Managing Communication Limitations in Partially Controllable Multi-Agent Plans

John Stedl and **Brian Williams**
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
32 Vassar St. Room 32-G275, Cambridge, MA 02139
stedl@mit.edu, williams@mit.edu
Keywords: temporal reasoning, uncertainty, multi-robot systems
ID number: 694

Abstract

In most real world situations, cooperative multi-agent plans will contain activities that are not under direct control by the agents. In order to enable the agents to robustly adapt to this temporal uncertainty, the agents must be able to communicate at execution time in order to dynamically schedule their plans. However, it is often undesirable or impossible to maintain communication between all the agents, throughout the entire duration of the plan.

This paper introduces a two-layer approach that clusters the tightly coordinated portions of a multi-agent plan into a set of group plans, such that each group plan only requires loose coordination with one another. The key contribution of this paper is a polynomial time, Hierarchical Reformulation (HR) algorithm that combines the properties of strong and dynamic controllability, in order to decouple the partially controllable group plans from one another, while enabling the tightly coordinated activities within each group plan to be scheduled dynamically.

Introduction

The domain of plan scheduling and execution for multiple robots cooperating to achieve a common goal has applications in a wide variety of fields such as cooperative observations of Earth orbiting satellites. These applications often require tight temporal coordination between the agents, which must also be able to robustly adapt to uncontrollable events.

Previous work on dispatching of temporally flexible plans (Muscettola, Morris, & Tsamardinos 1998) (Morris, Muscettola, & Vidal 2001) provided a framework for robust execution of temporal plans. The *executive* consists of a *reformulator* and a *dispatcher*. The reformulator is an off-line compilation algorithm that prepares the plan for efficient execution. The dispatcher is an online dynamic scheduling algorithm that exploits the temporal flexibility of the plan, by waiting to schedule events until the last possible moment. In this least commitment execution strategy, the dispatcher schedules and dispatches the tasks simultaneously, rather than scheduling the tasks prior to execution. This dynamic execution strategy enables the agent to adapt to runtime uncertainty, at the cost of some online constraint propagation. Specifically, the dispatcher must propagate the execution times of each event, through local temporal constraints, towards future events, every time an event is executed. This propagation enables the dispatcher to select consistent execution times for these future events. This work, however, did not tackle the case of a distributed dispatcher for multi-robot plans, for which the challenge resides in the fact that the agents must be cope with communication limitations at execution time.

Previous work on execution of *Simple Temporal Networks with Uncertainty* (*STNUs*) (Vidal & Fargier 1999) (Morris, Muscettola, & Vidal 2001) provided methods to achieve robust execution of plans that contain uncontrollable events. STNUs are an extension of Simple Temporal Networks (STNs) (Dechter, Meiri, & Pearl 1995), in which only some of the events (or *timepoints*) in the plan are fully controllable (or *executable*), while other *contingent* timepoints cannot be scheduled directly, but rather are observed during plan execution. *Links* between pairs of timepoints impose flexible temporal constraints, which express temporal coordination in the plan (in the form of *requirement* links), and model the duration of uncontrollable activities (in the case of *contingent* links).

(Vidal & Fargier 1996) defined a set of controllability properties for STNUs that determines under what conditions an agent can guarantee it successful execution of the plan. Informally, an STNU is controllable if there exists a consistent strategy for scheduling the executable timepoints of the plan, for all possible durations of the uncontrollable activities (subject to the constraints on the contingent and requirement links). There are three primary levels of controllability; a network is *strongly controllable* if there exists a viable execution strategy that does not depend on the outcome of the uncontrollable durations. In this case, it is possible to statically schedule all executable timepoints beforehand. A network is *dynamically controllable* if there exists a viable execution strategy that only depends on the knowledge of outcomes of past uncontrollable events. Finally, a network is *weakly controllable* if there is a viable execution strategy, given that we know the outcomes for all the uncontrollable events beforehand. Furthermore, strong controllability implies dynamic controllability which in turn implies weak controllability (Vidal & Fargier 1996).

