1) \bigcup Precond(A))$ which estimates the heuristic value if we add A to the relaxed plan. We then choose the action A that has the smallest $h_A(S)$ value.

When an action A is chosen, we put its preconditions into the current goal list G, and its effects into the set of supported facts SF. Moreover, we order A to be executed and finished *before* the action that has g_1 as its precondition. Using those ordering relations between actions in RP, we can update the makespan value $T(RP)$ of the current (partial) relaxed plan. The *positive interactions* are taken care of by merging the elements in the sets SF and G. Thus, if an action in the relaxed plan supports a goal (g_j, t_j) at time point $t_i < t_j$ then we will eliminate g_j from the set of goals G.

In our ongoing example, suppose that our objective function is $h(S) = f(C(P_S), T(P_S)) = \alpha.C(P_S) + (1 - \alpha).T(P_S)$, $\alpha = 0.55$ and the infinite-lookahead criterion is used to stop the cost propagation process. When we start extracting the relaxed plan, the initial setting is $G = \{at\_la\}$, $RP = \emptyset$ and $RF = \{at\_tucson\}$. Among the three actions $D^{c_2}_{t \rightarrow la}$, $T_{lv \rightarrow la}$ and $F_{p \rightarrow la}$ that support the goal at_la, we choose action $A = F_{p \rightarrow la}$ because if we add it to the relaxed plan RP, then the estimated value $h_A(S) = h(RP + A) + h((G \setminus at\_la) \bigcup at\_phx)$ = $(\alpha * C_{exec}(F_{p \rightarrow la}) + (1 - \alpha) * Dur(F_{p \rightarrow la})) + min_t(f(C(at\_phx), t))$ = (0.55*6.0 + 0.45*1.5) + (0.55*1.5 + 0.45*1.5) = 5.475. This is the smallest among the three actions. After we add $F_{p \rightarrow la}$, we update the goal set to $G = \{at\_phx\}$. It is then easy to compare between two actions $D^{c_2}_{t \rightarrow phx}$ and $D^{c_1}_{t \rightarrow phx}$ to see that $D^{c_2}_{t \rightarrow phx}$ is cheaper to achieve *at-phx* given the value $\alpha = 0.55$. The final cost $C(P_S) = 6.0 + 1.5 = 7.5$ and makespan of $T(P_S) = 1.5 + 1.5 = 3$ of the final relaxed plan can be used as the final heuristic estimation $h(S) = 0.55 * 7.5 + 0.45 * 3 = 5.475$ for the given planning problem.

Improving the relaxed plan heuristic estimation with static mutex relation

We now discuss a way of using the static mutex relations to help improve the heuristic estimation when extracting the relaxed plan. Specifically, our approach involve the following steps:

1. Find the set of static mutex relations between the ground actions in the planning problem based on their negative interaction.
2. When extracting the relaxed plan, besides the orderings between actions that have causal-effect relationship (one action gives the effect that supports other action's preconditions), we also establish the orderings according to the mutex relations. Specifically, when a new action is added to the relaxed plan, we use the pre-calculated static mutexes to:
   - Establish ordering between mutual exclusion action pairs so that they can not be executed concurrently
   - The ordering are selected in such a way that they violate least number of existing causal links in the relaxed plan.

By using the mutex relations, we can improve the makespan estimation of the relaxed plan, and thus the heuristic estimation. Moreover, in many cases, the mutex relations can also help us detect that the relaxed plan is in fact a valid plan, and thus can lead to the early termination of the search. The details are in TR.

Empirical evaluation

We implemented the cost propagation technique discussed in this paper and used it in *Sapa*(Do & Kambhampati 2001), a forward state space metric temporal planner. The main aim of our experiments was to demonstrate that *Sapa*(Do & Kambhampati 2001), armed with cost functions can efficiently generate plans that satisfy a variety of cost/makespan tradeoff. We tested in on the set of random generated temporal logistics problem provided with TP4(Haslum & Geffner 2001). In this set of problems, we need to move 4 packages

