From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

Dispatchable execution of schedules involving consumable resources *

R. J. Wallace, E. C. Freuder
Constraint Computation Center
University of New Hampshire, Durham, NH 03824
rjw,ecf@cs.unh.edu

Abstract

Earlier work on scheduling by autonomous systems has demonstrated that schedules in the form of simple temporal networks, with intervals of values for possible event-times, can be made "dispatchable", i.e. executable incrementally in real time with guarantees against failure due to unfortunate event-time selections. In this work we show how the property of dispatchability can be extended to networks that include constraints for consumable resources. We first determine conditions under which a component of the network composed of resource constraints associated with a single sequence of activities that use a resource ("bout") will support dispatchability. Then we show how to handle interactions between resource and temporal subnetworks to insure dispatchability and how to handle sequences of bouts interspersed with resource release. The results demonstrate that flexible handling of resource use can be safely extended to the execution layer to provide more effective deployment of consumable resources.

Introduction.

From both an intellectual and practical standpoint, the development of autonomous systems that can schedule their own operations is one of the most important areas of contemporary artificial intelligence. In this domain a *de facto* standard appears to have emerged, in which the overall task of plan creation and execution is apportioned to two distinct components, or "layers" of the system, a high-level Planner-Scheduler and a lower-level Executive. The Planner creates a plan together with an associated schedule of operations. This schedule is passed to the Executive, which carries it out by initiating execution of physical components of the system at designated times.

An example of such a system is found within the Remote Agent architecture developed at NASA-Ames and currently deployed for experimental testing in the Deep Space 1 spacecraft. (See (Muscettola 1994) (Muscettola, Nayak, & Pell 1998) for detailed descriptions of this system.) In this system the Executive is given certain leeway in selecting times for scheduled operations. This is necessary in order to adjust the schedule to the actual conditions of execution. For example, failure of a rocket engine to fire immediately could break a schedule that did not allow a certain amount of slack in this and subsequent activity times. This is done by sending time bounds for each event to be scheduled and allowing the Executive to choose a specific time within each pair of bounds.

Because the Executive is operating in real time, the constraints on its operation are severe. In particular, during the instantiation of a schedule the Executive cannot afford to backtrack, i.e. it cannot go back and reschedule earlier activities whenever its previous decisions have caused it to reach a point where there are no options (a 'dead end'), because these earlier activities may have already begun. For this reason, when actual plan execution begins there must be guarantees that a schedule derived from the time-envelopes is executable incrementally or "dispatchable". That is to say, regardless of the event times that are selected by the Executive (operating in real time), the result must be a viable schedule. In recent work it has been shown that consistent temporal constraint networks, which are a basic component of the Planner-Scheduler's output in the Remote Agent system, can be made dispatchable (Muscettola, Morris, & Tsamardinos 1998).

At present, flexibility of execution can only be provided with respect to temporal constraints. Ideally one would like to provide this flexibility for resource use as well, with similar guarantees. In the present work we extend the notion of dispatchability to networks that include constraints for *consumable* resources in addition to temporal constraints. An example of the kind of resource we are concerned with is the solid state recorder that is used in spacecraft to store data from recording devices prior to transmitting it to earth. In this case there is a series of activities that require data storage (resource use), punctuated at more or less regular intervals by activities in which data is transmitted, thus freeing storage space (resource release). Here, the

*Carried out in association with NASA-Ames Research Center, Moffet Field, CA and supported in part by NSF Grant No. IRI-9504316

Copyright © 2000, American Association for Artificial Intelligence. All rights reserved.

From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

problem is to insure that the capacity of the resource is never exceeded regardless of the start- and end-times that are chosen for these activities during plan execution.

As would be expected, establishing dispatchability for the resulting simple temporal plus consumable resource network (STN-cRN) is less straightforward than the analogous problem for the STN alone. We handle this problem through a series of decomposition strategies. Specifically, we establish conditions for dispatchability that pertain to the cRN alone, then for effects of each component on the other (STN—RN and RN—STN interactions), and finally for successive 'bouts' of activities that use the resource, that are separated by instances of resource release.