(Morris, Muscettola, & Vidal 2001) presented a polynomial time dynamic controllability algorithm that both checks if a plan is dynamically controllable and reformulates the plan for efficient dynamic execution. (Vidal & Fargier 1999) introduced a polynomial time algorithm to check for strong controllability. Waypoint controllability, introduced

by (Morris & Muscettola 1999), combines the properties of strong and weak controllability. In this framework, a subset of the timepoints, designated as waypoints, are scheduled prior to knowing the uncertain durations; the remaining timepoints are scheduled once all of the uncertainty in the plan has been resolved. This provides a means to partition a partially controllable plan; however, it does not enable the agents to adapt to the uncertainty at execution time.

In the area of collaborative multi-agent planning and scheduling, (**?**) presented a Temporal Decoupling Algorithm that provides a means to decouple a Simple Temporal Network by adding additional constraints to the network

This paper presents a novel hierarchical reformulation algorithm that mitigates the need for communication at execution time between loosely coupled agents, while enabling tightly coupled agents to dynamically adapt to uncertainty. In particular, the algorithm preserves the flexibility in places where tight coordination is required and fixes the schedule in places where the agents only require loose coordination.

Our two layer approach is similar to waypoint controllability; however, in this paper we seek a property that combines strong controllability with dynamic controllability instead of weak controllability. Specifically, in this paper we show a subset of the plan, i.e. the start of each group plan, can be scheduled off-line without knowing the uncertain durations, while the remaining portions (the rest of the group plans) can be scheduled dynamically.

Overall Approach

In this paper, we divide the full multi-agent plan into a set of tightly coordinated group plans. We assume that the agents that participate in these group plans are free to communicate with one another. These group plans are loosely coupled via a higher level mission plan. The mission plan uses a simplified abstraction for each group plan that hides the details of the group plan. This encapsulation enables the reformulation algorithm to reason about the overall mission, without getting into the internal details of the group plans.

Our overall approach is presented in Figure 1.The multi-agent plan is formulated as a two-layer Figure 1a. The HR algorithm, converts the two-layer plan into a set of decoupled, minimal dispatchable group plans. The HR algorithm decouples each group plan by applying to the a variant of the strong controllability algorithm (Vidal 2000)(Figure 1b), in the mission plan and applies the dynamic controllability algorithm (Morris, Muscettola, & Vidal 2001) to each group plan (Figure 1c). Finally, we apply an edge trimming algorithm (Tsamardinos, Muscettola, & Morris 1998) to each dispatchable group plan (Figure 1d), in order to remove the redundant constraints from the dispatchable plan.

First we introduce our two-layer plans. Then, we present the HR reformulation algorithm for two-layer plans. We end with a discussion of the HR algorithm.

Formal Definition of Two-Layer Plans

In this section we introduce a two-layer multi-agent plan that consists of a high level mission plan, and a set of lower level group plans, and a mapping between the group activities in the mission plan and group plans.

The mission plan, M, specifies the coordination requirements between a set of group activities A, where each group

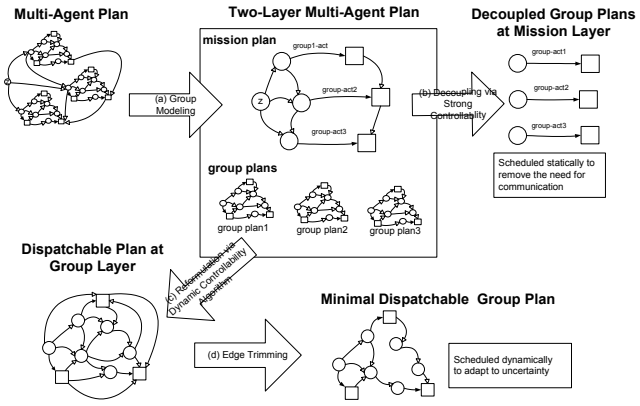

Figure 1: Hierarchical Reformulation Algorithm Overview

activity, $a_i \in A$, is mapped to a unique group plan, $g_i \in G$, via the mapping function, B. The contingent group activity, a_i contains of a start time, s_i, and end time e_i. The mission plan describes the high level structure of the mission. It specifies a set of temporal constraints on a set of abstract group activities. The group activities are a simplified representations of the detailed group plan.

