Combining Temporal Information by Collaborative Planning Agents*

Meirav Hadad[1] **Sarit Kraus**[1,2]
[1]Department of Mathematics and Computer Science
Bar-Ilan University, Ramat-Gan, 52900 Israel
{hadad, sarit}@macs.biu.ac.il
[2]Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742

Abstract

We present strategies for reasoning and combining temporal information of intelligent agents. The agents situated in a dynamic environment collaborate with other agents and interleave planning and acting. Our work, which is based on a formal model of collaborative planning, allows each agent to develop its timetable individually and to take into consideration various types of time constraints of its activities and of its collaborators. However, its individual timetable can be easily integrated with the activities of other agents. Thus, in contrast to other works on multi-agent systems, which suggest that either the team members maintain full synchronization by broadcasting messages or that a team leader determines the times of actions, our mechanism enables the individual agent to determine its time frame individually and to be integrated easily in the activities of other agents. We have developed a simulation of such environments and present simulation results comparing different methods of exchanging temporal information among team members.

Introduction

This paper considers the problem of temporal scheduling of actions in a cooperative activity under time constraints. Our work is based on the SharedPlan model of collaboration (Grosz & Kraus 1996) that supports the design and construction of collaborative systems. In order to carry out their cooperative activity, the agents perform collaborative planning that includes processes that are responsible for identifying recipes, assigning actions to agents, and determining the time of the actions. A recipe for an action consists of subactions which may be either *basic* actions or *complex* actions.

Basic actions are executable at will if appropriate situational and temporal conditions hold. A complex action can be either a single-agent action or a multi-agent action. In order to perform a complex action, the agents have to identify a recipe for it and there may be several known recipes for such an action. A recipe may include temporal constraints and precedence relations between its subactions. The general situation considering the actions without the associated constraints is illustrated in Figure 1, in which the leaves of the tree represent basic actions. We refer to this tree as "a complete recipe tree for α." For the agents to have achieved a complete plan, the values of the time parameters of the actions that constitute their joint activity must be identified in such a way that all the appropriate constraints are satisfied. Determining the time of the actions in a collaborative environment that we consider is complex because of the need to coordinate actions of different agents, the partiality of the plans, the partial knowledge of other agents' activities and the environment and temporal constraints. Furthermore, the temporal constraints require interleaving of planning and execution. In a former paper (Hadad & Kraus 2001) we briefly presented an algorithm for cooperative agents to determine the time of the actions that are required to perform their joint activity. In this paper we expand our former work and compare different methods for reasoning and combining temporal information in a team. The mechanism presented in this paper focuses on temporal scheduling. Thus, for simplification purposes, the planners do not take into consideration preconditions and effects.

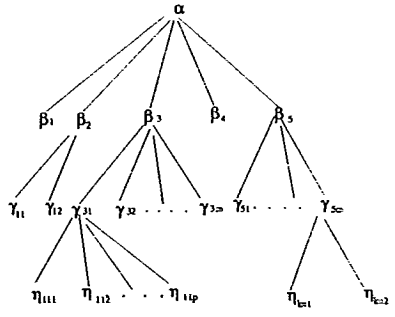

Figure 1: Recipe tree. The leaf nodes represent basic actions.

One of the main questions in multi-agent environment is at what stage an agent should commit itself to

*Preliminary version appeared in CIA-01. This work is supported in part by NSF under Grant No. IIS-9907482.

a timetable for performing a joint action, and inform the rest of the team of the time frame. If the individual agent commits itself to a specific timetable early and announces this commitment to the other agents, it may need to negotiate with the others if it needs to change its timetable later. Alternatively, if the commitment is made and announced as late as possible, i.e., only when requested by other team members, it may delay the planning and activity of these other team members. Another question arises regarding the order in which the team members should plan the subactions of their joint activity. If we force all of the team members to plan their activity together in the same order, the flexibility of the individuals is decreased. However, planning in different orders may delay the plan of some members of the group. This delay will occur when some members need certain information from other team members about a specific action that they want to plan, but they have to wait until the other members decide to plan this action too. In our work we have analyzed these questions by implementing different methods in a simulation environment.

An Algorithm for Temporal Reasoning

Building a collaborative agent that can flexibly achieve its goals in changing environments requires a blending of Real-Time Scheduling (RTS) computing and Artificial Intelligence Planning (AIP) technologies. Accordingly, our single-agent system consists of an AIP subsystem and an RTS subsystem. The goal of the AIP subsystems when the agents attempt to perform α is to identify a set of basic actions and a set of the temporal constraints associated with the basic actions, such that performing the basic actions under these temporal constraints constitutes performing α without conflict. As soon as it is identified, each basic action β is sent to the RTS subsystem, along with its temporal requirements $\langle D_\beta, d_\beta, r_\beta, p_\beta \rangle$, where D_β is the **Duration time**, i.e., the time necessary for the agent to execute β without interruption; d_β denotes the **deadline**, i.e., the time before the β should be completed; r_β refers to the **release time**, i.e., the time at which β is ready for execution; and p_β denotes the **predecessor actions**, i.e., the set $\{\beta_j | (1 \leq j \leq n)\}$ of basic actions whose execution must end before the beginning of β.