In Section 2 we describe the STN-cRN. Section 3 discusses conditions for dispatchability involving the cRN. Section 4 discusses how to insure dispatchability for an STN-cRN network, despite possible interactions between the STN and RN. Section 5 extends the analysis to cover successive bouts of resource use as described above. Section 6 describes the revised dispatching execution algorithm. Section 7 gives conclusions.

Structure of a temporal-consumable-resource network

In the present Remote Agent scheduling system, the Executive receives an envelope of acceptable scheduling times in the form of a simple temporal network (STN). An example of such a network is shown in Figure 1. The key feature of such networks is that each event is associated with a single interval. This insures that the network is tractable, since it can be transformed into a digraph and solved with shortest path algorithms (Dechter, Meiri, & Pearl 1991).

That the STN in Figure 1 is *not* dispatchable can be shown by some simple examples. As in the original work by (Muscettola, Morris, & Tsamardinos 1998), we assume that during execution an event x is selected from a pool of candidate events whose antecedents have already been instantiated, and that the current time is now within the interval bounded by the earliest lower and upper bounds for event-times in this candidate set. In addition, constraint propagation can take place after an event has been given a specific time of occurrence, and is restricted to adjacent nodes in the network. In the following example, instantiations are shown on the left and results of propagation on the right, in terms of the acceptable interval for events whose nodes are adjacent in the constraint graph.

a = 0	b = 4 − 9, c = 4 − 6
b = 5	d = 7 − 9
c = 6	e = 10 − 13
d = 7	f = 14 − 17
e = 13	g = 18 − 23
f = 17	g = ?

Here, legal assignments to d and e propagate to f and g, respectively, producing non-overlapping intervals for their domains, which causes execution to fail when the constraint specifying equality of the times for these latter events becomes active. As another example, suppose that e had been given the value of 12:

e = 12	g = 17 − 22
f = 16	g = ?

In this case, f is given a value from its current domain that is outside the current domain of g, so that the equality constraint cannot be satisfied, and again execution fails.

Figure 2 shows a dispatchable network derived from the STN of Figure 1. Inspection of the figure shows that an added explicit constraint between d and e prevents e from taking the value of 13 if d is given the value 7, as in our first example. The constraint between e and f prevents f from taking the value 16 if e is given the value 12. (For formal arguments that such a network is always dispatchable, the reader is referred to (Muscettola, Morris, & Tsamardinos 1998).)

Resource constraints can be incorporated into the data structure sent to the Executive via a separate subgraph with different characteristics (the cRN). In this case, intervals represent bounds on resource use for a given activity. For example, in Figure 3 each interval, [10,20], represents a range of possible use of a resource between 10 and 20 units. In addition, k-ary constraints between endpoints prevent the resource capacity from being exceeded. In the present example, the capacity is 30 resource units, and the sum of the upper bounds exceeds capacity by 10 units. Therefore, if activity x starts before y, and if the duration stipulated for the former activity results in its using more than 10 resource units, then the upper bound of y must be reduced by the excess amount in order to satisfy the constraint between x and y.

In the full data structure, cRN nodes are linked to STN nodes that correspond to the same activity (Figure 4). Resource use is assumed to be a nondecreasing function of time, and for expository purposes we will assume a linear relation, specifically, multiplication of the temporal bounds by a positive or negative quantity for resource use and release, respectively, this quantity being constant for a given activity. Of importance is the fact that the mapping from STN event to associated resource use is bijective, i.e. one-to-one and onto, as well as monotonic. The linkage between STN and cRN is indicated by the dashed lines in Figure 4, each labeled with its constant of proportionality.