The groups are in charge of scheduling their own plans, therefore, with respect to the mission plan, the duration of the group activities are uncontrollable. In order to maintain maximum flexibility in the group plans, the mission plan models each group plan as a contingent link that is bounded by the feasible duration of the group plan.

The group plans specify the details of each group activity. Specifically, each group plan contains a set of activities to be executed the agents in that group and a set of the temporal constraints on those activities.

The two layer, partially controllable, multi-agent plans with communication constraints are formalized as a two-layer Multi-Agent Temporal Plan Network with Uncertainty (MTPNU). A TPN is a set of activities to be performed, each of which includes a start and end time, together with a set of temporal constraints that specify the valid activity start and end times for each activity, specified as a simple temporal constraint. Hence a TPN is a generalization of a STN consisting of a set of activities A, and a mappings, $T^+ : A \rightarrow N^(+)$, and $T^- : A \rightarrow N^-$, mapping the start and end times of each activity to the timepoints in the STN. A TPN under uncertainty (TPNU) is analogous, where the temporal constraints are of the plan expressed as a STNU. In this case, the duration of each activity is either controllable and expressed as a requirement link, or uncontrollable and expressed as a contingent link. A multi-agent TPNU (MTPNU) extends the TPNU. A MTPNU introduces a set of agents, Q, and a distribution, $D : N \rightarrow Q$, mapping the timepoints, N, to an agent, Q.

A two-layer MTPNU extends the definition of the MTPNU. The two-layer MPTNU is the tuple $< M, G, B >$, where M is the mission plan and G is a set of group plans, and B is a function mapping the contingent group activities A in the mission plan to a group plan. Both the mission plan and group plans are specified as a MPTNU.

The mission plan, $M =< \Gamma, A, T, Q, D, \Pi, \Psi >$, where:

- Γ : is the $STNU =< N, E, l, u, C >$ that specifies the temporal constraints of the plan.
- A: is a set of group activities
- T: consists of two functions $T^+ : A \rightarrow N$, and $T^- : A \rightarrow N$, which map the start time and end time of each group activity to a timepoint, respectively.
- Q: is a set of groups.
- D: is a group distribution function, $D : N \rightarrow Q$, mapping each timepoint in the mission plan to a group.

A group plan $g \in G$ is a tuple $< \Gamma, A, T, Q, D, \Pi >$, where:

- Γ : is a $STNU =< N, E, l, u, C >$ that specifies the temporal constraints of the group plan.
- A : is a set of activities.
- T : is a mapping $T^+ : A \rightarrow N$, and $T^- : A \rightarrow N$, which map the start time and end time of each activity to a timepoint in Γ.
- Q : is a set of agents.
- D : is a distribution which is a mapping $D : N \rightarrow Q$ that maps each timepoint in the group plan to an agent in Q.

We assume that the agents within each group plan are able freely communicate with other agents in their group plan; however, communication in not guaranteed between agents in different groups.

Hierarchical Reformulation Algorithm

In this section, we present our novel Hierarchical Reformulation (HR) algorithm. The HR algorithm is a centralized reformulation algorithm that transforms a two-layer plan into a set of decoupled, minimally dispatchable group plans. The HR algorithm enables each group to dynamically execute their plan independently.

The HR algorithm operates on both layers of the two-layer plan. The dynamic controllability algorithm operates on the group plans, whereas, the decoupling algorithm, based on the strong controllability algorithm, operates on the mission plan.

The pseudo-code for the HR algorithm is shown in Figure 1. The algorithm takes in a two-layer MTPNU, P = $\langle M, G, B \rangle$, consisting of a mission plan, M, and a set of group plans, G, and mapping B and generates a set of decoupled, dispatchable MTPNUs. The algorithm returns true if the reformulation succeeds; otherwise, false.

