Speeding Up the Resource Envelope Computation for Activities with Linear Resource Impact

Paul H. Morris and **Jeremy Frank**
Computational Sciences Division
NASA Ames Research Center, MS 269-3
Moffett Field, CA 94035
(pmorris|frank)@email.arc.nasa.gov

Abstract

We present some advances in the area of Linear Resource Temporal Networks (LRTNs), which consist of activities that consume or produce a resource linearly over the course of their duration, subject to absolute and relative metric temporal constraints.

Previous work on LRTNs has shown how to construct tight bounds for resource availability as a function of time. The previous algorithm is pseudo-polynomial; in particular, the complexity has two components involving h, where h is the length of the scheduling horizon. However, h may be exponential in number of activities, making this of limited use for practical problems.

In this paper we build upon the previous results to prove a Persistence Theorem showing that the set of activity pieces that maximizes the availbility increases monotonically over time. This provides a polynomial bound on the number of slope changes and leads to an algorithm that eliminates one of the h factors in the complexity of the previous algorithm.

Introduction

Developing acceptable schedules for tasks that must satisfy temporal and resource constraints is a central problem of AI with numerous practical applications. Building schedules by ordering events rather than assigning event times preserves temporal flexibility; this permits the construction of a family of schedules without determining exactly when events take place while still guaranteeing that feasible solutions exist. Preserving flexibility has two potential advantages over finding a "ground" schedule. The first advantage is protection from uncertainty that can lead to costly rescheduling during schedule execution. The creation of temporally flexible plans to protect against some execution time uncertainty was described in (Morris, Muscettola, & Tsamardinos 1998) and was successfully used in controlling a spacecraft (Jónsson *et al.* 2000). The second advantage is in speeding up the search for a feasible schedule by seeking to avoid premature commitments. This approach to scheduling was studied in (Cheng & Smith 1995), (Laborie 2003a) and (Policella *et al.* 2004).

Laborie (Laborie 2003b) describes a simple but expressive formalism for scheduling problems called *Resource Temporal Networks* (RTNs). Briefly, RTNs consist of a *Simple Temporal Network* (STN) as described in (Dechter,

Meiri, & Pearl 1991), constant instantaneous resource impacts (either production or consumption) for each timepoint, and piecewise constant resource bounds. Instantaneous impacts are useful for modelling reusable resources that are allocated at the beginning of an activity and released at the end, such as power usage on a planetary rover. In this context, techniques have been developed to bound the resource availability for RTNs in polynomial time. These bounds can be used to provide early termination of search branches in the process of generating temporally flexible schedules while maintaining soundness and completeness. Both (Muscettola 2002) and (Frank 2004) provide bounds that are tight, in the sense that they justify the resource bound by proving the existence of a feasible schedule; tight bounds enhance the ability to support early termination.

More recent work (Frank & Morris 2007) has applied this approach to activities with resource impacts that are linear in the duration of the activity. This work shows that certain types of discretizations of the activities can provide *exact* resource bounds when the constraints are integer-valued. The main drawback of this work so far is that the discretization greatly increases both the set of times where the resource availability must be recomputed and the number of activities that must be considered in the calculation. In this paper, we reduce the set of times that must be considered to a polynomial size, and describe methods to control the number of discrete activities needed to calculate tight bounds.

Notation and Definitions

We will assume LRTNs have a single resource. We will also assume a constant resource upper bound R_{ub} and a lower bound of 0, and that the resource initially has R_{ub} available capacity. This easily generalizes to varying initial capacity, piecewise constant upper bounds, and (with some additional work) to piecewise linear upper and lower bounds.

Let $\mathcal{A}$ be the set of all activities of an LRTN and $n = |\mathcal{A}|$. Let $A \in \mathcal{A}$ be an activity. Let A_s be the start event of activity A, and let A_e be the end event of A. If G is a ground schedule, A_s^G denotes the value of A_s in G, and similarly for A_e^G. Activity durations and resource rates are denoted by A_d and A_r, respectively. If $A_r < 0$, then A is said to be a *consumer*; if $A_r > 0$, then A is a *producer*.