In our system, the agents develop the plan of their joint activity collaboratively. This section presents the major constituents of the time reasoning mechanism, which is based on the mechanism for an individual agent presented in (Hadad *et al.* 2002). However, expansion of collaborative plans differs from the individual case (Grosz & Kraus 1996). When an agent acts alone, it determines how to perform an action (i.e., find a recipe) and then carries out the necessary subactions. When agents work together, both the formation or selection of the recipe and the carrying out of the subactions is distributed. The group must reach a consensus on how they are going to perform the action (i.e., on the recipe they will use) and on who is going to do the

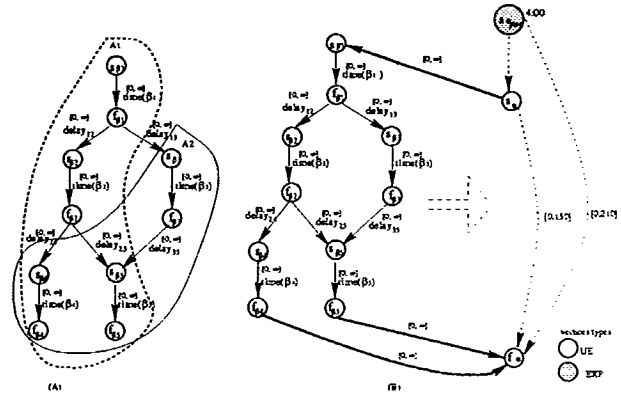

Figure 2: (A) An example of $Gr^p_{R_\alpha}$. (B) An example of adding a precedence graph to Gr^k_α.

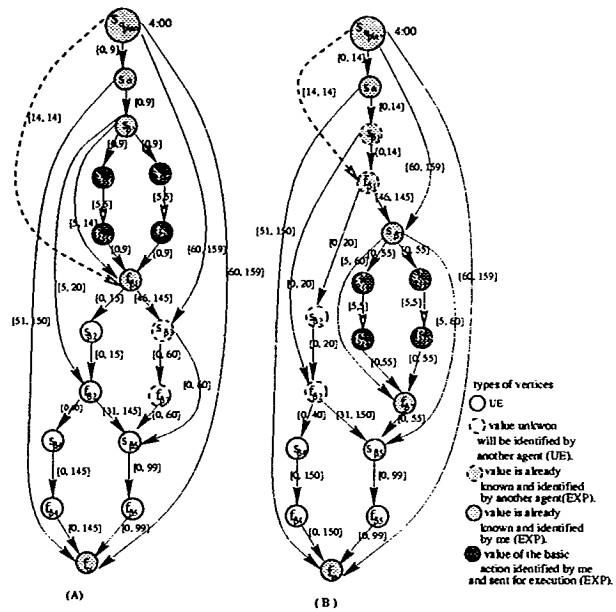

Figure 3: An example of temporal graphs Gr^k_α which are maintained by A_1 (graph A) and A_2 (graph B) during the planning of the joint activity α.

subactions. When each agent (or subgroup) decides on when to perform a subaction, the temporal constraints of the group members must be taken into consideration. Thus, recipe and agent selection in collaborative planning requires inter-agent negotiation and communication as well as individual agent reconciliation. In the mechanism presented, for simplicity, we assume that each action is associated with the agent that performed it. Thus, there is no need for decision-making concerning the performer of the action. Also, we do not discuss the recipe selection. We focus on the time scheduling problem and briefly describe some of its components.

Supporting Definitions and Notation

In (Hadad *et al.* 2002) we defined some basic concepts that were are used in the reasoning mechanism of the individual agent case. The main structure we defined was the *temporal constraints graph.* The agent uses this graph when it identifies the time parameters of its activity. As discussed in (Hadad *et al.* 2002), the form of this graph is based on *temporal constraint satisfaction networks* (Dechter, Meiri, & Pearl 1991, inter alia). However, because of the uncertainty and dynamic environment of the agents, in contrast to previous works, the constraints graph which is built by our agents may include only partial knowledge; i.e., our algorithm enables the agents to build this graph incremently and to backtrack when necessary. In addition, the agents are able to determine the actions for which the temporal parameters have been identified and to execute them, thus interleaving planning and execution.

In this section we define some new basic concepts and extend the former definition of the *temporal constraints graph* in such a way that it will also be appropriate for a cooperative multi-agent environment. The mechanism of the temporal reasoning in the multi-agent environment is very similar to that of the individual case. However, in contrast to a single-agent environment, in which all the vertices in the temporal constraints graph are associated with an action that has to be performed by the agent who develops the graph, in multi-agent activity, some vertices may be associated with an action which has to be performed jointly by a group of agents. Furthermore, some vertices may be associated with actions that have to be performed without the agent who develops the graph. We denote the performers of an action in the following definition:

Definition 1 *Let β be an action. The set $Ag_\beta = \{A_1 \ldots A_n\}$ $(n \geq 1)$, represents a group of agents who participate in the performance of β, where A_k, $(1 \leq k \leq n)$, is a single agent in the group.*

One of the main structures which is used in building the temporal constraints graph is the precedence graph. The *precedence graph* represents the precedence relationship between subactions in a recipe. The relationship may be between subactions that have to be performed by different agents. The precedence graph of action α in the multi-agent case is similar to that of the individual case and is given in the following:

Definition 2 *Let α be a complex action, and let R_α be a recipe which is selected for executing α. Let $\beta_1, \ldots, \beta_n$ be the subactions of the recipe R_α; $\theta^p_{R_\alpha} = \{(i,j) | \beta_i < \beta_j; i \neq j\}$ are the precedence constraints associated with R_α. The* precedence graph *of α, $Gr^p_{R_\alpha} = (V^p_{R_\alpha}, E^p_{R_\alpha})$ with reference to R_α and its precedence constraints $\theta^p_{R_\alpha}$ satisfies the following:*