The HR algorithm may fail for several reasons. The HR algorithm fails if either the mission plan or group plans are temporally inconsistent. Furthermore, the HR algorithm fails if the group plans are not dynamically controllable or if the mission plan is not strongly controllable.

Consider the two-layer plan illustrated in Figure 2. The mission plan (shown in Figure 2a) contains two group activities: group act1 and group act2. The corresponding group plans are shown Figure 2b,c. Both of the group plans consist of three timepoints and one contingent activity.

Lines 1-3 of the HR algorithm, shown in 1, calls the UPDATE_GROUP_ACTIVITIES function and returns false if the update reveals a temporal inconsistency in any of the group plans. This function is called at the beginning of the

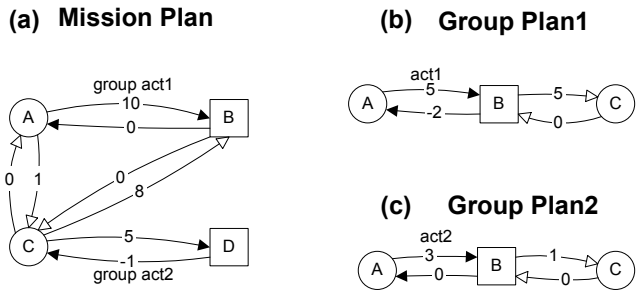

Figure 2: (a) The simple two-layer mission plan, (b) group plan1 (c) group plan2.

Alg. 1 HIERARCHICAL_REFORMULATION(P)

```
1: consistent ← UPDATE_GROUP_ACTIVITIES(G,M)
2: if ¬ consistent then
3:    return FALSE
4: end if
5: consistent ← COMPUTE_APSP_GRAPH(M)
6: if ¬ consistent then
7:    return FALSE
8: end if
9: UPDATE_GROUP_PLANS(G,M)
10: for each g ∈ G do
11:    controllable ← DC(g)
12:    if ¬ controllable then
13:       return FALSE
14:    end if
15: end for
16: UPDATE_GROUP_ACTIVITIES(G,M)
17: success ← DECOUPLE(M,G)
18: return success
```

Alg. 2 UPDATE_GROUP_ACTIVITIES(M,G)

```
1: for each group plan g ∈ G do
2:    s ← start timepoint of g
3:    if ¬ BELLMAN_FORD_SSSP(g,s) then
4:       return FALSE
5:    end if
6:    ub ← max( d[n] for each n ∈ N[g])
7:    BELLMAN_FORD_SDSP(g,s)
8:    lb ← -min( d[n] for each n ∈ N[g])
9:    groupactivity ← GET_GROUP_ACTIVITY(g)
10:   UPDATE_EDGE(M,START[m], END[m], ub)
11:   UPDATE_EDGE(M,END[m], START[m], -lb)
12: end for
13: return TRUE
```

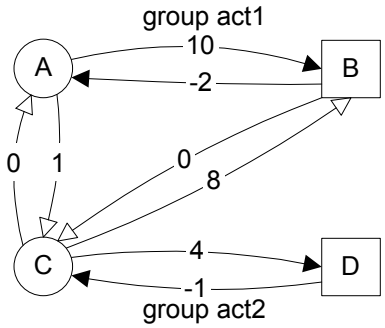

Figure 3: The UPDATE_GROUP_ACTIVITIES function updates the edges associated with the group activities in the mission plan. AB is updated to 9 and CD is updated to 4.

HR algorithm, in order to synchronize the mission plan with the constraints specified in the group plans.

The UPDATE_GROUP_ACTIVITIES function first computes the feasible duration of each group plan. Next, it updates the contingent timebounds of the corresponding contingent bounds in the mission plan. The pseudo code for the UPDATE_GROUP_ACTIVITIES function is shown 2. The feasible durations are computed by calling two Bellman-Ford Single-Source Shortest-Path (SSSP) computations (Dechter, Meiri, & Pearl 1995) (Cormen, Leiserson, & Rivest 1990). If the SSSP computation detects an inconsistency in any of the group plans, the algorithm returns false, otherwise true.