Each activity A has associated with it a constraint:

$$A_s + A_d = A_e$$

Let $\mathcal{E}$ be the set of timepoints (start or end times of activities) and suppose $E_1, E_2 \in \mathcal{E}$. There may be many *simple temporal constraints* of the form

$$x_1 \le E_1 - E_2 \le x_2$$

We let the "dummy" activity H indicate the scheduling horizon, thus $H_s = 0$, $H_e = h$ and (obviously) $H_d = h$. Absolute constraints on events are then translated into simple temporal constraints between events and H_s or H_e. (For example, the constraint $A_s \in [x_1, x_2]$ translates to $x_1 \le A_s - H_s \le x_2$.) Recall that an STN can be transformed into a *distance graph* (essentially rewriting each lower-bound constraint $x_1 \le E_1 - E_2$ as an upper-bound constraint $E_2 - E_1 \le -x_1$, so that all the constraints are upper bound constraints). For a consistent STN, we denote the shortest path distance from a timepoint E_1 to a timepoint E_2 in the distance graph by $d(E_1, E_2)$. This provides an upper-bound on the temporal distance from E_1 to E_2; thus, $E_2^G - E_1^G \le d(E_1, E_2)$ in every grounded schedule G. Note that $d(H_s, E)$ and $-d(E, H_s)$ provide absolute upper and lower bounds on E; we denote these by E_{ub} and E_{lb}, respectively. An STN may be regarded as a concise representation of a flexible schedule.

Given a ground schedule G, we denote by $Avail_G(t)$ the available resource at t in G:

$$Avail_G(t) = \sum_{A \in \mathcal{A} | A_e^G \le t} A_r A_d + \sum_{A \in \mathcal{A} | A_s^G \le t < A_e^G} (t - A_s^G) A_r$$

.

We denote by $L_{max}(t)$ the *maximum available* resource at a time t over all schedules, and by $L_{min}(t)$ the minimum available resource at t. Thus, $L_{max}(t) = \max_G Avail_G(t)$ and similarly for $L_{min}(t)$. We say G justifies $L_{max}(t)$ if $L_{max}(t) = Avail_G(t)$. We denote $\max_t L_{max}(t)$ by L_{max}^+ and $\min_t L_{max}(t)$ by L_{max}^- (similarly for L_{min}^+ and L_{min}^-).

Note that $L_{max}(t)$ and L_{min} are functions that tightly bound the availability of the schedules; we call these the upper and lower envelopes, respectively, following (Muscettola 2002).

The theory of maximum flows is used to calculate the resource bounds. It is described in standard textbooks, for example (Ahuja, Magnanti, & Orlin 1993). Let F be a flow graph and *flow* a flow; we will denote pipes of the flow graph by p. As in (Muscettola 2002), we will denote the *residual capacity* of a *flow* over a subset of the flow graph $G \subset F$ by $r_{flow}(G)$. That is, $r_{flow}(G)$ denotes the cumulative remaining capacity of pipes $p \in G$ reachable from the source for the maximum flow.

We introduce the following definitions:

Definition 1 *Given an LRTN, we define the* instants I *as* $\bigcup_{E \in \mathcal{E}} (\{E_{lb}\} \cup \{E_{ub}\})$.

Definition 2 *Given an LRTN, at time t an event $E \in \mathcal{E}$ is* pending *at t if $E_{lb} \le t \le E_{ub}$,* open *at t if $t \le E_{lb}$ and* closed *at t if $E_{ub} \le t$. An activity A is* closed *if A_e is closed,* open *if A_s is open,* completely pending *if A_s and A_e are pending, and* partially pending *otherwise.*[1]

---
[1]Note an event may be both pending and open/closed, but an activity with duration > 0 is in only one state.

Definition 3 *Given an LRTN and a pair of activities A and B, A* anti-precedes *B if $d(A_e, B_e) \le 0$.*[2]

Definition 4 *Given an LRTN, a set of activities $\mathcal{S}$ is a* predecessor set *if, when activity $S \in \mathcal{S}$, then every activity T that S anti-precedes is also in $\mathcal{S}$.*

Definition 5 *Given an LRTN and a ground schedule G, G is* split *at t if there is some activity A such that $A_s^G < t < A_e^G$. A schedule that is not split at t is* intact *at t.*

Overview

For LRTNs with integer constraints, it has been shown that $L_{max}(t)$ and $L_{min}(t)$ can be found in $O(h(nh)^6)$ time (Frank & Morris 2007). The results are limited to the case where the activity durations and resource consumption rates are constant and provided as inputs to problem instances. We preserve this restriction; to emphasize this, we henceforth denote the durations and rates by a_d and a_r, respectively, for an activity A. Also, in this paper we consider only $L_{max}(t)$; by symmetry, the results for L_{min} are similar.