	solving time (secs)			makespan			cost		
prob	la 0	la 1	la ∞	la 0	la 1	la ∞	la 0	la 1	la ∞
zeno1	157	0.15	0.16	320	320	320	5	5	5
zeno2	-	38.40	0.78	-	990	880	-	22	15
zeno3	1.69	0.31	0.38	650	420	420	17	13	13
zeno4	-	0.32	0.36	-	420	420	-	13	13
zeno5	3.25	9.05	2.99	670	690	660	19	20	18
zeno6	238	0.27	0.33	330	330	330	10	10	10
zeno7	333	0.27	0.24	450	370	340	10	10	10
zeno8	177	0.17	0.20	200	200	200	6	6	6
zeno9	216	0.21	0.24	330	300	300	8	8	8
log 1	424	0.42	0.45	10	10	10	16	16	16
log 2	2.21	171	159	16.02	19.37	19.37	21	21	21
log 3	-	-	0.99	-	-	10.82	-	-	13
log 4	4.28	0.55	0.62	7.42	7.12	7.12	16	12	12
log 5	-	1.93	2.23	-	12.42	12.42	-	16	16
log 6	2.43	2.28	2.59	16.42	16.42	16.42	21	21	21
log 7	3.64	2.71	34.61	19.32	17.32	26.29	29	27	30
log 8	6.53	3.88	-	17.32	15.32	-	29	27	-
log 9	3.60	3.55	4.01	20.42	20.42	20.42	31	31	31

Table 1: Quality and solving time comparison for different termination criteria.

between locations in 3 different cities. There are multiple ways to move packages, each option has different time and cost requirements. Airplanes are used to move packages between airports in different cities. Moving by airplanes takes only 3.0 time units, but is expensive, it costs us 15.0 cost units. Moving packages by trucks between locations in different cities costs only 4.0 cost units, but takes longer time of 12.0 time units. We can also move packages between locations inside one city such as offices and airports. Driving between locations in one city will cost us 2.0 units and takes 2.0 time units. Load/unload packages into truck or airplane takes 1.0 unit of time and cost 1.0 unit.

We tested with the first 20 problems in the set with the objective function being a linear combination of both total execution cost and makespan values of the plan. Specifically, the objective function is set to

$$O = \alpha.C(Plan) + (1-\alpha).T(Plan)$$

We tested with different α values ranging from $\alpha = 0$ to $\alpha = 1$. Among the techniques discussed in this paper, we used sum-propagation rule, infinite look-ahead, and relax-plan extraction using static mutex relation. Figure 5 shows how the average cost and makespan values of the solution change according to the variation of the α value. The results show that the total execution cost of the solution decrease as we increase the α value (thus, giving more weight to the execution cost in the overall objective function). In contrast, when α decreases, thus giving more weight to the makespan, then the final cost of the solution increases and the makespan value decreases. The results show that our approach indeed produces solutions that are sensitive to the objective functions that involve both time and cost.

The right most graph in the Figure 5 shows the average

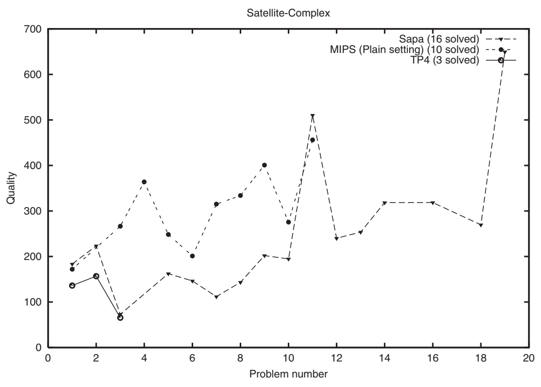

Figure 6: Results for the *complex* setting of the Satellite domain (from IPC3 results).

