Analyzing Characteristics of Task Structures to Develop GPGP Coordination Mechanisms

Wei Chen
University of Delaware
Department of Computer & Information Sciences
Newark, DE 19716
wchen@cis.udel.edu

Keith S. Decker
University of Delaware
Department of Computer & Information Sciences
Newark, DE 19716
decker@cis.udel.edu

ABSTRACT

Previous research about multi-agent coordination has concentrated at a high level, e.g. developing communication protocols for coordination, constructing special purpose agents to dictate the coordination behaviors of an entire system, or associating rules or coordination mechanisms with every agent to achieve cooperative behaviors. Much less research addresses multi-agent coordination at a low level: evaluating the effects of agents' task structures upon agents' coordination behaviors. This paper presents an Extended Hierarchical Task Network (EHTN) to represent precisely those structural features that affect coordination. Using this EHTN formalism, an extended set of Generalized Partial Global Planning (GPGP) coordination mechanisms has been developed for multi-agent coordination. Each coordination mechanism is defined in terms of EHTN rewriting rules and an associated set of pre-defined EHTN behaviors. This set of GPGP coordination mechanisms has been applied to a simulated emergency medical service (EMS) system. The experimental results reveal some of the performance relationships between specific mechanisms and external environmental characteristics.

Keywords

Multi-Agent Coordination

1. INTRODUCTION

Well coordinated behaviors can greatly improve agent performance, while inappropriate coordination results in reduced system efficiency, unfinished tasks, misuse of key resources, and even system crashes. It is critical to understand how to achieve well coordinated behaviors for involving intelligent software agents. While much research has focused on high-level coordination approaches, less attention has been focused on the use of task structure detail for coordination.

In order to make use of task structural information, it is key to find a suitable way to represent agents' task structures. STRIPS-style planning systems were developed almost thirty years ago. Most current practical planning systems within the past decade have been based on Hierarchical Task Network (HTN) decomposition. The analysis of the syntax and semantics of HTNs has been well studied [5, 11, 13]. However, the traditional HTNs do not represent what is happening in dynamically changing systems. Particularly, traditional HTNs are not expressive enough to represent certain problems concerning worth-oriented goals, contingencies, and the uncertainties that arise when task plans are in fact distributed over multiple agents. There has been no adequate representation method to capture the task features useful for coordination purposes.

We have developed a highly expressive representation method, Extended Hierarchical Task Networks (EHTNs), to accurately describe the features of agents' tasks, such as information flow and control flow of a complicated task network, the dependency relationship between the tasks of involving agents, and effects of task execution results. We believe that task structural information is helpful for multi-agent coordination, i.e., agents' task structures can be analyzed and manipulated for managing the interdependency relationships among agents.

This paper further articulates a domain-independent approach for specifying an extended set of seventeen GPGP (Generalized Partial Global Planning) coordination mechanisms, which are recast using the EHTN formalism.

We chose a simulated emergency medical service (EMS) system to demonstrate the effectiveness of the extended set of GPGP coordination mechanisms. Based on the experimental results, quantitative and qualitative analysis has been carried out and significant conclusions have been presented to guide agents in selecting the best mechanisms in various environments.

The EHTN task representation language will be presented in Section 2. Our extended set of GPGP coordination mechanisms will be explained in Section 3. A simulated emergency medical service system is briefly introduced in Section 4. The experimental results of the application of these coordination mechanisms in EMS is discussed in Section 5. Finally, we state our conclusion and future work.