For example, the two-layer plan shown in Figure 2. For group plan1, the maximum SSSP distance is 10 for the path ABC, and the minimum SDSP is -2 for the path CBA. The UPDATE_GROUP_ACTIVITIES function leaves the distance of the contingent edge AB in the mission plan at 10; however, the distance of the contingent edge BA is updated to -2. For group plan2, the maximum SSSP distance is 4 for the path ABC, and the minimum SDSP is 0 for the path CBA. For this group plan, the UPDATE_GROUP_ACTIVITIES function updates the distance of the mission plan's contingent edge CD to 4, while the contingent edge DC remains at -1. Both group plans are temporally consistent; therefore, the UPDATE_GROUP_ACTIVITIES function returns true. Figure 3 shows the updated mission plan after calling the UPDATE_GROUP_ACTIVITIES in the HR algorithm.

Lines 4-7 of the HR algorithm computes the All-Pairs Shortest-Path graph (APSP-graph) of the mission plan's distance graph (returning false if the mission plan is temporally inconsistent). Then the HR algorithm updates the timebounds of the group plans if the edges associated with the group activities are tightened by the APSP-graph. The COMPUTE_APSP_GRAPH function in Line 4 computes the APSP-graph, given the mission plan's distance graph. This APSP-graph is maintained separate from the mission plan's distance graph. The APSP computation is performed by either Johnson's algorithm or Floyd-Warshall's algorithm (Cormen, Leiserson, & Rivest 1990).

The APSP-graph is computed for two purposes. First, it checks if the mission plan is temporally consistent. If the mission plan is inconsistent, then the algorithm returns false

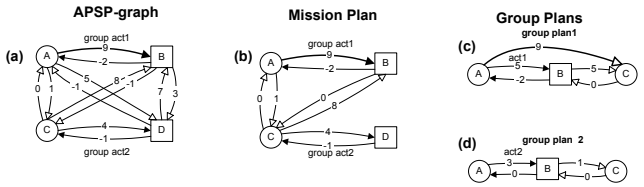

Figure 4: (a) mission plan's APSP-graph (b) Updated mission plan (c) Updated group plan 1 (d) Updated group plan 2.

in Line 6. Second, the APSP-graph is used to deduce any tightenings on the group activities implied by the constraints in the mission plan's distance graph. If the edges in the APSP-graph, corresponding to the group activity edges, are tightened, then the HR algorithm updates the corresponding group plan. The group plans are updated by calling the UPDATE_GROUP_PLANS function, in Line 7 of the HR algorithm.

Alg. 3 UPDATE_GROUP_PLANS(M,G)

1: **for** each group activity $m \in Group_activities[M]$ **do**
2: $s \leftarrow$ START(m)
3: $e \leftarrow$ END(m)
4: $ub \leftarrow$ M.APSPgraph[s, e]
5: $lb \leftarrow$ -M.APSPgraph[e, s]
6: UPDATE_EDGE(M,s,e,ub)
7: UPDATE_EDGE(M,e,s,$-lb$)
8: $g \leftarrow$ GROUP(m)
9: $s \leftarrow$ START(g)
10: $e \leftarrow$ END(g)
11: UPDATE_EDGE(g,s,e,ub)
12: UPDATE_EDGE(g,e,s,$-lb$)
13: **end for**

The pseudo code for the UPDATE_GROUP_PLANS function is shown in Figure 1-24. This function loops through each group activity in the mission plan and updates the bounds in the corresponding group plan.

For example, the mission plan's APSP-graph is shown in Figure 4. The APSP-graph edge AB is smaller than the edge AB in the mission plan's distance graph. This edge AB is associated with the upper bound on the group act1. The edge AB is tightened from 10 to 9. The UPDATE_GROUP_PLANS function updates the mission plan's distance graph accordingly, as shown in Figure 4b. The UPDATE_GROUP_PLAN function then updates the group plans. The updated group plans are shown in Figure 4(c-d). The UPDATE_GROUP_PLANS function adds the edge AC = 9 to group plan1, corresponding to the edge AB = 9 in the mission plan, and adds the edge CA = -1 to group plan2, corresponding to the edge DC = -1 in the mission plan. Note that the edge DC = -1 was present in the original mission plan, whereas the edge AB = 9 was derived by the APSP-graph.