In order to find $L_{max}(t)$, (Muscettola 2002) shows that it is sufficient to find $L_{max}(t)$ at the instants I, since the envelope is constant between instants. The maximum flow problem can be solved using many well-known polynomial time algorithms; for RTNs with n events, thus, the complexity of finding $L_{max}(t)$ and $L_{min}(t)$ is polynomial. Suppose we are given an RTN with $c(X)$ denoting the resource impact of event X. To find the value of the schedule justifying $L_{max}(t)$, a maximum flow problem is constructed using all anti-precedence links derived from the arc-consistent STN. The rules for building the flow problem to find $L_{max}(t)$ are as follows: all pending events are represented by nodes of the flow problem. If $d(X, Y) \le 0$ then the flow problem contains an arc $X \to Y$ with infinite capacity. If $c(X) > 0$ then the problem contains an arc $\sigma \to X$ with capacity $c(X)$. If $c(X) < 0$ then the problem contains an arc $X \to \tau$ with capacity $|c(X)|$. (The flow problem for $L_{min}(t)$ is constructed similarly, except if $c(X) > 0$ then the problem contains an arc $X \to \tau$ with capacity $c(X)$, and if $c(X) < 0$ then the problem contains an arc $\sigma \to X$ with capacity $|c(X)|$.) The maximum flow of this flow network matches all possible production with all possible consumption in a manner consistent with the precedence constraints. An RTN and associated flow problems for finding $L_{max}(t)$ and $L_{min}(t)$ are shown in Figure 1. The set of events reachable in the residual flow network is a predecessor set (since the infinite pipes resulting from the anti-precedes relation always have residual capacity), and is denoted $P_{max}(t)$. $L_{max}(t)$ is justified by scheduling all pending events in $P_{max}(t)$ before t, and all other pending events after t. The tightness of the bound is guaranteed by proving that adding the constraints that force these activities to occur before or after t is consistent with the original STN.

The approach in (Frank & Morris 2007) for finding L_{max}^+ involves chopping all activities into unit-sized pieces, and

---
[2]Implies $B_e - A_e \le 0$, i.e., every part of B non-strictly precedes some part of A, in every schedule.

(a) Resource Temporal Network

(b) Flow to find $L_{min}(9)$. (c) Flow to find $L_{max}(10)$.
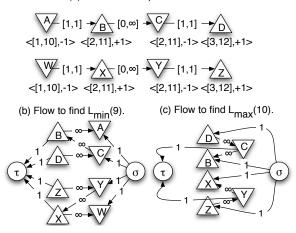
Figure 1: An RTN, the flow problem to find $L_{min}(9)$, and the flow problem to find $L_{max}(10)$.

solving the maximum flow problem described in (Muscettola 2002) at integer times. Then finding L_{max}^- involves a further chop of units into two pieces of size $\lfloor t \rfloor$ and $1 - \lfloor t \rfloor$ and solving a time-varying linear programming problem based on the maximum flow formulation, for between-integer times.

The complexity of this approach scales according to the number of maximum flow/linear programming problems that need to be solved, and according to the size of each such problem. In the previous approach, separate maxflow/LP problems need to be solved at each integer timepoint, so the number of such is bounded by $O(h)$. The size of each problem depends on the number of chopped activities, which is bounded by $O(nh)$, but may be considerably less if the activities have short durations relative to the horizon. If the maximum duration of all activities is D, then the number of chopped activities is at most nD.

The size of the horizon is typically very large in real problems. For example, it may be as large as the number of seconds in a week (604800), which limits the usefulness of the current approach.

Although chopping to units does not lead immediately to a practical algorithm, it provides a useful theoretical characterization of the schedules that determine the envelope. In this paper, we exploit that to prove a result (called the "Persistence Theorem") showing that once an activity piece becomes part of the set determining the maximum availability value at some time t, it never leaves it for larger values of t. This allows us to bound the number of slope changes and leads to a new algorithm without the need to solve a separate problem at each integer time. This eliminates the first h factor.

We also make progress on the other h factor, which arises from the need to chop each original activity into as many as h pieces, but do not eliminate it completely. As mentioned

above, this is more accurately estimated as the number of chopped activities. We present a generalization of the results in (Frank & Morris 2007) that requires chopping only until certain "drag alignment" conditions are achieved. A "worst case" example shows that may still result in chopping to units, but that is an improvement over the previous method, which required chopping to units in all cases.

Persistence Theorem

If we could bound the number of places in the envelope where slope changes can occur, then $L_{max}(t)$ can be found by calculating the slope only where it changes. The computational complexity of finding $L_{max}(t)$ will drop as long as the number of places the slope can change is smaller than h and those places can be found efficiently. Recall that for RTNs, changes of $L_{max}(t)$ can only occur at the $O(n)$ instants, and the instants are found in linear time. For LRTNs, however, previous work has shown that local minima can occur at times that are non-instant and even non-integer. Local *maxima* can only occur at integer times, but the following example shows that the times at which these changes occur may not be instants either.
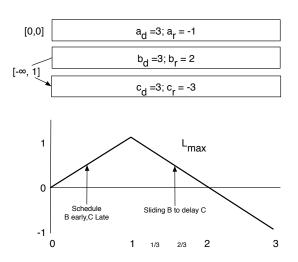

Figure 2: An LRTN for which $L_{max}(t)$ has a slope change at a time that is not an instant.