Both here and in what follows, we focus on cases in which there is a single consumable resource. If there is more than one such resource that must be handled, then each resource is associated with a separate cRN,

From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

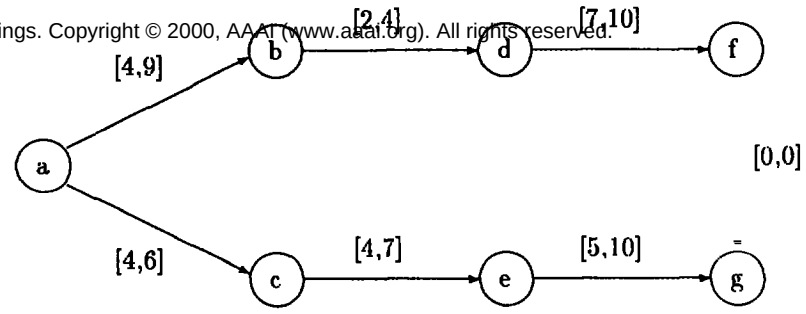

Figure 1: A consistent simple temporal network.

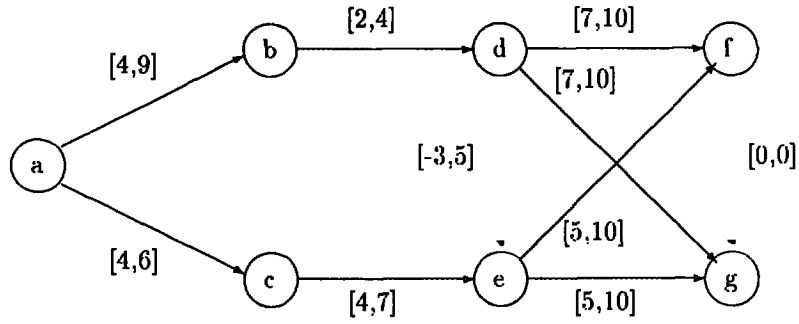

Figure 2: The temporal network of Figure 1 made dispatchable.

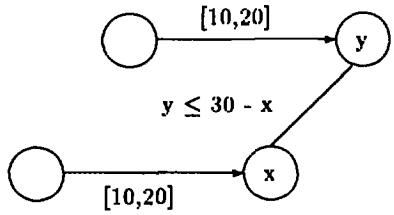

Figure 3: A consumable resource network.

and each member of the set of cRNs is connected to the STN in the manner depicted in Figure 4.

Before beginning the discussion of dispatchability of the composite network, it is important to note that the tractability of this network as a constraint satisfaction problem is not in question. This is because all constraints in both the temporal and consumable resource subnetworks are in the same tractability class, which (Jeavons, Cohen, & Gyssens 1995) refer to as Class 2 (constraints closed under binary operations that are associative, commutative, and idempotent).

Making the cRN support dispatchability.

If activities can either consume or release a given resource, then the entire sequence of such activities can be divided into 'bouts' of resource use separated by instances of release. In the next two sections we will confine our attention to a single bout of resource use and its associated cRN, and conditions for dispatchability will be specified within this context. In a later section we show that the conditions for dispatchability discovered for a single bout can be extended in a straightforward way to an entire sequence of activities.

To support dispatchability, a cRN must allow any sequence of instantiations to be made in the 'mother' STN without resource capacity being exceeded. Given the bijective mapping from STN event to resource use, this implies that any choice of value for an instance of resource use must allow some values to be chosen for all future (as yet uninstantiated) variables. We refer to this loosely as "cRN dispatchability".

For a single bout, the simplest sufficient condition for cRN dispatchability is that the sum of the upper bounds on resource use be less than or equal to the initial resource capacity,