After the group plan's timebounds are updated, the HR algorithm calls the dynamic controllability (DC) algorithm (Morris, Muscettola, & Vidal 2001), in order to reformulate each group plan into a minimal dispatchable group plan, on

Line 9. If this reformulation succeeds, then the group plan is dynamically controllable and the HR algorithm continues. However, if the DC algorithm fails (for any group plan), then the HR algorithm terminates and returns FALSE.

The complete description of the DC algorithm is presented in (Morris, Muscettola, & Vidal 2001). For now the reader only needs to understand that the DC algorithm is a reformulation algorithm that either adds or tightens the constraints of the group plan. These additional constraints may alter the range of feasible durations of the group plan. If the range of feasible durations of a group plan is tightened (the lower bound is increased or the upper bound is decreased), then the HR updates the edges of the corresponding group activity by once again calling UPDATE_GROUP_ACTIVITIES. This is done in Line 14. Note that tightening the constraints of the group activities only serves to remove uncertainty from the mission plan. Thus, the update performed in Line 14 only serves to make the decoupling algorithm more likely to succeed.

In our simple example, both of the group plans are dynamically controllable. Furthermore, feasible durations of the group plans are unchanged by the DC algorithm; therefore, the UPDATE_GROUP_ACTIVITIES call in Line 14 of the HR algorithm does not change the mission plan.

In Line 15, the HR algorithm calls the decoupling algorithm on the mission plan. The decoupling algorithm fixes the schedule for the start of each group plan. If the decoupling algorithm succeeds, then the HR algorithm returns true; otherwise, the HR algorithm returns false.

The Decoupling Algorithm

In this section we describe the *decoupling algorithm*, which temporally decouples each group activity in the mission plan. The effect of decoupling the group activities in the mission plan is that each group plan may be scheduled independently. The simplest method to perform this decoupling is to use a slight variation of the strong controllability algorithm, introduced by (Vidal 2000). Figure (**?**) shows the decoupling procedure. First, the strong controllability algorithm decouples the executable timepoint from the contingent timepoints, by making all requirement edges, connecting contingent timepoints, dominated (redundant). Next, the decoupling algorithm selects a consistent assignment to the executable timepoints in the mission plan.

This decoupling algorithm operates on the mission plan, in order to generate a fixed schedule for the start timepoint of each group activity. These fixed start times are then passed to their respective group plans. The resulting group plans can be scheduled independently. The decoupling builds upon the strong controllability checking algorithm (Vidal & Fargier 1999). The decoupling algorithm transforms the distance graph of the mission plan using the strong controllability transformation rules. If this transformed graph is consistent, the decoupling algorithm generates a schedule for the timepoints of the transformed graph. Note that any consistent schedule would work; however, we elect to schedule the group activities as early as possible. This schedule is used to fix the time of the corresponding group plans.

The pseudo-code for the decoupling algorithm is shown in Figure 6. The algorithm takes in a two-layer plan, consisting of a mission plan, M, and a set of group plans, G and fixes the schedule for the mission plan.

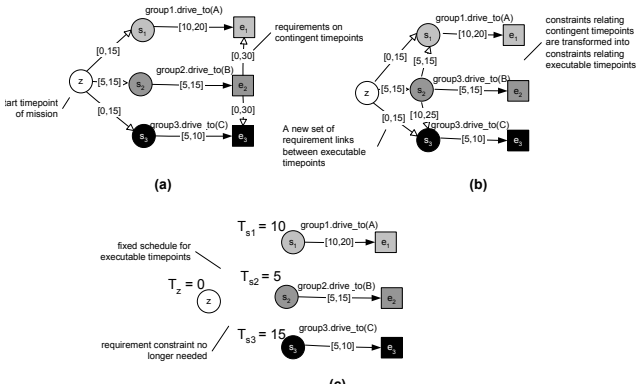

Figure 5: (a) The original mission plan containing requirement edges connecting contingent timepoints. (b) The mission plan after the contingent timepoints are decoupled by the strong controllability algorithm. Note, all requirement edge connecting contingent timepoints are removed. (c) The decoupling algorithm fixes the start time for each executable timepoints. This eliminates the need to propagate scheduling times during execution.