The example in Figure 2 consists of 3 activities A, B, C with rates indicated and durations 3. The following constraints hold:

$$A_s = 0, \quad B_s \geq 0, \quad C_s \geq 0, \quad C_s - B_s \leq 1$$

There are no absolute upper bounds, so 0 and 3 are the only instants. Let availability be 0 initially and consider $t \leq 1$. The best schedule for all these times is where $B_s = 0$ and $C_s = 1$. Thus $L_{max}(t)$ rises linearly between $t = 0$ and $t = 1$.

Now consider $t \geq 1$. In this case, the best schedule is to let B and C slide to the right. (Otherwise C would kick in.) Thus, the availability decreases to 0 at $t = 2$.

We see that the $L_{max}(t)$ has a peak at $t = 1$, which is not an instant, and it slopes down to 0 on both sides [3].

In spite of this, we can nevertheless bound the number of slope changes, and thus the number of points where we need to solve for the envelope. The key idea is to analyze how the set of activities that determines the maximum availability evolves with time.

The availability of a resource at a time t is determined by the set of activities or activity portions (if the activity overlaps t) that occur prior to t. We show this set grows monotonically with t, i.e., once an activity or activity portion enters this set, it never leaves it. This result also applies in the context of events with instantaneous resource impact. Although not specifically called out in (Muscettola 2004), the result is implicit in the argument (middle left column page 6) that "$F(P_{max,i-1})$ can be ignored during the computation ..." (of $F(P_{max,i})$ and subsequently). We isolate the result and give a more intuitive proof here.

The set of activities/events that occur prior to time t determines the availability at time t. The maximum flow approach identifies a unique set $P_{max}(t)$ that maximizes the incremental availability at time t, and is minimal in terms of the number of activities/events among such sets. If we add to this the closed set at time t (which also contributes to the availability at t), we see that there is a unique set $Q_{max}(t)$ that can occur prior to t that maximizes the availability at time t, and is minimal in terms of the number of activities/events.

Theorem 1 (Persistence Theorem) *The set $Q_{max}(t)$ increases monotonically with t.*

Proof: We show that the maxflow network can be reformulated in a way that makes the result immediate.

First note that the network can be modified so that all intermediate (not source or sink) pipes go directly from producers to consumers. That is, we can delete the producer/producer and consumer/consumer pipes. Since the antiprecedes relation is transitive, and intermediate pipes have infinite capacity, this does not alter the feasible flows in the source and sink pipes, and so does not change the $P_{max}(t)$ solutions.

Second it does no harm to include nodes and intermediate pipes for *all* the events in the network (even if they are not in the pending set). We need only omit the source/sink pipes for the open and closed events; since there will then be no flow through the nodes, they cannot affect the rest of the network so the maximum flow solution is unchanged.

Next we can see that in fact the sink pipes for the consumer events that are open can be included in the network without affecting the maximum flow solution. The reason for this is that any flow to these consumers can only come from producers that anti-precede them. These producers will also still be in the open set and so will have zero flow. Thus, the flow to the open consumers will be zero even if their sink pipes are included in the network.

---

[3] The consumer A constrained to start at 0 makes $t = 1$ a local maximum of $L_{max}(t)$; eliminating A leads to a slope change at $t = 1$ but $L_{max}(t)$ remains at 1

Similarly, producer source pipes can be retained in the network even after they become closed, since the only consumer events that can drain them will have had their sink pipes removed. (However, the contribution of these producers to the availability will now appear in the network contribution rather than the closed set contribution.)

With this reformulation, the incremental changes to the network as time advances consist of (1) adding source - producer pipes (as they enter the pending set) and (2) deleting consumer - sink pipes (as they exit the pending set). Intuitively, adding producers and deleting consumers means that the competition for consumers to drain the flow from the producers can only get worse over time, so if a source pipe to a producer becomes unsaturated in a maximum flow at some time, it will stay unsaturated at subsequent times. More formally, adding a source pipe or deleting a sink pipe cannot create a new augmenting path for an unsaturated producer node. This means the producer will continue to be in $Q_{max}(t)$.

□

In the context of LRTNs, the $L_{max}(t)$ problem can be formulated as a maximum flow network in terms of the chopped activities (Frank & Morris 2007), which allows us to apply the Persistence Theorem. Note that an original (unchopped) activity contributes at least partially to the availability if its earliest chopped part contributes to the availability, and it contributes totally if its latest chopped part contributes to the availability. Thus, the Persistence Theorem can be applied to the original activities also. Suppose we let $T_{max}(t)$ denote the set of (original) activities that contribute totally to the maximum availability and let $S_{max}(t)$ denote the set of activities that contribute at least partially but not totally. We will also let $U_{max}(t)$ be the set of activities that do not contribute, i.e., the activities that are neither in $S_{max}(t)$ nor in $T_{max}(t)$. Note that activities can move only from $U_{max}(t)$ to $S_{max}(t)$, $S_{max}(t)$ to $T_{max}(t)$, or $U_{max}(t)$ to $T_{max}(t)$. Then the Persistence Theorem allows us to conclude that each of $S_{max}(t)$ and $T_{max}(t)$ changes its composition at most $O(n)$ times as t increases.