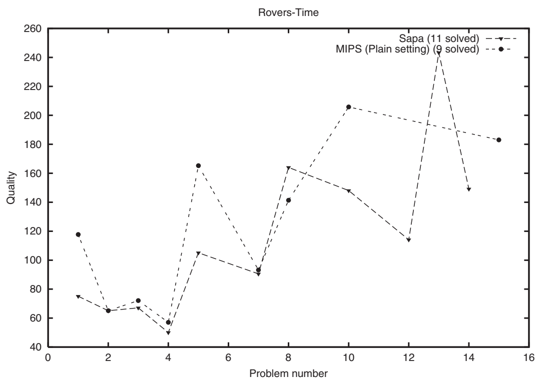

Figure 7: Results for the *time* setting of the Rover domain (from IPC3 results).

solving time for this set of experiments. For all the combinations of $\{problem, \alpha\}$, 79% (173/220) are solvable within our limit cutoff time of 300 seconds. The average solving time is 19.99 seconds and 78.61% of the instances can be solved within 10 seconds. The solid line shows the average solving time for different α values. The dashed line shows the average solving time if we take out the 20 (11%) combinations where the solving time is more than 40 seconds (more than two times the average value). We can see that while generally have larger deviations, solving the multi-objective problems are not significantly costlier than single-objective problems (which corresponds to the end points of the plots).

Beside the quality of the solution, our preliminary results indicated that the sum and combo propagation rules are similar and are much more informative than the max propagation rule. Moreover, in this test suite and limited test on some metric temporal domains, one and infinite lookahead options work similar and are generally better than zero-lookahead. Table 1 shows the comparison results for zero, one, and infinite lookahead for the set of metric temporal planning problems that come with *Sapa*. In this test suite, the Zeno flying domain involves moving packages between the cities by airplanes with different speeds and the logistics domain where trucks and airplanes move packages between different locations. However, only airplanes are allowed to move between cities. In both domains, airplanes and trucks are required to have enough fuels to execute the action. We set $\alpha = 1$, actions costs are 1 unit and action durations are dependent on the distances between locations. In this test suite, the results show that while one and infinite lookahead work similar, they are generally better than zero-lookahead, both in term of solving time and solution quality.

Sapa in the Planning Competition

We entered an implementation of Sapa, using several of the techniques discussed in this paper, in the recent international planning competition. In the competition, we focused solely on the metric/temporal domains. Even though action cost is not part of the standard PDDL 2.1 language used in the competition, infinite-lookahead unit-cost propagation employed in *Sapa* helped it achieve very good performance in problems involving both metric and temporal constraints. Figure 6, 7 and 8 show the comparison results in the highest level of PDDL2.1 setting (in terms of the complexities of temporal and metric constraints involved) for the three domains Satellite, Rovers, and ZenoTravel. The first two are inspired by the real-world applications being investigated by NASA and the third is similar to the one described in this paper. The results were collected and distributed by the IPC3's organizers and can be found at (Fox & Long 2002). Detail descriptions of domains used in the competition are also available at the same place.

Figure 6 shows that only three planners (*Sapa*, MIPS, and TP4) submitted results for the *complex* setting of the Satellite domain. In this setting, action durations depend on the setting of instruments aboard a particular satellite and the directions it needs to turn to. Moreover, each satellite has different limited capacity to store only a certain amount of image data. Goals involve taking images of different planets and stars located at different coordinate directions. Among the three planners, *Sapa* was able to solve the most number of problems (16) with better quality than MIPS. TP4 produced the best quality solutions, but was able to solve only three smallest problems.

Next, in the Rover domain, the *time* setting requires a set of scientific analysis to be done using a number of rovers. Each rover carries different set of equipments, and has different energy capacity. Moreover, each rover can only recharge its battery at certain points, which may be unreachable, that are under the sun. Figure 7 shows that only *Sapa* and MIPS were able to handle the constraints involved with this problem set. *Sapa* again solved more problems (11 vs. 9) than MIPS and also returned better or equal quality solutions in all but one case. In the *time* setting for the ZenoTravel domain, which is similar to the one discussed in the previous section, Figure 8 shows that *Sapa*, MIPS and LPG are the only three planners that submitted the results. MIPS was able to solve the most number of problems (20) but *Sapa* was not far behind (15). The quality of solutions returned by *Sapa* and MIPS are similar and are generally better than LPG (quality setting) in this domain.

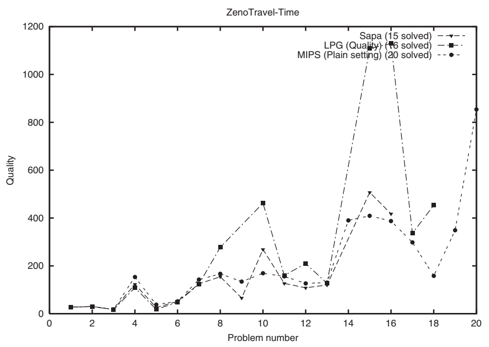

Figure 8: Results for the *time* setting of the ZenoTravel domain (from IPC3 results).