$$\sum_{i=1}^{k} ub_i \leq C_{init} \qquad (1)$$

From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

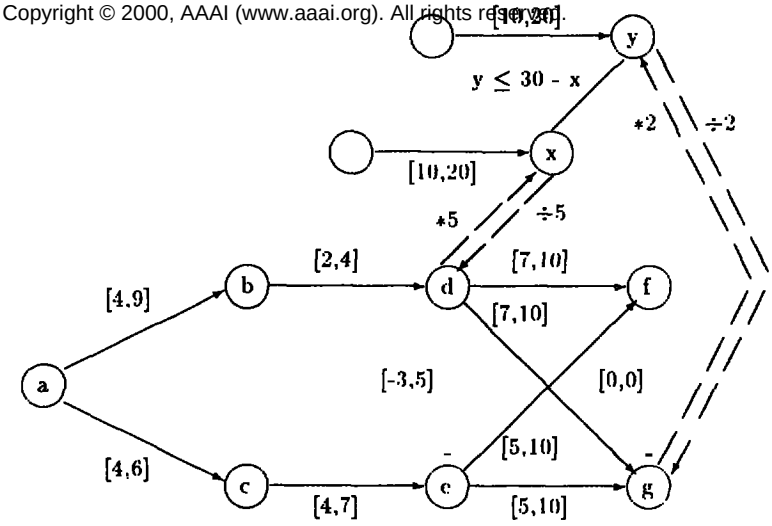

Figure 4: Combined temporal-and-consumable-resource network, in which intervals for duration (STN) and resource use (cRN) associated with the same activity are linked together. Such links are indicated by dashed lines; the linked intervals are those associated with arcs directed *toward* the nodes of origin and destination for the cross-links. For example, activity x in the cRN is associated with arc (b,d) in the STN.

Obviously, in this case the Executive does not need to process the cRN at all, since whatever values it selects from the STN, the resulting resource use will be within capacity.

Unfortunately, this simple condition puts limitations on the range of choice that can be given to the Executive. This can be seen in Figure 3, where the sum of upper bounds (40) is well above the capacity (30). Nonetheless, the cRN in this figure is dispatchable, because for every value of resource use chosen for activity X there is a usage value for activity Y within the designated bounds. Here, dispatchability obviously depends on the constraint between x and y. This suggests a weaker condition for dispatchability that at the same time allows more flexibility in the initial upper bounds on resource use by single activities.

For a set of k activities that use a given resource, this condition can be stated as follows. For all subsets of $k-1$ activities, the difference between the initial resource capacity and the sum of the upper bounds of usage for these activities is greater or equal to the lower bound of usage for the remaining kth activity. I.e.,

$$C_{init} - \sum_{i=1}^{k-1} ub_i \geq lb_k \qquad (2)$$

Or, to put this in a form corresponding to the first inequation,

$$\sum_{i=1}^{k-1} ub_i + lb_k \leq C_{init} \qquad (3)$$

That the time complexity for determining whether this condition holds is no greater than that required for the first condition is shown by the following argument. There are $O(k)$ subsets of size $k-1$, and these can be generated in sequence by swapping single activities in and out, and respective sums after the first can be generated by single additions and subtractions. This indicates that dispatchability in this sense can be determined for a set of k activities in $O(k)$ time. (Note also that this processing will be done during the planning phase, where time constraints are not as severe.)

Adjusting upper bounds when we find that the dispatchability condition is violated also appears to be easy, at least under some conditions. Thus, if the sum for one of the subsets is $\leq$ the capacity, then the upper bound that does *not* appear in the sum can be reduced to insure dispatchability. The amount it must be reduced is equal to a difference of differences. Suppose a is the element in question, i.e. the activity whose *lower* bound is in the sum that is $\leq$ the limit. And suppose that b is the element whose lower bound is in the sum that exceeds the limit by the greatest amount. Viz,

$$\sum_{i=1}^{k-2} ub_i + ub_b + lb_a \leq C_{init}$$

and

$$\sum_{i=1}^{k-2} ub_i + ub_a + lb_b > C_{init}$$

Then the upper bound of a can be reduced by the following amount to insure dispatchability for this set:

$$(u_a - l_a) - (u_b - l_b) \qquad (4)$$

From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

Since $(u_a - l_a)$ must be greater than $(u_b - l_b)$, we know that this difference must be positive, and obviously it is $\leq$ the original difference between u_a and l_a. If there is no sum less than the limit, it is sufficient to choose a sum, reduce one or more upper bounds until the condition in equation (3) is met, and then use the above procedure.