Now recall the equation

$$Avail_G(t) = \sum_{A \in \mathcal{A} | A_e^G \leq t} a_r a_d + \sum_{A \in \mathcal{A} | A_s^G \leq t < A_e^G} (t - A_s^G) a_r.$$

If we know the sets $S_{max}(t)$ and $T_{max}(t)$ that contribute to the maximum availability at t, we can use this equation to formulate a linear program to determine a schedule G that maximizes the availability at time t. Thus, for fixed $T_{max}(t)$ and $S_{max}(t)$ and fixed t, we wish to maximize

$$\sum_{A \in T_{max}(t)} a_r a_d + \sum_{A \in S_{max}(t)} (t - A_s) a_r$$

over the variables A_s and A_e for every $A \in \mathcal{A}$, subject to the STN constraints and the additional constraints $A_e \leq t$ for each $A \in T_{max}(t)$, $A_s \leq t \leq A_e$ for each $A \in S_{max}(t)$, and $t \leq A_s$ for each $A \in U_{max}(t)$.

We would like to avoid having to solve this LP individually for every value of t for which a particular

$S_{max}(t)/T_{max}(t)$ pair is valid. (We will call this an *S/T region*.) Notice that the objective and the constraints of this LP are linear in t. Thus, the feasible region with t added to the set of variables is still a convex polytope. It follows that the maximum objective solution as t increases will be piecewise-linear and convex in t.

It is well known that the set of all solutions to an LP corresponds to the feasible region when some of the inequalities are changed to equalities, and the ones that are changed can be identified by solving the dual problem. Note that changing inequalities to equalities in the above problem amounts to tightening the STN constraints. Thus, the set of all solutions can be represented as a tighter STN. Recall that any STN has a unique earliest-time solution. We will refer to this as the earliest-time solution of the above LP. This "canonical" solution provides a useful theoretical tool for deriving results about the maximum availability curve.

The succession of earliest-time maximum-availability solutions for t, as t advances, may be visualized as a kind of "movie." Of particular interest are the "trajectories" of individual activities with respect to t, which we can analyze in part by determining the inequalities that become equalities. Note these cannot include $A_s = t$ or $t = A_e$ for any $A \in S_{max}(t)$; otherwise A would necessarily be in $T_{max}(t)$ or $U_{max}(t)$. However, they may include $A_e = t$ for some of the $A \in T_{max}(t)$ and $t = A_s$ for some of the $A \in U_{max}(t)$. Apart from the LP-derived equalities, an activity A in $U_{max}(t)$ will also be forced to move by the $t \leq A_s$ inequality once t reaches the absolute lower-bound of A. These *moving* activities may in turn push or pull other activities via the STN constraints. Otherwise activities remain stationary in the earliest-time solution. Thus, activities can either move along with t (maintaining a constant separation from t), or stay fixed in absolute time such that t can pass them by.

Suppose t_0 is the initial time of the S/T region. Using this approach, we can solve the dual LP for t_0 and $t_0 + \epsilon$ for some small ϵ, identify the moving activities, and project forward the earliest-time solution. Because of the linearity and convexity over t, the projected values of the variables will provide a solution until we hit a "boundary" where one of the STN or $t \leq A_e$ bounds is reached and would be violated if we projected further. (This includes the case where we begin to push an A in $U_{max}(t)$ by virtue of reaching its lower-bound.) At that point we can re-solve and project again and thus trace out the whole piecewise-linear maximum availability curve for the S/T region where this $S_{max}(t)/T_{max}(t)$ pair is valid. We remark that whether an activity is moving or stationary can vary from projection to projection within the S/T region.

Note that a $t = A_s$ equality for $A \in U_{max}(t)$ cannot cease to be true after a re-projection since activities can only keep up or fall back relative to t, and falling back would imply A is no longer in $U_{max}(t)$. Also, if an $A_e = t$ equality ceases to be true after a re-projection, it can never be true again because A_e can never "catch up" to t. Furthermore, the absolute lower-bound of an activity in $U_{max}(t)$ can be reached only once. Thus, a particular "boundary" constraint can be hit only once for this S/T region.