1. *There is a set of vertices $V^p_{R_\alpha} = \{s_{\beta_1}, \ldots, s_{\beta_n}, f_{\beta_1}, \ldots, f_{\beta_n}\}$ where the vertices s_{β_i} and f_{β_i} represent the start time and the finish time points of β_i $1 \leq i \leq n$, respectively*[1].
2. *There is a set of edges $E^p_{R_\alpha} = \{(u_1, v_1), \ldots, (u_m, v_m)\}$ where each edge $(u_k, v_k) \in E^p_{R_\alpha}$ $1 \leq k \leq m$ is either;*
   (a) *the edge $(s_{\beta_i}, f_{\beta_i})$ that represents the precedence relations between the start time point s_{β_i} and the finish time point f_{β_i} of each subaction β_i. The edge is labeled by the time period for the execution of the subaction β_i; or*
   (b) *the edge $(f_{\beta_i}, s_{\beta_j})$, $i \neq j$, which represents the precedence relation $\beta_i < \beta_j \in \theta^p_{R_\alpha}$, that specifies that the execution of the subaction β_j can start only after the execution of the subaction β_j ended. This edge is labeled by the delay period between β_i and β_j. If $Ag_{\beta_i} \neq Ag_{\beta_j}$ the precedence relation "$\beta_i < \beta_j$" is called* multi-precedence constraint.
3. *Initially, all the edges of the precedence graph are labeled by $[0, \infty]$.*
4. *There is a set of vertices $\{s_{\beta_{b_1}}, \ldots, s_{\beta_{b_m}}\} \subseteq V^p_{R_\alpha}$ with in-degree 0. These vertices are called* beginning points.
5. *There is a set of vertices $f_{\beta_{e_1}}, \ldots, f_{\beta_{e_m}} \subseteq V^p_{R_\alpha}$ with out-degree 0. These vertices are called* ending points.

Example 1 *Figure 2(A) illustrates the precedence graph $Gr^p_{R_\alpha}$ of a possible recipe R_α. As shown in this figure the actions β_1 and β_2 are single-agent action and have to be planned by A_1. Thus, A_2, for example, cannot begin the execution of β_2 without knowing the temporal requirements of its precedent action β_1, which is planned by A_1. The vertex s_{β_1} is a beginning point and the vertices f_{β_4}, f_{β_5} are ending points.*

In the individual reasoning mechanism, we distinguish between complex and basic actions in the graph. A vertex which is associated with a basic-level action is called a *basic vertex* and a vertex which is associated with a complex level action is called *complex vertex.* Similarly, an edge which is associated with a basic-level action is called a *basic edge* and an edge which is associated with a complex level action is called *complex edge.* In a cooperative environment, a complex action may be either a multi-agent action or a single-agent action as we define in the following definition.

Definition 3 *Let $Gr^p_{R_\alpha} = (V^p_{R_\alpha}, E^p_{R_\alpha})$ be a precedence graph of an action α. Let $(s_{\beta_i}, f_{\beta_i})$ be an some edge in $E^p_{R_\alpha}$ which denotes the relation between the start time point, s_{β_i}, and the finish time point, f_{β_i}, of the subaction β_i.*

1. *If the subaction β_i is a single-agent action (i.e., $|Ag_{\beta_i}| = 1$), the edge $(s_{\beta_i}, f_{\beta_i})$ is called a* single-edge *and the vertices s_{β_i} and f_{β_i}, are called* single-vertices.
2. *If the subaction β_i is a multi-agent action (i.e., $|Ag_{\beta_i}| > 1$), the edge $(s_{\beta_i}, f_{\beta_i})$ is called a* multi-edge *and the vertices s_{β_i} and f_{β_i}, are called* multi-vertices.

[1] We will refer to a node which is labeled by s as node s.

3. *We say that the actions* $\{\beta_{j_1}, \ldots, \beta_{j_m}\}$ hinder β_i *if there is a set of of edges* $E' \subseteq E^p_{R_\alpha}$ *such* $E' = \{(f_{\beta_i}, s_{\beta_{j_1}}), \ldots, (f_{\beta_i}, s_{\beta_{j_m}})\}$, *where* $Ag_{\beta_i} \neq Ag_{\beta_{j_k}}, (1 \leq k \leq m)$.

In following we expand the definition of the *temporal constraints graph* for a multi-agent case. When the group Ag_α works on the action α, each agent $A_k \in Ag_\alpha$ maintains a temporal constraints graph. This graph is changed during the agents' planning. The graph of each individual agent in Ag_α my be different with respect to its individual actions, but similar with respect to the first level of multi-agent actions.

Definition 4 *Let* A_k *be an agent. Let* $\{\alpha, \beta_1, \cdots, \beta_n\}$ *be a set of basic and complex-level actions that the group* Ag_α *intends to perform, where* α *is the highest level action in the recipe tree which consists of these actions and* $A_k \in Ag_\alpha$. *Let* $V^k_\alpha = \{s_\alpha, s_{\beta_1}, \cdots, s_{\beta_n}, f_\alpha, f_{\beta_1}, \cdots, f_{\beta_n}\} \cup \{s_{\alpha_{plan}}\}$ *be a set of variables where the variable* $s_{\alpha_{plan}}$ *represents the time point that* A_k *starts to plan action* α. *Let* $\theta^p_{R_\alpha} = \{(\beta_i, \beta_j) | \beta_i < \beta_j; i \neq j\}$ *be the set of all the precedence constraints which are associated with all the recipes of the actions* $\{\alpha, \beta_1, \cdots, \beta_n\}$. *Let* $\theta^m_\alpha = \{v_i, v_j | a_{ij} \leq (v_i - v_j) \leq b_{ij}\}$ *be all the temporal metric constraints that are associated with* $\{\alpha, \beta_1, \cdots, \beta_n\}$ *and their recipes. The temporal constraints graph of* α, $Gr^k_\alpha = (V^k_\alpha, E^k_\alpha)$, *satisfies the following:*