At present, this condition appears to be the weakest one possible that is still practical. Consider the next weakest condition,

$$\sum_{i=1}^{k-2} ub_i + lb_{k-1} + lb_k \leq C_{init} \qquad (5)$$

To insure dispatchability in this case, one must check $k(k-1)$ subsets of $k-2$ activities. In addition, instead of adding one constraint in the cRN, one must add k constraints to insure dispatchability. Obviously, the situation will be worse with still weaker conditions, and, although for sums of a few upper bounds, the number of subsets to test decreases, the number of constraints to add does not.

So, under the most general conditions, where it is impossible to assume dependencies among the activities with respect to resource use (and for indefinite k), the present condition is evidently the most powerful possible.

To summarize, we have two conditions for cRN dispatchability:

(i) $\sum_{i=1}^{k} ub_i \leq C$

(ii) $\sum_{i=1}^{k-1} ub_i + lb_k \leq C$ for all subsets of $k-1$ ub's

In the first case, we don't need to bother with the cRN at all during execution; in the second case we need a constraint to insure dispatchability. Hereafter, these will be referred to as conditions (i) and (ii).

Handling Interactions between STN and cRN.

For purposes of schedule execution, the STN and cRN are combined into a single connected network (cf. Figure 4), so that changes in either component can affect the other. Therefore, to establish dispatchability in this network, we must consider interactions between these basic components. (Note that we are still considering a single bout of instances of resource use prior to release.) The basic problem is that propagation in one component that leads to domain restrictions can, in turn, lead to restrictions in the other component that can compromise the conditions for dispatchability. Specifically,

1. Reductions in cRN upper bounds may delete values in the STN that are necessary to insure dispatchability in the temporal subnetwork.
2. Increasing a lower bound of an STN interval may require an increase in the lower bound of the corresponding resource interval in the cRN, thus violating cRN dispatchability condition (ii).

In this section we describe procedures that can be followed during execution to avoid compromising dispatchability in these ways. Since these are different for the two kinds of interaction, each is described in turn.

For the cRN→STN interaction, the following observation is pertinent. If changes are made to the STN, the only part of the graph we have to worry about is between the point of change, which we will call the "critical point", and variables that are already instantiated (i.e. events that are already fixed). Dispatchability will still hold with respect to future domains by virtue of the original STN dispatchability. Now, when we detect that a resource constraint may be violated, if instead of lowering the upper bound of a future resource-interval in order to satisfy that constraint, we change the upper bound associated with the variable currently being instantiated, then we reduce the 'dangerous' region of the STN (variables with domains that might contain unsupported values) to NULL. Moreover, condition (ii) for cRN dispatchability insures that we will not have to reduce any upper bounds for resource use until we arrive at the penultimate member of the set of activities - regardless of the order in which these activities are fixed. In this case, reduction of the upper bound of the penultimate activity cannot compromise dispatchability, given the dispatchability of the original STN and the bijective character of the mapping of temporal onto the resource intervals.

To insure that the STN→cRN interaction does not compromise dispatchability, before selecting a temporal value for an event we must ascertain that this will not lead to an increase in any lower bound for the interval of a future activity that uses the resource. Given condition (ii), the possibility that increasing a lower bound for resource use will compromise dispatchability does not even arise until one reaches the last activity in the bout. This means that if a subset of the activities associated with use of a resource can be designated as "candidate-last activities", then we do not have to consider this problem unless one of these activities is affected. Alternatively, we can consider the set of "candidate-penultimate activities", and in this case we can coordinate the set of STN domains with the set of cRN domains that are relevant to the prior interaction.

Both interaction problems can be solved, therefore, if it can be guaranteed that when we encounter a situation where a change can compromise dispatchability by limiting future options, we can always choose a value that will not have this effect. Fulfilling the requirement that such values always exist is simplified by the following theorem.

Theorem 1. *The requirements, that the lower bound be present in the penultimate cRN domain, so that the final domain does not need to be adjusted, and that there will be a value in a temporal domain that does not necessitate increasing the lower bound of an*

From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

adjacent domain in the candidate-penultimate set, each imply the other.