At the transition from one S/T region to another, it is possible to have sudden (discontinuous) changes in the earliest-time solution. However, by the Persistence Theorem activities can only jump backward (to earlier times); there can be no sudden jumps forward. Thus, the "can't catch up" argument in the last paragraph applies across the whole horizon, and a slope change caused by a particular constraint can be hit only once for all the regions.

The number of STN and $t \leq A_e$ constraints is $O(n^2)$, so there can be at most $O(n^2)$ slope changes (i.e. pieces) resulting from reaching constraint bounds. (Generalizing Figure 2 by adding independent pairs of producers and consumers like B and C with maximum separation constraints can lead to linear slope changes between instants, but we do not yet have a good example of quadratic slope changes arising due to constraint bounds.) There can be an additional $O(n)$ changes resulting from transitions between regions. Thus, computing the maximum availability curve for the entire horizon involves solving $O(n^2) + O(n) = O(n^2)$ LP problems. Unlike previous work, we do not need to formulate the LP using the maximum flow problem; it requires only $O(n)$ LP variables (we must choose the start times for activities in S_{max}) and at most $O(n^2)$ constraints.

However, the above analysis assumes we know where the different regions start and end. We still need to determine the specific times where the $S_{max}(t)/T_{max}(t)$ pair changes.

Determining the S/T Regions

We now consider how to determine the specific times where $S_{max}(t)/T_{max}(t)$ pair changes, or equivalently to determine when an individual activity enters the $S_{max}(t)$ and $T_{max}(t)$ sets. One way of doing this is to chop the activities to unit pieces and solve the maximum flow network at integer times as in the previous work. This will tell us when the different units come into $Q_{max}(t)$. However, this would not reduce the complexity.

If we focus on determining when a specific unit piece enters $Q_{max}(t)$, we can do better. We can do a binary search over the horizon looking for the integer time when the piece first comes in. The search itself requires solving $O(\log h)$ flow problems, which is linear in the compact representation of the problem. After solving each flow problem at time t, we ask whether the first (temporally ordered) unit sized component of any activity A is now in $Q_{max}(t)$, when it was not in $Q_{max}(s)$. If so, we conduct a classical binary search to determine when A entered Q_{max}. (It requires $\log_2 h$ bits to specify the value of h.) However, we still need to solve maximum flow problems at each of the $\log h$ steps.

The overall complexity of this approach is as follows: $O(n^6)$ ($O(n^2)$ linear programming problems, of complexity $O(n^3)$ each at worst (using Karamakar's algorithm, e.g.) for each of $O(n)$ S/T partitions, plus $O(\log(h)n^6 h^5)$ ($O(\log h)$ flow problems of $O(n^5 h^5)$ worst case (using Edmunds Karp, e.g.) for each of n activities.

The average case complexity is significantly smaller than this. Simplex often has performance of $O(n)$ on average, reducing the first factor to $O(n^3)$; flow problem solvers are often closer to $O(n^2)$ on sparse problems, it is likely that fewer

than $\log h$ flow problems must be solved, and it is likely that there are fewer than n^2 slope changes in $L_{max}(t)$.

Notice the difference between this approach and the previous approach; we no longer have different computational problems to identify L_{max}^+ and L_{max}^-, but we find the whole envelope by calculating $L_{max}(t)$ at each time when the slope is known to change. We do not need the chopped network to calculate $L_{max}(t)$ itself. We still need to search for times when $Q_{max}(t)$ changes, and this part of the search does require solving the flow problem on the chopped network. As a result, we have dramatically reduced the impact of h on the complexity, but not completely eliminated it.

Reduced Chopping

The D discussed above arises from the need to chop activities to unit size in order to take advantage of the results in (Frank & Morris 2007). Thus, the key to further improvements is reduce the amount of chopping needed to apply the maximum flow method.

One simple possibility is a transformation of units. If the greatest common devisor (gcd) of the integer constraint bounds is larger than 1, then we can divide all of the constraint bounds (including durations) by the gcd without changing the essential problem. This reduces the required amount of chopping. For example, in a practical application, constraints might have a granularity of 15 minutes while activities range in duration up to 2 hours in multiples of 15 minutes. In this case, the gcd is 15, and so $D = 8$.

A more sophisticated approach is to reformulate the maximum flow approach in a way that requires less chopping. This takes advantage of the fact that the network can be modified so that all intermediate (not source or sink) pipes go directly from producers to consumers. (As discussed in the proof of the Persistence Theorem, this does not alter the feasible flows in the source and sink pipes, and so does not change the $P_{max}(t)$ solutions.) This suggests that we focus on the anti-precedes relationship specifically between producers and consumers. Intuitively, we want to bring as much production as we can into the $Q_{max}(t)$ set, but that production is countered by the cost of the consumption that is dragged in also by the producer/consumer constraints. The cost of a particular consumption may be shared by several producers, and we seek a balance that maximizes the availability at t.