2. EXTENDED HIERARCHICAL TASK NETWORKS (EHTNS)

Following the definition of coordination as "managing dependencies between activities"[9], interdependency relationships among multiple agents introduce the problem of con-

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

tingency, uncertainty, and worth-oriented goal tradeoffs. Traditional HTNs do not have the language constructs to model the quantitative features of task execution results and the information flow and control flow of the task networks in a distributed environment. For example, one prerequisite input of a task, named TaskA, of Agent A is the result value of a task, named TaskB, of another agent, Agent B; in the absence of any additional information it is unlikely that TaskA in Agent A could be listed as a valid candidate plan step at all. Several approaches have tried to address parts of this problem [7, 10], however, there is no formal representation that is able to quantitatively denote worth-oriented goals, contingencies, and uncertainties distributed across multiple agents. In discussing the Markov decision process (POMDP) approach Boutilier[1] observes that in practice standard AI representations and algorithms survive because real "problems commonly possess structure ... [therefore] specialized representations, and algorithms employing these representations, can achieve computational leverage by exploiting these various forms of structure." It is such a practical approach to extend the existing traditional hierarchical task networks with enriched syntax, semantics, and reasoning capabilities to tackle the aforementioned coordination problem, merging features from TÆMS[12]; the HTN-style agent languages used by RETSINA[**?**] and DECAF[6]; and Erol's formalization of HTNs[5].

It is important to explicitly represent non-local tasks with EHTNs. If one agent's action(s) depends on another agents' action(s), we call this kind of relationship a *coordination relationship* or *interdependency*[4, 12]. An interdependency is *a relationship between a local task (of one agent) and a non-local task (of another agent) where the execution of one changes some performance-related characteristics associated with the other*. A non-local task denotes a task network from a remote agent and is the key to understanding and managing task interdependencies. If non-local tasks can be properly represented, identified, and analyzed, it is possible to develop coordination mechanisms to manipulate non-local tasks based on the task features.

We will briefly present Extended Hierarchical Task Networks (EHTNs) based on the well known work of Erol, et al. [5] and focus primarily on the differences. Note that we only concentrate on an agent's (partially) *local* view—primitive or compound tasks at other agents are represented locally as *non-local tasks* (NLTs). The vocabulary of the language $\mathcal{L}$ is a tuple $\langle V, C, P, F, T, N, NLT, I, O, A, CAF\rangle$, where V, C, P, F, T, N have the same meanings as defined by Erol [5]: V is an infinite set of variable symbols; C is a finite set of constant symbols; P is a finite set of predicate symbols; F is a finite set of *primitive* task symbols, T is a finite set of *compound* task symbols, and N is an infinite set of symbols used for labeling tasks. Our new language constructs are introduced as follows: NLT is a finite set of *non-local task* symbols; I is a finite set of input provision symbols; O is a finite set of outcome symbols including "OK" and "FAIL"; and A is a finite vector of domain-dependent task/action characteristics represented by text values, $\langle A_1, A_2, ..., A_n\rangle$, where A_i is the i^{th} attribute of a task/action (quality, duration, cost, completeness, reputation, etc.). An instance of attribute A_i is shown as a_i. For example, the duration of an information searching task using a specialized database search engine is one minute, the cost is $1.00, and the quality, or satisfaction rate, is 10; while using a free search engine, the corresponding characteristics are two minutes, $0 cost, and the satisfaction rate of 5. CAF is a finite set of characteristic accumulation functions (CAF); a CAF specifies how the value of an attribute A_i of a parent task is derived from the cumulative values of its direct child tasks. Typically functions are *MAX, MIN, SUM*, etc. For example, the cost of a task could be the *sum* of the costs of all its children; the duration of a task is the maximum (*max*) child duration in parallel execution and *sum* in serial execution; and those tasks that only need one child to be completed ("OR" tasks, or "OR" nodes) may have *max* as a quality-CAF, i.e. characteristic accumulation function on quality, etc. We modify Erol's HTN representation to include explicitly named input provisions and outcomes (representing possible contingencies). The information flow and control flow represented with EHTNs can be managed for coordination. Details about the syntax, semantics, and algorithms based on ETHNs are explained in [2]. In Figure 1,