Proof: Given the cRN guarantee, the bijective, monotonic mapping from STN to cRN implies that the original lower bound will be present in the corresponding STN domain and, therefore, that values were present in adjacent domains to support this value. Although the cRN guarantee involves a specific penultimate activity, it must hold for any activity that might become the penultimate one. It therefore pertains to the same set of activities as the STN guarantee. Conversely, the STN guarantee is that all lower bound values in the candidate-penultimate set can be supported, and this implies the cRN guarantee by virtue of the bijective, monotonic mapping.

Given this theorem, a demonstration of either guarantee is sufficient to solve the 'interaction problem'. We will show how to guarantee STN lower bounds.

First, we must determine which activities fall into the candidate-penultimate set This can be done as follows. First find the resource-activity in the current bout with the latest end-time. If this activity doesn't overlap with any other resource-activity, then it needn't be considered, and one can start with the next-latest activity. After locating the first activity to be considered, we must also find all other resource-activities whose time bounds overlap with the first. Together, these comprise the candidate-penultimate set.

Now, the only situation where the lowest value in a critical domain might necessarily be increased is one with, (i) a variable, or node, C that represents the end-time of a resource-activity in the candidate penultimate set and, (ii) an arc (constraint), AC to that node from a node other than the start-time, B. Moreover, there will only be a problem if constraint AC forces the end-time C to be greater than a given value, without putting similar constraints on the start-time. In this case, depending on the start-time chosen, the end-time and hence the interval-duration can be forced to take a value greater than the minimum. This can be avoided when the STN is made dispatchable by replacing the constraint between A and C with one between A and B, the start-time for the same activity. This can be done (given the triangle inequality) if $|AB| + |BC| = |AC|$ (Muscettola, Morris, & Tsamardinos 1998). We will assume that this can be done during the planning stage, where there is more time for processing and even undoing results to meet this criterion. As a result of this manipulation, both the start- and end-times are subject to the same constraint, so the restriction on end-times that we must avoid cannot occur. If this is done for each such situation involving a candidate-penultimate activity, then this establishes the guarantee.

The following idea allows us to generalize these guarantees so that values need not be present to support lower bounds, but only a specified lowest value.

Definition 1. *We will refer to the accumulated difference between the original upper bounds for resource use, u and the actual usage r,*

$$\sum_{i=1}^{j} ub_i - \sum_{i=1}^{j} r_i.$$

as the (accumulated) credit that we may apply in the future when choosing values for resource use.

By "applying credit", we mean that one can allow for more than minimal usage, in effect increasing the lower bounds, as long as one does not exceed the credit. (In this case, of course, we must reduce the quantity of credit that is available by an amount equal to the increase in the lower bound.) An important special case is when the credit equals or exceeds the maximum excess use. Since we can calculate the latter quantity before execution by subtracting the capacity from the sum of upper bounds, we can compare this with the credit during execution. If at any point during a bout of resource use, the credit exceeds this quantity, then dispatchability cannot be compromised by any further choices of values for this bout.

More generally, the quantity of credit can be used to relax requirements on changing the upper bound for an activity in the candidate-penultimate set. In this case, we can select a value if the consequent decrease in the upper bound for the last activity is less than or equal to the credit.

With these procedures we can insure dispatchability in the combined STN-cRN network with only a very modest restriction on the 'free-wheeling' execution that was possible with the STN alone. That is, we must introduce a degree of look-ahead into the procedure in order to handle the cRN-STN interaction. Fortunately, condition (ii) insures that look-ahead will be fairly restricted. To handle cRN—STN interaction, at the time when a penultimate STN node is considered for instantiation, the Executive must check a node in the cRN adjacent to the associated cRN node. On the other hand, since the STN—cRN interaction is taken care of before execution, no look-ahead is required to effect the associated guarantee. Moreover, if we are able to set up a schedule so that only a limited number of activities can ever become the penultimate activity, we can also reduce decision making during planning.