1. *The vertex* $s_{\alpha_{plan}} \in V^k_\alpha$ *denotes the time point at which* A_k *starts to plan action* α. *This point is a fixed point in time which is called the* **origin**. *The vertex* s_α *is the time point at which* A_k *starts* α*'s performance, and* f_α *is the time point at which* A_k *finishes* α*'s performance.*
2. *If* α *is a basic-level action, then* $Ag_\alpha = \{A_k\}$ *and* $V^k_\alpha = \{s_{\alpha_{plan}}, s_\alpha, f_\alpha\}$. *The set of edges* $E^k_\alpha = \{(s_{\alpha_{plan}}, s_\alpha), (s_\alpha, f_\alpha), (s_{\alpha_{plan}}, f_\alpha)\}$, *where each edge* $(v_i, v_j) \in E^k_\alpha$ *is labeled by the following weight:*

$$weight(v_i, v_j) = \begin{cases} [a_{ij}, b_{ij}] & \text{if } v_i, v_j \in \theta^m_\alpha \\ [0, \infty) & \text{if } v_i, v_j \notin \theta^m_\alpha \end{cases}$$

3. *If* α *is a complex-level action then:*
   (a) *There is a set of* **basic vertices** $V_{basic} = \{s_{\beta_{b_1}} \cdots s_{\beta_{b_k}}, f_{\beta_{b_1}}, \ldots, f_{\beta_{b_k}}\} \subseteq V^k_\alpha$ *and a set of* **basic edges** $E_{basic} = \{(s_{\beta_{b_1}}, f_{\beta_{b_1}}), \ldots, (s_{\beta_{b_k}}, f_{\beta_{b_k}})\} \subseteq E^k_\alpha$, *where* β_{b_i} $(1 \leq i \leq k)$ *is some basic-level action in the recipe tree of* α, *and each edge* $(s_{\beta_{b_i}}, f_{\beta_{b_i}})\} \in E^k_\alpha$ *is labeled by the time period of* β_{b_i}. *Each such basic edge is a single-edge and each basic vertex is a single-vertex. If* $Ag_{\beta_{b_i}} \neq \{A_k\}$ *then* β_{b_i} *is a subaction in a recipe of a multi-agent action* β' *such* $A_k \in Ag_{\beta'}$.
   (b) *There is a set of* **complex vertices** $V_{comlex} = \{s_{\beta_{c_1}}, \ldots, s_{\beta_{c_l}}, f_{\beta_{c_1}}, \ldots, f_{\beta_{c_l}}\} \subseteq V^k_\alpha$ *and a set of* **complex edges** $E_{complex} = \{(s_{\beta_{c_1}}, f_{\beta_{c_1}}), \ldots, (s_{\beta_{c_l}}, f_{\beta_{c_l}})\} \subseteq E^k_\alpha$ *where each edge* $(s_{\beta_{c_j}}, f_{\beta_{c_j}}) \in E_{complex}$, $1 \leq j \leq l$ *is labeled by the following weight:*

$$\begin{cases} [a_{ij}, b_{ij}] & \text{if } (s_{\beta_{c_j}}, f_{\beta_{c_j}} \in \theta^m_\alpha \text{and} \\ & \beta_{c_j}\text{'s period is unknown} \\ \beta_{c_j}\text{'s period} & \text{if } (s_{\beta_{c_j}}, f_{\beta_{c_j}} \notin \theta^m_\alpha)\text{and} \\ & \beta_{c_j}\text{'s period is known} \\ [a_{ij}, b_{ij}] \cap (\beta_{c_j}\text{'s period}) & \text{if } (s_{\beta_{c_j}}, f_{\beta_{c_j}} \in \theta^m_\alpha)\text{and} \\ & \beta_{c_j}\text{'s period is known} \\ [0, \infty) & \text{if } (s_{\beta_{c_j}}, f_{\beta_{c_j}} \notin \theta^m_\alpha)\text{and} \\ & \beta_{c_j}\text{'s period is unknown} \end{cases}$$

   Each complex edge is either a single-edge or a multi-edge. Similarly, each complex vertex is either a single-vertex or a multi-vertex. If $A_k \notin Ag_{\beta_{c_j}}$ *then* β_{c_j} *is a subaction in a recipe of multi-agent action* β' *such that* $A_k \in Ag_{\beta'}$.
   (c) *There is a set of* **delay edges** $E_{delay} = \{(u_1, v_1), \ldots, (u_n, v_n)\} \subseteq E^k_\alpha$ *where* $u_i, v_i \in V^k_\alpha$, $(1 \leq i \leq n)$. *The vertices* u_i, v_i *may be of the following forms:*
      i. u_i *is the start time point of some action* β_u *and* v_i *is some beginning point in the precedence graph* $Gr^{time_p}_{\beta_u}$ *of action* β_u.
      ii. u_i *is some ending point in the precedence graph* $Gr^{time_p}_{\beta_v}$ *of some action* β_v *and* v_i *is the finish time point of action* β_v.
      iii. u_i *is the finish time point of some action* β_u *and* v_i *is a start time point of another action* β_v, *where* $\beta_u, \beta_v \in \theta^{time_p}_{R_\alpha}$.