The decoupling algorithm runs in polynomial time. Lines 1-13 run in the same time as the strong controllability algorithm (i.e. O(NE)). In Lines 14-20, the decoupling algorithm loops through each timepoint and fixes the start time of each group plan. Using a simple lookup, the GET_GROUP_ACTIVITY and GET_GROUP_PLAN run in time linear in the number of group plans. Therefore, Lines 14-20 run in O(NG), where G is the number of group plans. The number of group plans G is less than the number of edges in the distance graph; therefore, the decoupling algorithm is dominated by the Bellman-Ford SDSP computation. The running time of the decoupling algorithm is O(NE).

For our simple example, the decoupling algorithm succeeds. The decoupling algorithm is applied to the updated mission plan, as shown in Figure 14(b). The distance graph of the mission plan is converted into the transformed STN, as shown in Figure 15(a). The decoupling algorithm first copies over the executable timepoints, A and C, then it transforms the edges, using the strong controllability transformation rules. The decoupling algorithm copies over the requirement edges AC = 1 and CA = 0 from the mission plans distance graph. The edge CB = 8 is transformed into an edge CA = -1, which relaxes the edge CA in the transformed STN. The edge BC = 0 is transformed into an edge AC = 2, which is greater than the existing edge, so there is no change in the transformed STN. Finally, the decoupling algorithm computes the earliest execution time for each timepoint, using an SDSP computation. The earliest execution time for A = 0 and B = 1, and, therefore, the start timepoint associated with group plan1, is fixed at 0, and the start timepoint for group plan2 is fixed at 1. The decoupled group plans are shown in Figure 6(b,c).

Discussion

After running the HR algorithm on the two-layer MTPNU, each group is able to efficiently execute the plan by using

Alg. 4 DECOUPLE(M,G) $Input$: A mission plan M and a set of group plans G. $Effects$: Decouples the group plans by fixing the start time of each group plan. $Output$: $True$ mission plan is strongly controllable; otherwise, $False$.

```
1: G_m ← get distance graph of mission plan
2: copy all executable timepoints of G_m to T
3: initialize all edges of T to NIL
4: for each requirement edge (u,v) ∈ E[G_m] do
5:    transform the edge (u,v) using SC transformation
      rules to and edge (u',v') with d(u',v') = x
6:    UPDATE_EDGE(T,u',v',x)
7: end for
8: s ← start timepoint of T
9: consistent ← BELLMAN_FORD_SDSP(T,s)
10: if ¬ consistent then
11:    return FALSE
12: else
13:    for each timepoint n ∈ N[T] do
14:       m ← GET_GROUP_ACTIVITY(n)
15:       if m ¬ NIL then
16:          g ← GROUP_PLAN(m)
17:          fix start time of g to -d[n] as computed by line
             10
18:       end if
19:    end for
20: end if
21: return TRUE
```

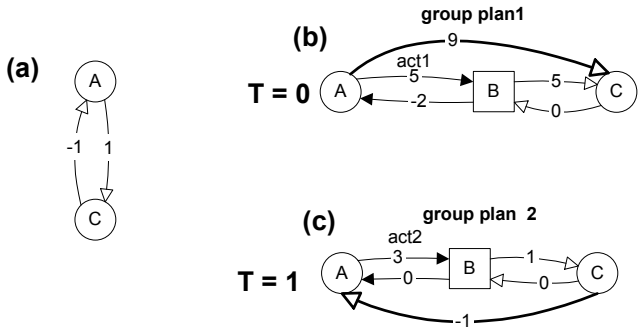

Figure 6: (a) The transformed STN (b) The start time of group plan1 is fixed at T = 0 (c) The start time of group plan2 is fixed at T = 1.

the dispatching algorithm presented by (Morris, Muscettola, & Vidal 2001).