One other potential restriction on dispatchable execution with a simple STN must be mentioned. In the original description of dispatchable networks by (Muscettola, Morris, & Tsamardinos 1998) the authors describe a procedure for deriving STNs with the minimum number of arcs consistent with dispatchability. In the present situation, although it is still possible to derive a "minimal network", this might not include all the arcs that represent activities associated with resource use. As a consequence, it would sometimes be more difficult to calculate resource use appropriately during

From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

schedule execution. In many cases, where activities associated with resource use are a small minority of all activities to be scheduled, the use of networks that are not completely 'minimalized' will probably have only minor effects on execution efficiency.

Dispatchability over Sequences of Resource Release.

The conditions described in previous sections pertain to schedule instantiation involving a single bout of resource usage, either before the first instance of resource release, or between such instances if these latter return the capacity to its initial value. Complications arise when the capacity is not restored to its original value. For one instance of release, dispatchability conditions associated with resource use can be expressed in terms of C_{new},

$$C_{new} = C_{init} - \max(0, \min(\sum_{R_{prev}}, C_{init}) - lb_p) \quad (6)$$

where lb_p is the lower bound for release and the sum on R_{prev} is a sum of upper bounds on resource use prior to release.

More generally, we have the following nested recurrence relation for the ith instance of resource release,

$$C_i = C_0 - \max_{i}(0, \max_{i-1}(0, .. \max_{1}(0, \min(\sum_{R_0}, C_0) - lb_{p_1})$$
$$+ \min(\sum_{R_1}, C_1) - lb_{p_2}) \ldots$$
$$+ \min(\sum_{R_{i-1}}, C_{i-1}) - lb_{p_i}) \quad (7)$$

This condition on dispatchability is conservative. However, during execution successive precise limits on capacity can be calculated when the values for usage and release are established. In this case, the formula is simply,

$$C_i = C_{i-1} - max(0, \sum_{R_{i-1}} - P_i) \quad (8)$$

If, to avoid extra bookkeeping during execution, limits are pre-calculated, this analysis suggests some strategies for building a plan/schedule:

1. Limit the length of the plan sent to the Executive so that there are only a small number of instances of resource release. This will reduce the cost of computing equation (8) and will avoid overly conservative values for C_i, due to the fact that the latter is based on upper bounds for resource use.
2. Arrange to have all of the capacity released; overschedule this activity if necessary.

The argument above does not consider situations in which resource use overlaps release. In such cases, a simple ordering by start times can be used to allocate such activities to bouts so that dispatchability conditions can be calculated correctly.

1. **Let A = {start}**
   curr_time = 0
   S = { }
   boutsize = k_0
   boutcounter = 0
   curr_capacity = C_{init}
2. **choose TP ∈ A such that curr_time ∈ [TP$_l$, TPu]**
3. **Let TP = curr_time**
   S = S ⋃ TP
4. **Propagate to neighbors in STN and cRN**
5. If TP is the start of a resource activity
   increment boutcounter
   If (boutcounter = boutsize − 1)
   check constraint in cRN to see if activity duration must be reduced; if so, adjust [lb,ub] for endtime associated with this TP
   else if TP is the end of a resource activity
   update $\sum r$ and $\sum(u - r)$
   else if TP is the end of a release activity
   update curr_capacity
   set boutsize = k_{next}
   boutcounter = 0
6. **Let A = A ⋃ {TP$_i$} such that no negative edges from TP$_i$ have destination ∉ S**
7. **While curr_time ∉ [min TP$_{A_l}$,min TP$_{A_u}$]**
   curr_time = curr_time + Δ time
8. **If ∃ TP ∉ S go to 2.**
   else
   done

Figure 5: Dispatching execution controller for cRN-STN. Lines in boldface are from original dispatching controller for STN in Muscettola et al 1998 (their Fig. 3).