   Each edge $(v_i, v_j) \in E_{delay}$ *is labeled by the following weight:*

$$weight(v_i, v_j) = \begin{cases} [a_{ij}, b_{ij}] & \text{if } v_i, v_j \in \theta^m_\alpha \\ [0, \infty) & \text{if } v_i, v_j \notin \theta^m_\alpha \end{cases}$$

   (d) *There is a set of directed edges* $E_{metric} = \{(u_1, v_1), \ldots, (u_n, v_n)\} \subseteq E^k_\alpha$ *labeled by its metric constraints, where* $u_i, v_i \in V^k_\alpha$, $(1 \leq i \leq n)$ *but* $(u_i, vi) \notin E_{basic} \cup E_{complex} \cup E_{delay}$.

The algorithm in general

During the agents' planning process, each agent A_k selects a vertex s_β from its Gr^k_α. For this vertex, which is associated with the action β, the planning of all the actions which precede β have been completed. Then, A_k attempts to complete β's plan. Action β is planned by A_k only if A_k participates in β's performance (i.e., $A_k \in Ag_\beta$). If s_β is a basic vertex, then the action β along with its temporal requirements $\langle D_\beta, d_\beta, r_\beta, p_\beta \rangle$ is sent to A_k's RTS subsystem for execution. Otherwise, A_k performs the following major activities:

(1) If s_β is a multi-vertex, then A_k has to obtain an agreement form the other members of Ag_β that A_k will develop β's plan.

(2) A_k finds an appropriate recipe, R_β, for the complex action, β.

(3) A_k constructs the precedence graph $Gr^p_{R_\beta}$ for β and incorporates it along with the other temporal constraints of R_β in Gr^k_α. $Gr^p_{R_\beta}$ is incorporated in Gr^k_α by inserting directed edges from s_β into the beginning points of $Gr^p_{R_\alpha}$, and by inserting directed edges from the ending points of $Gr^p_{R_\alpha}$ into f_β (see figure 2(B)).

(4) A_k checks the consistency of the updated temporal graph and determines the new values of the vertices in Gr^k_α by using an algorithm such as the Floyd-Warshall algorithm for solving a temporal constraints problem (see (Hadad *et al.* 2002)).

(5) If s_β is a multi-vertex A_k sends the information about the selected recipe to the other participants. If the others agree about the selected recipe, each of them performs the activities **(3)-(4)**.

(6) If Gr^k_α is either not consistent or if a some member in Ag_β objects to the selected R_β, then A_k searches for a different recipe for β, R'_β, and replaces the information it added in **(3)** with the information associated with R'_β and checks consistency again. If such R_β does not exist, it backtracks.

(7)A_k obtains messages from the RTS about the execution of basic-level actions and updates the timing of the appropriate actions.

(8)A_k obtains messages from the other team members and updates the graph with respect to the information that is listed below.

The development of a shared plan between the agents requires information exchange. Consider the case of two agents, A_i and A_j, that intend to perform a joint activity α. According to the algorithm sketched above, they will exchange information in the following cases: (1) When they identify a recipe for their joint action or for any joint subaction in their plan. This exchange of messages may be about possible recipes and also may be part of their group decision-making. (2) Agent A_i may inform agent A_j about the time values that it identified for its individual actions $\beta_1 \ldots \beta_m$ when $\beta_1 \ldots \beta_m$ *hinder* the planning of action γ and γ has to be performed by A_2. We assert that $\beta_1 \ldots \beta_m$ *hinder* the planning of γ if they directly precede γ (see definition 3). (3) When agent A_i completes the plan of a subaction in a recipe of a joint action with A_j, it informs A_j that the plan for this action has been completed (but does not send the details of the plan). (4) If A_i has already sent information to A_j about some action β, but it failed to perform β, then A_i backtracks and informs A_2 about the failure of β or about new properties of their plan that were determined as a result of its backtracking.

Example

Our example comes from the rescue domain. We consider the collaborative planning that arises when two robots, whom we call A_1 and A_2, have to "*rescue disaster survivors in areas A and B*" together. A_1 is a small robot who is able to maneuver deep into crevices in the rubble, flatten itself to crawl through tight spaces, or rear up to climb and look over objects. The second robot, A_2, is a large and strong robot. It is designed to pick up concrete slabs, pipes, broken boards and etc., in the collapsed area. Suppose that A_1 and A_2 arrived at the collapsed area at 4P.M. Thus, both agents start the collaborative plan for the joint action "*rescue disaster survivors in areas A and B*" together at 4P.M. As a result, this action can be started only after 4P.M. Suppose that the batteries of A_2 are restricted for 150 minutes; thus this action must be finished within 150 minutes. We also assume that this action must be finished before the evening (at 7:30P.M.). If we denote the action "*rescue disaster survivors in areas A and B*" by α, and its time interval by $[s_\alpha, f_\alpha]$, the temporal constraints as extracted from the example are the following: $\{(f_\alpha - s_\alpha \leq 150), (s_\alpha \text{ after } 4:00), (f_\alpha \text{ before } 7:30)\}$. Note that the length of the time interval $[s_\alpha, f_\alpha]$ which denotes the time of α execution is unknown initially. Both agents start the collaborative plan for α together at 4P.M., thus the fixed time point $s_{\alpha_{plan}} = 4P.M.$. At this stage, the vertices set of the temporal graph which is maintained by each agent is $V^k_\alpha = \{s_{\alpha_{plan}}, s_\alpha, f_\alpha\}$, and the edges set is $E^k_\alpha = \{(s_{\alpha_{plan}}, s_\alpha, [0,\infty]), (s_\alpha, f_\alpha, [0,150]), (s_{\alpha_{plan}}, f_\alpha, [0,210])\}$. After determination of the new values of the vertices, the new intervals of the edges in the example will be $E^k_\alpha = \{(s_{\alpha_{plan}}, s_\alpha, [0,210]), (s_\alpha, f_\alpha, [0,150]), (s_{\alpha_{plan}}, f_\alpha, [0,210])\}$ (see the dashed edges in Figure 2). Next, they need to agree on how they are going to perform the action α (i.e., on a recipe for α). Suppose that the agents have agreed upon the following plan: A_1 will "*find and identify the victims in area A*" - we denote this subaction as β_1. A_1 will "*find and identify the victims in area B*" - we denote this subaction as β_2. A_2 will "*carefully clear the piles of rubble block doorways in area A*" in order to rescue the victims outside the building- we denote this subaction as β_3. A_1 and A_2 together will "*rescue the victims outside the building in area A*" - we denote this subaction as β_4. A_1 and A_2 will "*rescue the victims outside the building in area B*" together - we denote this subaction as β_5. In addition, we assume that β_1 must be carried out before β_2, since A_2 must know about the location of the "victims" before it begins to clear block doorways in the area. Also, since area B is less dangerous than area A, area B will be scanned by A_1 only after area A (i.e., β_1 will be performed before β_2). In addition, they will "*rescue the victims outside the building in area A*" only after A_2 clears the block doorways in area A (i.e., will perform β_3 before β_4), and after A_1 "*finds and identifies the victims in area B*" (i.e., will perform β_2 before β_4). They will "*rescue the victims outside the building in area B* (β_5) after A_1 "*finds and identifies the victims in area B*" (β_2 before β_5).