The HR algorithm is a polynomial time algorithm. It gains efficiency by dividing the reformulation problem into a set of smaller sub-problems.

Consider the runtime complexity of the HR algorithm. In this discussion, we use the following notation.

- G = number of group plans.
- N_m = number of timepoints in the mission plan.
- E_m = number of edges in the mission plan.
- N_g = maximum number of timepoints in any group plan.
- E_g = maximum number of edges in any group plan.

In Line 1, HR calls the UPDATE_GROUP_ACTIVITIES function. The UPDATE_GROUP_ACTIVITIES function loops through each group plan and the time of each loop is dominated by the Bellman-Ford algorithm. Therefore, the UPDATE_GROUP_ACTIVITIES runs in O($G * N_g * E_g$). Lines 2-3 of the HR algorithm run in constant time. In Line 4, the HR algorithm calls COMPUTE_APSP_GRAPH. The Floyd-Warshall algorithm is used, which runs in (N_m^3). Lines 5-6 run in constant time. Line 7 calls the UPDATE_GROUP_PLANS function. The UPDATE_GROUP_PLANS function loops through each group activity and each loop is performed in constant time. Therefore, the UPDATE_GROUP_PLANS runs in O(G) time. Lines 8-13 of the HR algorithm loop through each group plan and calls the DC algorithm.

The time complexity of the DC algorithm is polynomial (Morris, Muscettola, & Vidal 2001) however, experimental results exhibit a running time of O(N_g^3). Given this, the running time of Lines 8-13 is experimentally shown to be O($G * N_g^3$). Line 14 calls the UPDATE_GROUP_ACTIVITIES function. Finally, in Line 15, the HR algorithm calls DECOUPLE, which runs in O($G * N_g$) time.

Adding the terms together, we get an expression for the running time of the HR algorithm as O($G * N_g * E_g$) + O(N_m^3) + O(G) + O($G * N_g^3$) + O($G * N_g$) + O(1), which can be simplified to O($G * N_g^3 + N_m^3$). The N_g^3 term is derived by an All-Pairs Shortest-Path (APSP) computation, applied to the group plan used in the DC algorithm. The N_m^3 term is due to the APSP computation on the mission plan.

The HR algorithm, presented in this paper, is unique in its ability to cope with both communication limitations and temporal uncertainty, by combining properties of strong and dynamic controllability. We believe this paper lays out a framework that will enable multi-agent systems to manage communication limitations in a pragmatic way.

References

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1990. *Introduction to Algorithms*. Cambridge, MA: MIT Press.

Dechter; Meiri; and Pearl. 1995. Temporal constraint newworks. In *Artificial Intelligence*, 61–95.

Morris, P. H., and Muscettola, N. 1999. Managing temporal uncertainty through waypoint controllability. In *IJCAI*, 1253–1258.

Morris, P.; Muscettola, N.; and Vidal, T. 2001. Dynamic control of plans with temporal uncertainty. In *IJCAI*, 494–502.

Muscettola, N.; Morris, P.; and Tsamardinos, I. 1998. Reformulating temporal plans for efficient execution. In *Principles of Knowledge Representation and Reasoning*, 444–452.

Tsamardinos, I.; Muscettola, N.; and Morris, P. 1998. Fast transformation of temporal plans for efficient execution. In *AAAI/IAAI*, 254–261.

Vidal, and Fargier. 1996. Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *In Proc. of 12th European Conference on Artificial Intelligence (ECAI-96)*, 48–52.

Vidal, and Fargier. 1999. Handling contingency in temporal constraint networks: from consitency to controllabilities. In *Journal of Experimental and Theoretical Artificial Intelligence*, 23–45.

Vidal, T. 2000. Controllability characterization and checking in contingent temporal constraint newtorks. In *Proc. of Seveth Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'2000)*.