Dispatching execution for cRN-STN

Once proper conditions are established for execution, the latter is not much different from the original dispatching execution algorithm of (Muscettola, Morris, & Tsamardinos 1998) (Figure 5). The only important difference is that the procedure must now keep track of the instantiation of resource activity start-times in order to detect when the next to last activity in a bout has started so it can perform the tests for cRN dispatchability. (The procedure in Figure 5 does not include the case where the summed credit exceeds the maximum excess usage for this bout, but this can be handled by elaborating the test within the first if clause in step 5.)

The procedure in Figure 5 handles only one resource. With more than one resource one may have to track more than one bout simultaneously. This can be done

From: AIPS 2000 Proceedings. Copyright © 2000, AAAI (www.aaai.org). All rights reserved.

with arrays whose indices indicate the resource in question, and where each resource time point is associated with an index, without altering the time complexity.

To show the operation of this algorithm, we return to the earlier example of STN dispatchability (Figure 2) and run through it again, this time considering also the resource constraint (Figure 4). In the following execution, A is the set of candidates for instantiation, as in the pseudocode above, and I is the interval used in step 7.

a = 0 b = 4 − 9, c = 4 − 6
A = {b,c}, I = [4,6]
b = 5 d = 7 − 9
test cRN d = 7 − 7
A = {c,d}, I = [4,6]
c = 6 e = 10 − 13
A = {d,e}, I = [7,7]
d = 7 e = 10 − 12, f = 14 − 17, g = 14 − 17
A = {e,f}, I = [10,12]
e = 12 f = 17 − 17, g = 17 − 17
A = {f,g}, I = [17,17]
f = 17 g = 17 − 17
A = {g}, I = [17,17]
g = 17
A = {}

It will be observed that, in this case, the only effect of the additional dispatchability requirements is the lookahead step in the cRN after node b is instantiated, which leads to a restriction of activity (b,d) to the minimum duration.

Summary and Conclusions.

In this paper we have shown how to extend the important property of dispatchability to the case where consumable resource constraints are involved. Moreover, we have shown that it is possible to allow the same kinds of flexibility with respect to feasible values that is possible with simple temporal networks, as demonstrated in the work of (Muscettola, Morris, & Tsamardinos 1998).

Throughout the discussions in this paper, the criterion for cRN dispatchability referred to as condition (ii) has played a prominent role. As shown in Section 4, this criterion not only guarantees dispatchability in the cRN component, but figures critically in procedures that prevent interactions between the STN and cRN from compromising dispatchability in either component.

Not surprisingly with a more complex network to process and with more conditions to test, the procedure during execution is more complicated than it was with the STN alone. However, since it is possible to establish many of the conditions for dispatchability before schedule execution, the effect on efficiency of execution should be relatively modest. At the same time, the example in the previous section suggests that insuring dispatchability under these conditions may set additional limits on the utilization of a resource. (But note that with only two resource-activities, as in the present example, this effect may be maximal.) This suggests a need for further work to develop strategies for enhancing usage while complying with requirements for dispatchability, and to better characterize the tradeoff that may exist between these two goals.

The present discussion is pertinent to a large class of problems encountered in planning by autonomous systems such as spacecraft, of which the solid state recording problem is one example. In these systems, greater flexibility of resource use can make operations involving tasks such as data collection more effective, thus increasing the likelihood that overall mission goals are accomplished.

References

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Jeavons, P.; Cohen, D.; and Gyssens, M. 1995. A unifying framework for tractable constraints. In Montanari, U., and Rossi, F., eds., *Principles and Practice of Constraint Programming - CP '95*, volume 976 of *Lecture Notes in Computer Science*. Berlin: Springer. 276 291.

Muscettola, N. 1994. HSTS: Integrating planning and scheduling. In Zweben, M., and Fox, M. S., eds., *Intelligent Scheduling*. San Mateo, CA: Morgan Kaufmann. 169 212.

Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Proceedings, Sixth International Conference on Principles of Knowledge Representation and Reasoning, KR-98*.

Muscettola, N.; Nayak, P. P.; and Pell, B. 1998. Remote agent: to boldly go where no AI system has gone before. *Artificial Intelligence* 103:5 47.