Thus, the identified recipe R_α consists of the subactions $\{\beta_1 \ldots \beta_5\}$ where β_1 and β_2 will be performed only by A_1, β_3 will be performed only by A_2 and β_4,

and β_5 are multi-agent actions. In addition, this identified recipe is associated with the precedence relations $\{\beta_1 < \beta_2;\ \beta_1 < \beta_3;\ \beta_2 < \beta_4;\ \beta_2 < \beta_5;\ \beta_3 < \beta_5\}$. Suppose that the temporal constraints for performing these actions are: $\{(f_{\beta_2} - s_{\beta_1} \leq 20), (s_{\beta_4} - f_{\beta_2} \leq 40), (f_{\beta_5} - s_{\beta_3} \leq 60), (s_{\beta_3} \text{ after } 5:00)\}$. Each agent builds the $Gr^p_{R_\alpha}$=($V^p_{R_\alpha}$,$E^p_{R_\alpha}$) and incorporates it into its Gr^k_α by adding directed edges from s_α to each vertex v_{b_1}, $ldots v_{b_m} \subseteq V^p$, with an indegree of 0, and by adding edges from each vertex $v_{e_1}, \ldots v_{e_m} \subseteq V^p_{R_\alpha}$, with an outdegree of 0 to f_α. A pictorial illustration of adding this $Gr^p_{R_\alpha}$ to Gr^k_α is given in Figure 2. $Gr^p_{R_\alpha}$ is incorporated into Gr^k_α by adding the edges $(s_\alpha, s_{\beta_1}), (f_{\beta_4}, f_\alpha), (f_{\beta_5}, f_\alpha)$. After this, the agents also update the temporal constraints that are associated with R_α.

Then, each agent A_k tries to continue the development of its Gr^k_α by selecting an action to be developed. At this stage, Gr^1_α is identical to Gr^2_α but only the action β_1 may be selected since it does not depend on any other action. Since A_1 is the single-agent involved with β_1, A_2 has to wait until it receives the values of the temporal parameters of β_1 from A_1. Suppose that the recipe that A_1 selected for β_1 consists of the basic actions γ_{11} and γ_{12}, and that the execution time for each of them is exactly 5 minutes. Thus, A_1 should identify $\langle D_{\gamma_{11}}, r_{\gamma_{11}}, d_{\gamma_{11}}, p_{\gamma_{11}}\rangle$ and $\langle D_{\gamma_{12}}, r_{\gamma_{12}}, d_{\gamma_{12}}, p_{\gamma_{12}}\rangle$ and send it to its RTS subsystem. The decision regarding the exact time in which β_1 will be executed is determined by the RTS subsystem. In this example, we assume that the RTS subsystem of A_1 decides to execute γ_{11} at 4:02 and γ_{12} at 4:09. Thus A_1 will inform A_2 that it intends to terminate the execution of β_1 at 4:14. Following this announcement, A_2 can start developing β_3. Similarly, A_1 can develop β_2. A_2 will use this information to develop the plan for β_3. In addition, both of them will add the edge $(s_{\alpha_{plan}}, f_{\beta_1})$ and will label this constraint on it (i.e., the weight of this edge will be [14, 14]). The temporal graph Gr^k_α which is maintained by each agent in this stage of their collaborative plan is given in Figure 3. As shown in this figure, each agent maintains a different graph according to its plan. Graph (A) is the graph which is built by A_1 and graph (B) is built by A_2. The identical vertices represent the subactions which appear in the recipe of their collaborative action.

Possibilities for Team Scheduling and Planning

As described above, agent A_i informs agent A_j about the time values that it identified for its individual actions $\beta_1 \ldots \beta_m$ when $\beta_1 \ldots \beta_m$ *hinder* the planning of action γ that has to be performed by A_j. The problem is at what stage A_i should commit itself to a timetable for performing $\beta_1 \ldots \beta_m$, and inform A_j about this timetable.