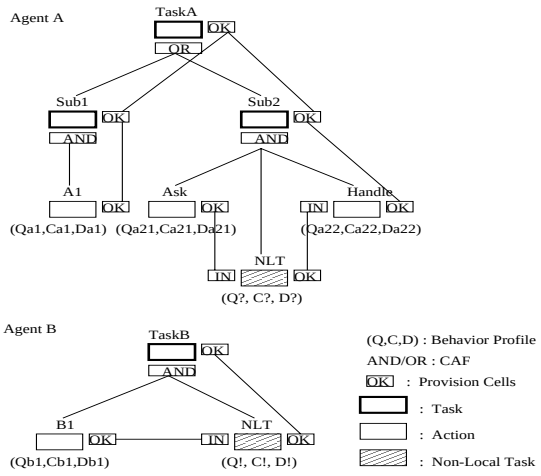

Figure 1: Example Agent Task Structure.

the task networks, TaskA and TaskB, belong to Agent A and Agent B respectively. The new language construct, NLT, of EHTNs means the set of non-local tasks; the task labeled "NLT" within the task network, TaskA, is an example of a non-local task. The cell labeled "IN" associated with the action, "Handle", is a provision, which indicates that the action can not be executed until a value is filled for this provision. The cell labeled "OK" is an output of the action, Handle, which refers to the execution result of this action. The information flow and control flow are indicated by the lines connecting the provisions and outcomes. For example, the outcome "OK" of the action, "Ask", connects to the provision cell, "IN", of the non-local task, "NLT"; the outcome "OK" of "NLT" connects to the provision "IN" of "Handle". The connections indicates that "NLT" will not be executed until "Ask" is finished; and "Handle" will not be executed until a value has been transported from outcome "OK" of "NLT" to "IN" of "Handle". The cell labeled "AND" associated with the sub-task, Sub2, dictates that all child tasks (Ask, NLT, and Handle) should be finished to reach the achievement of their parent task "Sub2". The vector, (Q,C,D), below each action specifies the quantitative features of that action. Notably, the quantitative features of a non-local task are uncertain, which results in the failure of

proper scheduling for Agent A. The existence of uncertainty is because the NLT actually represents the execution of a requested remote task, TaskB of Agent B in this case; Agent A has limited view about other agents and has no knowledge about TaskB. Certain coordination activities may be carried out so that the uncertainty can be removed, as explained in Section 3.

Our proposed coordination process has two steps: (1) locating the interdependencies in the task structure, and (2) applying the coordination mechanisms at those dependencies. We will introduce the application of mechanisms in Section 3. Here we explain the rules for locating interdependencies within agents' tasks.

Given that tasks are represented by our EHTNs, it is easy to detect interdependencies within agents' tasks structures. Any non-local tasks that meet the following conditions are interdependencies, or called *coordination points*, and require the application of a suitable coordination mechanism:[1]

- A non-local task, "NLT", exists within a task network;
- A provision of "NLT" is connected with an outcome of one child action, which shares the same parent task;
- An outcome of "NLT" connects to a provision of another child action of the same parent task.
- There is no other non-local tasks for the parent task.

With the explicit EHTN representation of coordination and the rules for detecting interdependencies among agents' tasks by analyzing structural features, algorithms have been developed for our agents to find out the needs for coordination.

3. AN EXTENDED SET OF GPGP COORDINATION MECHANISMS

Previously, we have introduced the detection of interdependencies by analyzing characteristics of agents' task structures. In this section, we will introduce an extended set of GPGP (Generalized Partial Global Planning) coordination mechanisms and the actual operations for these mechanisms.

Each coordination mechanism consists of (1) a set of coordination communication protocols and (2) a pattern-directed re-writing of our extended hierarchical task networks, EHTNs [2]. Figure 1 shows a general example, Agent A's action, Handle, can not be executed until action B1, is finished by Agent B, (B1 of Agent B enables Handle of Agent A). Thus, Agent B is an *enabler* (enabling the task of a remote agent) and also called a *predecessor*; Agent A is an *enablee* and called a *successor* as well. There are many potential coordination mechanisms for this situation. The coordination component in Agent A detects the interdependency relationships between these two agents (as explained in Section 2), and the task NLT with uncertain characteristics in Agent A is replaced (the task tree is rewritten) by a selected coordination mechanism which may include information communicated about *TaskB* from Agent B.