In summary, the competition results showed that *Sapa* is one of the best planners in IPC3 in solving problems involving both metric and temporal constraints.

Related Work and Discussion

Although there have been several recent domain-independent heuristic planners aimed at temporal domains, most of them have been aimed at makespan optimization, ignoring the cost aspects. For example, both TGP (Smith & Weld 1999) as well as TP4 (Haslum & Geffner 2001) focus on makespan optimization and ignore the cost aspects of the plan. As we have argued in this paper, ultimately metric temporal planners have to deal with objective functions that are based on both makespan and cost. One recent research effort that recognizes the multi-objective nature of planning is Refanidis' MO-GRT system (Refanidis & Vlahavas 2001). On one hand, the MO-GRT approach is more general than our approach in the sense that it deals with the set of non-combinable quality metrics. MO-GRT approach however treats time similar to other consumable resources (with infinite capacity). Temporal constraints on the planning problems (such as when should an effect occur during the course of action), goal deadlines, or the concurrency between actions are *ignored* to scale down the problem to the classical planning assumptions. Multi-Pegg (Zimmerman 2002) is another recent planner that considers cost-time tradeoffs in plan generation. Multi-Pegg is based on Graphplan approach, and focuses on classical planning problems with non-uniform cost actions. ASPEN (Chien et. al. 2000) is another planner that recognizes the multi-attribute nature of plan quality. ASPEN advocates an iterative repair approach for planning, that assumes the availability of a variety of plan repair strategies and their characterization in terms of their effects on the various dimensions of plan quality.

Although we evaluated our cost-sensitive heuristics in the context of *Sapa*, a forward chaining planner, the heuristics themselves can also be used in other types of planning algorithms. For example, TGP can be made cost-sensitive by making it propagate the cost functions as part of planning graph expansion. These cost functions can then be used to guide its backward branch-and-bound search. A similar approach in classical planning has been shown to be successful in (Kambhampati & Nigenda 2000).

Our work is also related to other approaches that use planning graphs as the basis for deriving heuristic estimate such as *Sapa*(Do & Kambhampati 2001), Graphplan-HSP(Kambhampati & Nigenda 2000), AltAlt(Nguyen, Kambhampati, & Nigenda 2001), RePOP(Nguyen & Kambhampati 2001), and FF(Hoffmann 2000). In the context of these efforts, our contribution can be seen as providing a way to track cost information on planning graphs. An interesting observation is that cost propagation is in some ways inherently more complex than makespan propagation. For example, once a set of literals enter the planning graph (and are not mutually exclusive), the estimate of the makespan of the shortest plan for achieving them does not change as we continue to expand the planning graph. In contrast, the estimate of the cost of the cheapest plan for achieving them can change until the planning graph levels off. This is why we had to carefully consider the effect of different criteria for stopping the expansion of the planning graph on the accuracy of the cost estimates.

In this paper, we concentrated on developing heuristics that can be sensitive to multiple dimensions of plan quality (specifically, makespan and cost). An orthogonal issue in planning with multiple criteria, that we did not explicitly address here, is how the various dimensions of plan quality should be combined during optimization. The particular approach we adopted in our empirical evaluation–viz., considering a linear combination of cost and coverage–is by no means the only reasonable way. Other approaches involve non-linear combinations of the quality criteria, as well as "tiered" objective functions (e.g. rank plans in terms of makespan, breaking ties using cost). A related issue is how to help the user decide the "weights" or "tiers" of the different criteria. Often the users may not be able to articulate their preferences between the various quality dimensions in terms of precise weights. A more standard approach out of this dilemma involves generating all non-dominated plans (the so-called "pareto-set" (Dasgupta, Chakrabarti, & DeSarkar 2001; Papadimitriou & Yannakakis 2001)), and presenting them to the user (unfortunately, often the set of non-dominated plans can be exponential (c.f. (Papadimitriou & Yannakakis 2001))). The user is then expected to pick the plan that is most palatable to them. Further, the users may not actually be able to judge the relative desirability of plans when the problems are complex and the plans are long. Thus, a more practicable approach may involve resorting to other indirect methods such as preference elicitation techniques (c.f. (Chajewska et. al. 1998)).