One possibility is that when A_i finishes the planning of all its actions that directly precede γ, it commits itself to their performance times and sends the appropriate values to A_j. This method enables A_j to begin the plan of γ immediately when the planning of all the actions which precede γ have been completed. Furthermore, A_j does not need to ask A_i for the relevant times, since A_i informs A_j about them as soon as possible. However, since A_i has to commit itself to these times, A_i has less flexibility in determining the time performance for its other actions. If it decides to change the announced timetable, it will need to negotiate with A_j. Thus, we suggest an alternative mechanism. Following this approach, A_j plans its individual actions until it is no longer able to continue its plan, since it depends on another team member A_i. In such a case, A_j sends a message to A_i and asks A_i for the appropriate time values. When A_i completes the planning of the relevant actions, it sends the appropriate values to A_j. In this manner the commitment is left to the latest time possible, but, it may delay the planning of the agent waiting for an answer and it requires additional message exchange. In our terminology the first method is called *provide-time* and the second is called *ask-time*.

Another problem involves the order in which the members of the group should plan their activities. As mentioned above, each agent A_k in the group has to select an action β from its Gr^k_α during the planning process. Action β is selected by A_k only if it satisfies certain conditions. However, since in most cases there is more than one action that satisfies all the required conditions, the agent has to decide which of them to select. There are several possibilities for the methods of selection. In this work we consider three methods. The first is called *random-order*, where A_k randomly selects one of the several actions that may be developed. In the second, called *dfs-order*, A_k selects the action according to the depth-first search order of Gr^k_α. The third is called *bfs-order*. According to this method, A_k selects one of the actions according to the breadth-first search order of Gr^k_α. In the first method, the planning order of the team members differ. In the two latter methods, all the agents in the team plan their actions in the same order. In the dfs-order, the agent selects an action from the lowest levels of the recipe tree. Thus, it develops a subaction until it reaches the basic actions level; then it continues to the next subaction. In the bfs-order, the agent tries to complete the plans of the highest level actions in each stage of the planning process of the recipe tree. The following section presenting the simulation results shows that the planning order influences the success rate of the agents.

Experimental Results

We have developed a simulation environment comprising 2 agents to evaluate the success rate of the system when the agents use the provide-time method and when they use the ask-time method. Each method was tested when the agents used the random-order method, the dfs-order method, and the bfs-order method. The combined methods are called random-provide, dfs-provide,

bfs-provide, random-ask, dfs-ask, and bfs-ask, respectively. We have also compared these methods with a "central-planner." The central planner in our system planned all the actions centrally. During the planning process it sent the basic actions with their associated temporal constraints to the appropriate agents. The RTS subsystems of the appropriate agents handled the scheduling and execution of these actions.

In our experiments, we made the simplifying assumption that all of the time constraints of the agent's action are associated either with the action-type or with the appropriate recipe.[2] We ran the algorithm on several different recipe libraries which were created randomly. Each recipe library included at least one possible solution to the joint activity. For each case, we ran 120 experiments with randomly drawn parameters from ranges with high success rates of a single-agent. In each run, the following parameters were drawn from the following given range: (1) the average number of the *precedence constraints* between the actions that have to be performed by the same agent was between 0.20 and 0.55; (2) the average number of the *metric constraints* was between 0.9 and 1.1; (4) the number of the total *basic actions* in the recipe tree was between 100 and 120 (between 50 and 60 for each agent); the duration of each basic action could have been between 1 to 10 minutes; (5) the *idle time* in the schedule was between 120 and 170 seconds; (6) the number of the *multi-agent actions* in the recipe tree was 3. In addition, in all the runs we provided the schedule with a waiting time at the beginning of the schedule process (i.e., all the basic actions which arrive at the RTS subsystem have to begin 120 seconds (or more) after the schedule process has been started).

We tested the effects of the number of the *multi-precedence constraints* on complex subactions. We asserted that the group succeeded in performing the highest level action α if: (1) the AIP subsystem of each agent finished the planning for α and built the complete recipe tree for α which satisfies the appropriate time constraints θ_α; and if (2) the RTS subsystem of each agent found a feasible schedule, which consists of all.the basic level actions in the complete recipe tree of α. A failure is defined as either a failure in planning by the AIP subsystem of one member in the group, or as a failure in finding a feasible schedule by the RTS subsystem of one member.

Figure 4 shows the success rate of each method in a given range of multi-precedence constraints. As shown in the graph, the method with the best results is bfs-ask, with a success rate between 90% – 100%. The success rate of random-ask was between 86% – 96% and of the dfs-ask was between 75% – 95%. In general, the ask-time method is better than the provide-time method. We can see that bfs-provide (80% – 100% success rate) is better than random-provide (64% – 92% success rate) and dfs-provide (64% – 92%). Thus, we conclude that the bfs-order is the best planning method for our system. However, it is unclear if the random order is better than dfs-order. In addition, the dfs-provide is the sole method with a success rate that is a linear function of the number of the multi-precedence constraints. In the other methods, there are changes in the success rate of the system. We hypothesize that the reason for these changes results from the fact that a high number of multi-precedence constraints provides more knowledge about the subactions slots. As a result, the precedence constraints leads the scheduler and the other team members to the correct solution (which always exists in our examples). On the other hand, multi-precedence constraints decrease the flexibility of the individuals in the team since they cause them to make more commitments in their timetable. The low flexibility leads to a lower success rate in cases of 3-4 multi-precedence constraints. The central-planner attained good results in some cases (between 80% – 100%), but it was 5 times slower than all the other methods. Also, a high number of multi-precedence constraints reduces its success rate.