Our extended set of seventeen GPGP coordination mechanisms are briefly listed and explained as below. *Avoidance* and *sacrificial avoidance*: Agent A removes the branch containing the NLT if the CAF of TaskA is OR (with or without some sacrifice of system quality); *reservation*: Agent A reserves the execution of B1 by Agent B at a future time, so that Agent A is able to schedule its own activities in the mean time; predecessor-side commitments (Agent B): Agent B commits to do task B1 at sometime in the future (*predecessor notice at start*), to do it by a deadline (*predecessor deadline commitment*), to an earliest-start-time (*predecessor EST commitment*), or to notify when complete (*predecessor sending result*); simple successor-side commitments (Agent A): Agent A commits to do the action, Handle, with or without a specific Earliest Start Time (EST) (*successor EST commitment*); Agent A commits to do the action by specifying a deadline (*successor deadline commitment*); polling approach: Agent A consistently queries about the execution of the enabling tasks, B1, (*polling for result*) or the schedule of Agent B (*polling for schedule*); *constant headway / timetabling*: Agent A schedules its tasks according to a constant pattern, e.g., either in a fixed periodic manner (every ten minutes) or following a timetable, e.g, catching a bus based on a published bus schedule; shifting task dependencies: either Agent A sends its task structure (Sub2) to Agent B (*promotion shift*), or Agent B sends B1 to Agent A (*demotion shift*), by learning or mobile code (promotion or demotion); third-party mechanisms: a third agent, Agent C schedules the tasks for both Agent A and Agent B (*third party coordinator*), or Agent C executes B1 for Agent B if Agent B is busy (*third party execution*); *bidding*: Agent A broadcasts a coordination request to all available agents for the engagement of coordination process; or other complex multi-stage negotiation strategies. Details about these coordination mechanisms will be explained in the simulated EMS system shown in Section 4.

Let us explain the operation of *reservation* mechanism as an example. This mechanism is named after the real world activity of reservations. For example, if you want to have dinner in a restaurant, before you go there you had better make a reservation so that at some agreed future time, you will be there and the people there will be ready to serve you to avoid the potential waiting time. Imagine Figure 2 with task $Sub1$ removed so that Agent A has no alternative choice. The reservation mechanism includes a new protocol, instantiated as new task structures (called "GPGP_ReservationR" at Agent B and "GPGP_ReservationE" at Agent A) that processes a message from Agent A indicating a request to do a task (B1 of Agent B) sometime in the future. The reply is a message from Agent B indicating whether the request is agreed or refused; and if agreed, when (if ever) the task will occur. The reservation mechanism rewrites the task structure for Agent A so that a new subtask ("GPGP_ReservationE") is executed and invokes the "GPGP_ReservationR" at Agent B and then processes the return message. The result is an annotation on the non-local task that allows Agent A to predictively schedule task $Sub2$ and execute other potential tasks locally at a time before the enabling task (B1) has been completed.

This mechanism includes both a rewriting of the local task structure and some external communication protocol depicted in Figure 3. Only three rounds of message transfer are needed for the basic reservation process. We assume that there is no need to introduce a confirmation phase in this process. After the re-writing process, the new task structure for Agent A is very similar to that shown in Figure 2,

[1]Formalization of of interdependency detection is presented in [2] together with the full introduction of the EHTNs.