The maximum flow method is designed to find the optimum balance by seeking the best match of consumption to production. However, the matching in our case is complicated by the partial dragging that can result from the temporal constraints, so only portions of the consumers get matched to portions of the producers. Chopping to units ensures there is no partial dragging. However, it turns out a weaker condition can achieve the same effect.

First we define some new concepts. Intuitively, the drag relationship between a producer and a consumer is the temporal relationship that results when the producer is pulled to the left (i.e., earlier), and the consumer is pulled to the right, until the constraint between them becomes active. We can then consider the Allen relationships in this context. Of particular interest are the cases where the consumer Starts-Within the producer, and where the producer Ends-Within the consumer. If the former case, we say there is a leading producer; in the latter case, there is a trailing consumer. Intuitively, a leading producer "sticks out in front" and a trailing consumer sticks out behind.

More formally, we say a temporal constraint in which a producer A drags a consumer B has a *leading producer* if $0 < d(A_s, B_s) < a_d$. Similarly, it has a *trailing consumer* if $0 < d(A_e, B_e) < b_d$. This leads to the following result.

Theorem 2 *Suppose the drag relationships are such that there are no leading producers or trailing consumers. Then $Q_{max}(t)$ contains only complete activities (i.e., $S_{max}(t)$ is empty).*

Proof: We need only show that any producers in $Q_{max}(t)$ are complete. Since there are no trailing consumers, complete producers can only drag (i.e., force into $Q_{max}(t)$) complete consumers.

Let $\mathcal{P}$ be the set of incomplete producers in $Q_{max}(t)$. Suppose contrary to the theorem that $\mathcal{P}$ is non-empty. Let $\mathcal{C}$ be the set of incomplete consumers in $Q_{max}(t)$. Since complete producers can only drag complete consumers, the incomplete consumers in $\mathcal{C}$ can only be dragged by incomplete producers in $\mathcal{P}$. Note that each producer A in $\mathcal{P}$ must drag at least one incomplete consumer; otherwise the availability could be increased simply by moving A completely into $Q_{max}(t)$.

Now let

$$r = \sum_{A \in \mathcal{P}} a_r + \sum_{B \in \mathcal{C}} b_r$$

i.e., r is the net rate of production by the activities in $\mathcal{P}$ and $\mathcal{C}$. Note that if r is negative, the optimality of $Q_{max}(t)$ is contradicted since the availability could be increased by moving a portion of the incomplete activities out of $Q_{max}(t)$. On the other hand, if r is positive, the availability could be increased by moving an additional portion into $Q_{max}(t)$. Finally, if r is zero, the minimality of $Q_{max}(t)$ is violated. Thus, $\mathcal{P}$ must be empty and the result follows. $\square$

The significance of Theorem 2 is that it allows us to chop only to the point where there are no leading producers or trailing consumers. Recall that A is a leading producer with respect to a consumer B when $0 < d(A_s, B_s) < a_d$. We can eliminate this occurrence by chopping A at a point $d(A_s, B_s)$ after A_s. Similarly, a trailing consumer occurrence where $0 < d(A_e, B_e) < b_d$ can be removed by chopping B at a point $d(A_e, B_e)$ before B_e. However, it is easy to see that each chopping of a leading producer may create new trailing consumers and vice versa, so the chopping process must be iterated. With integer constraints, the activities cannot be chopped to smaller than unit pieces, so the iteration must terminate.

We now present an example showing that the chopping may still proceed to units in the worst case, so, unfortunately, this does not eliminate the exponential factor in general.

PRODUCERS $A_0, A_1, ..., A_n$
CONSUMERS $B_0, B_1, ..., B_n$
DURATIONS (all same) $D = 2^n$.
DRAG CONSTRAINTS

$$d(A_{0,s}, B_{i,s}) = 0 \text{ for } i = 1, ..., n$$

$$d(A_{i,s}, B_{0,s}) = 0 \text{ for } i = 1, ..., n$$

$$d(A_{i,s}, B_{i,s}) = D/2^i \text{ for } i = 1, ..., n$$

We show that the activities get chopped into 2^n pieces by induction on n. The result is easily seen for $n = 1$. Assume it is true for n. Consider the $n + 1$ case. By the inductive hypothesis, both A_0 and B_0 are chopped into 2^n pieces by the $i = 1, ..., n$ subset of the constraints. These chops are propagated, respectively, to B_{n+1} and A_{n+1}. Then the drag relationship between A_{n+1} and B_{n+1}, which is offset by $D/2^{n+1}$, subdivides each of the chops in both A_{n+1} and B_{n+1} into two further pieces, i.e., they are chopped into 2^{n+1} pieces in all. These additional chops then get propagated back to A_0 and B_0 and from them to all the A_i and B_i. The result follows.
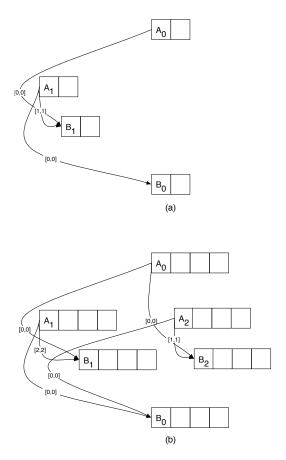

(a)

(b)

Figure 3: An LRTN exhibiting exponential chopping to enforce completeness of $Q_{max}(t)$.