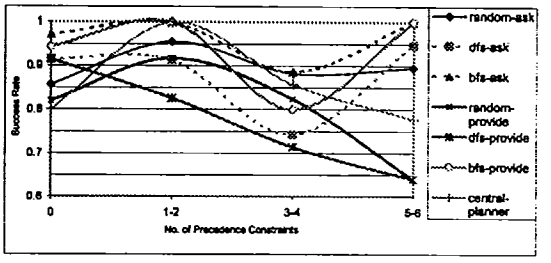

Figure 4: The success rate of the system in a given range of multi-precedence constraints.

Comparison with Alternative Works

Our effort focuses on developing a system which solves problems that cannot be solved by a central controller, but which should be solved by partitioning the problem and distributing it among several agents according to their skills. Each agent reasons individually while interacting with the other team members. The agent dynamically determines both the duration and the time frame of the actions it has to perform in such a way that these times satisfy all the appropriate temporal constraints. The times and the periods of the events that occur during the agents' activities need not be known in advance. That is, an agent may change its timetable easily if it identifies new constraints due to changes in the environment or if it receives new time constraints

[2] An example of a more complex scenario is one in which the agent receives new temporal constraints as a consequence of changes in its dynamic environment.

from other team members. Thus, unlike other collaborative multi-agent systems, which either suggest broadcasting messages between the team members to maintain the full synchronization (Sonenberg *et al.* 1992; Tambe 1997) or suggest that the team leader be responsible for the timing of the individual actions (Jennings 1995), our mechanism does not restrict the activity of the individual agents.

The ability of the agents to schedule their activities distributively, when the duration and the time frame of the actions need not be known in advance, is in contrast to the existing work on distributed scheduling, which assumes that these temporal requirements are known in advance. In particular, most of the existing research (e.g., (Neuendorf & Hannebauer 2000; Wellner & Dilger 1999)) focuses on the development of heuristic algorithms for solving different types of distributed scheduling problems which are known as NP-complete. Other works (e.g., (Rabelo & Camarinha-Matos 1998; Hunsberger 2002)) focus on negotiation and auction methods which enable the agents to reach a consensus on how to allocate a portion of their activity. Thus, existing research does not suggest any method for representing and reasoning about incomplete and indefinite qualitative temporal information that can be used during the planning process of the agents, as we do.

Planning under uncertainty contrasts with the situation of other planners, who rely on perfect domain knowledge throughout plan development and execution (e.g., (El-Kholy & Richards 1996)). Others, who can plan under uncertainty, are complex and cannot be easily extended for use in cooperative multi-agent environments (e.g., (Wilkins *et al.* 1995)). Our mechanism is simple and appropriate for uncertain, dynamic, multi-agent environments, and enables agents to reason about their timetables during the planning process and thus to interleave planning and acting. In multi-agent systems which are intrinsically dynamic, it is impractical to maintain this classical distinction between planning and executing (Paolucci, Shehory, & Sycara 2000).

Conclusions

In this paper, we have presented a mechanism for time planning in uncertain and dynamic multi-agent systems. Our mechanism allows each agent to develop its timetable individually and to take into consideration all types of time constraints of its activities and of its collaborators.

Through simulations, we have compared different methods for information exchange and planning order for scheduling in a team. The simulation results show that it is better to commit to a timetable as late as possible. The results have also proved that if the team agents perform their plan in the same order and if they first develop the highest level actions, then they have a better chance of succeeding in their joint activity. In addition, it is more efficient to distribute the planning among the group members than to solve the problem by a central planner.

References

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

El-Kholy, A., and Richards, B. 1996. Temporal and resource reasoning in planning: the parcPlan approach. In *ECAI '96*, 614–618.

Grosz, B. J., and Kraus, S. 1996. Collaborative plans for complex group action. *AIJ* 86(2):269–357.

Hadad, M., and Kraus, S. 2001. A mechanism for temporal reasoning by collaborative agents. In *CIA-01*, 229–234.

Hadad, M.; Kraus, S.; Gal, Y.; and Lin, R. 2002. Time reasoning for a collaborative planning agent in a dynamic environment. *Annals of Math. and AI.* (In press) www.cs.biu.ac.il/ sarit/articles.html.

Hunsberger, L. 2002. Generating bids for group-related actions in the context of prior commitments. ATAL-01 (to appear).

Jennings, N. R. 1995. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* 75(2):1–46.

Neuendorf, K. P., and Hannebauer, M. 2000. Formal modeling of multi-agent interaction in distributed scheduling. In *Proceedings of the 16th IMACS World Congress on Scientific Computation.*

Paolucci, M.; Shehory, O.; and Sycara, K. 2000. Interleaving planning and execution in a multiagent team planning environment. Technical Report CMU-RI-TR-00-01.

Rabelo, R. J., and Camarinha-Matos, L. M. 1998. Generic framework for conflict resolution in negotiation-based agile scheduling systems. In *IFAC Workshop on Intelligent Manufacturing Systems.*

Sonenberg, E.; Tidhar, G.; Werner, E.; Kinny, D.; Ljungberg, M.; and Rao, A. 1992. Planned team activity. Technical Report 26, Australian Artificial Intelligence Institute, Australia.

Tambe, M. 1997. Toward flexible teamwork. *Journal of AI Research* 7:83–124.

Wellner, J., and Dilger, W. 1999. Job shop scheduling with multiagents. In *13th Workshop on Plan and Konfigurieren (PuK).*

Wilkins, D. E.; Myers, K. L.; Lowrance, J. D.; and Wesley, L. P. 1995. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI* 7(1):197–227.