Conclusion

In this paper, we addressed the problem of deriving heuristics that are able to estimate both the cost and makespan of a plan for achieving a set of literals. We argued that this involves estimating the cost of individual literals (goals) as a function of time. We described techniques for computing these cost functions using temporal planning graphs. We discussed how these time-sensitive cost functions can be used as the basis for deriving heuristics to support any objective function based on makespan and execution cost. Finally, we implemented these heuristics on top of *Sapa*, and empirically demonstrated that they facilitate efficient generation of plans that offer a large variety of cost-makespan tradeoffs. We also discussed *Sapa*'s performance in the recent international planning competition. The results here show that *Sapa* is one of the best performers in the most complex metric/temporal domains.

While we considered cost of a plan in terms of a single monetary cost associated with each action, in more complex domains, the cost may be better defined as a vector comprising the different types of resource consumption. Further, in addition to cost and makespan, we may also be interested in other measures of plan quality such as robustness and execution flexibility of the plan. Our longer term goal is to support plan generation that is sensitive to these extended set of tradeoffs. To this end, we plan to extend our methodology to derive heuristics sensitive to a variety of quality measures.

References

Blum, A. and Furst, M. 1995. Fast planning throught planning graph analysis. In *Proc. of IJCAI-95.*

Bonet, B., Loerincs, G., and Geffner, H. 1997. A robust and fast action selection mechanism for planning. In *Proc. of AAAI-97.*

U. Chajewska, L. Getoor, J. Norman and Y. Shahar. 1998. Utility Elicitation as a Classification Problem. In *Proc. of UAI-98.*

S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, D.Mutz, T. Estlin, B. Smith, F. Fisher, T. Barrett, G. Stebbins, D.Tran 2000. ASPEN - Automating Space Mission Operations using Automated Planning and Scheduling. *SpaceOps 2000*

P. Dasgupta, P.P. Chakrabarti, and S.C. DeSarkar. Multiobjective Heuristic Search. *Vieweg & Son/Morgan Kaufmann,* 2001.

Do, M. and Kambhampati, S. 2001. Sapa: A domain independent heuristic metric temporal planner. In *Proc. of ECP-01*

Fox, M. and Long, D. 2002. Third International Planning Competition. http://www.dur.ac.uk/d.p.long/competition.html

Ghallab, M. and Laruelle, H. 1994. Representation and control in IxTeT, a temporal planner. In *Proc. of AIPS-94.*

Haslum, P. and Geffner, H. 2001. Heuristic Planning with Time and Resources In *Proc. of ECP-01*

Hoffmann, J. 2000. The FF Planning System: Fast Plan Generation Through Heuristic Search In *JAIR-14*:p.253 - 302.

Kambhampati, S. and Nigenda, R. 2000. Distance based goal ordering heuristics for Graphplan. In *Proc. of AIPS-00.*

Nguyen, X., Kambhampati, S., and Nigenda, R. 2001. Planning Graph as the Basis for deriving Heuristics for Plan Synthesis by State Space and CSP Search. *AIJ-135*:p.73-123.

Nguyen, X., Kambhampati, S., 2001. Reviving Partial Order Plan In *Proc. of IJCAI-01.*

Papadimitriou C. H. and Yannakakis M. 2001. Multiobjective Qury Optimization. In *ACM Conference on Principles of Database Systems (PODS).*

Refanidis, I. and Vlahavas, I. 2001. A Framework for Multi-Criteria Plan Evaluation in Heuristic State-Space Planning In *Workshop on Planning with Resources, IJCAI-01.*

Smith, D. and Weld, D. 1999. Temporal Planning with Mutual Exclusion Reasoning. In *Proc. of IJCAI-99*

Zimmerman, T. 2002. Generating parallel plans satisfying multiple criteria in anytime fashion. In *Workshop on Planning and Scheduling with Multiple criteria, AIPS-2002*