Related Work

Problems involving activities with linear resource impact have been considered in two other papers (Beldiceanu & Poder 2007) (Kumar 2005), besides the (Frank & Morris 2007) paper cited earlier.

In (Beldiceanu & Poder 2007), the authors consider tasks consisting of sequences of subtasks where each of the subtasks has linear resource impact. The complexity of computing the envelope for a single resource is $O(p \log p)$ where p is the number of subtasks. However, the task model does not allow any additional constraints between the subtasks (such as the STN constraints considered here) and thus avoids the delicate balancing issues.

In (Kumar 2005), the author considers a model that includes what are called *Type 3* actions that could have linear (or more general) resource impact. However, the paper restricts Type 3 actions to be *consumers* only. This simplification also avoids the difficult balancing issues that arise for LRTNs when producer/consumer interactions are included in the picture.

Conclusions and Future Work

While not completely eliminating the pseudo-polynomial aspect of the envelope problem for LRTNs, we have removed one of the h factors and reduced the other to a point where it may be useful in practical problems. We have also answered some outstanding questions about the nature of the envelope; in particular, we have shown that even though slope changes can occur at non-instants, the number of such changes is nevertheless limited to a polynomial in n.

Since we have not found an example of quadratic slope changes between S/T boundaries, it is possible that the bound on the number of such slope changes might be loose. For example, Figure 2 is not impacted if multiple consumers are constrained to the producer in the same manner; the producer "pushes" all of the consumers ahead of it to maintain the envelope after $t = 1$. A proof of a tighter bound will shave another factor of n off of the complexity. Some practical improvements may also be possible. For example, rather than searching the entire horizon h to find the S/T boundary for an activity, we need only search where the activity is pending. We could also exploit precedences that restrict the order in which activities can enter the S/T sets. More promising yet is the prospect of a complete formulation of the problem of finding $L_{max}(t)$ as a linear program, either directly or based on a compact encoding of the maximum flow or minimum cut representation. While such a formulation may avoid all factors of h in the complexity, to date all such formulations we have found are nonlinear programs.

References

Ahuja, R.; Magnanti, T.; and Orlin, J. 1993. *Network Flows*. Prentice Hall.

Beldiceanu, N.; and Poder, E. 2007. A Continuous Multi-resources *cumulative* Constraint with Positive-Negative Resource Consumption-Production. In *CP-AI-OR 2007*. Springer-Verlag.

Cheng, C., and Smith, S. 1995. A constraint posting framework for scheduling under complex constraints. In *Joint IEEE/INRIA conference on Emerging Technologies for Factory Automation*.

Cormen, T.; Leiserson, C.; Rivest, R.; and Stein, C. 2001. *Introduction to Algorithms*. MIT Press.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–94.

Frank, J. 2004. Bounding the resource availability of partially ordered events with constant resource impact. In *Proceedings of the 10th International Conference on the Principles and Practices of Constraint Programming*.

Frank, J.; and Morris, P. 2007. Bounding the resource availability of activities with linear resource impact. In *ICAPS-2007*.

Jónsson, A.; Morris, P.; Muscettola, N.; Rajan, K.; and Smith, B. 2000. Planning in interplanetary space: Theory and practice. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*.

Kumar, T.K.S. Differential Anti-Chain Algorithms for the Generalized Resource-Envelope Problem. In *Planning, Scheduling and Constraint Satisfaction: From Theory to Practice*. McKenzie and Castillo, Eds., IOS Press, 2005.

Laborie, P. 2003a. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artif. Intell.* 143:151–188.

Laborie, P. 2003b. Resource temporal networks: Definition and complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 948 – 953.

Morris, P.; Muscettola, N.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proc. of the 15th National Conference on Artificial Intelligence*.

Muscettola, N. 2002. Computing the envelope for stepwise constant resource allocations. In *Proceedings of the 8th International Conference on the Principles and Practices of Constraint Programming*.

Muscettola, N. 2004. Incremental maximum flows for fast envelope computation. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*.

Policella, N.; Smith, S.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling*.