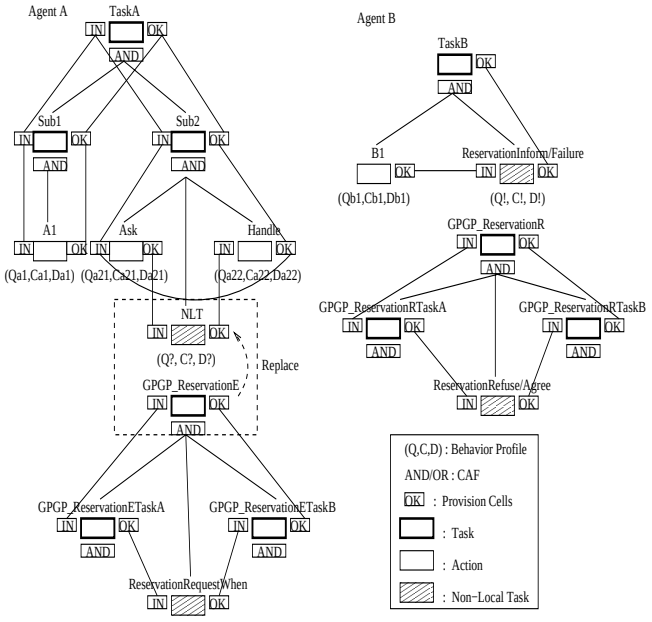

Figure 2: The domain tasks and the inserted GPGP mechanism – coordination by reservation.

except that the task NLT is replaced with the task network GPGP_ReservationE in the dashed box with links is altered. Details about the other mechanisms are in [2].

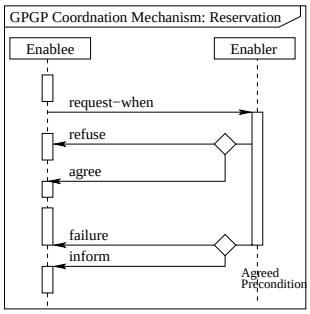

Figure 3: Sequence diagram of GPGP coordination mechanism protocol: Reservation.

4. A SIMULATED EMERGENCY MEDICAL SERVICE (EMS) SYSTEM

Emergency Medical Service, EMS, is defined as a comprehensive, coordinated arrangement of health and safety resources designed to provide expedient care to victims of sudden illness and injury. This paper focuses on *prehospital EMS*, which is generally defined as the response process from the start of an incident to the hospitalization of victims. In EMS systems events happen rapidly and unexpectedly; decision making is highly time-critical based on involving response organizations, institutions, and even geographical sites; doctors, nurses, and paramedics are constantly on the move. Such high-velocity environments pose special computing and communication needs. Information flow among the related entities needs to be highly effective; however, communication channels could be radically impaired based on certain environmental changes, such as traffic jams and power failures. Thus, effective dynamic coordination is key to EMS systems.

A typical EMS process is briefly explained as shown in Figure 4: when an emergency happens, an incident call is reported to the EMS system dispatcher; based on the nature of the incident, e.g. the type of the incident, whether there is any victim at the incident, the dispatcher contacts an appropriate response agency or multiple agencies and sends appropriate vehicles to the incident location; the response vehicle finds its way to the incident location; then proper management of the incident is provided until the incident is considered finished; if any victim needs further medical care, an ambulance will be called to the incident location and the victim will be transported to a proper medical facility by the ambulance; if the incident still requires additional care, other appropriate response vehicles will be dispatched to the incident location as well; after the incident is finished, all participating response vehicles find their ways back to their base stations or move on in response to other incident calls. Further description about the simulated EMS system can be found in [3, 2]. In this paper, we will concentrate on explaining the task structures of response agents within EMS and the application of our extended GPGP mechanisms in this environment.

The first interdependency is named *EMS coordination point one*. The coordination points are also shown as dashed boxes in Figure 4. When an incident happens, the dispatcher needs to deliver the incident message to certain response agents; response agents will send a message back to the dispatcher indicating whether they are available to manage this incident; based on response agents' replies, the dispatcher decides which agent should be selected in response to this incident. That is, the dispatcher can not decide which agent to dispatch for an incident until reply messages arrive; furthermore, the task of a response agent can not start until the notification of the the selection of certain agents. Clearly the dependency is between the message-sending task of the dispatcher and the task of a response agent.

The second interdependency is named *EMS coordination point two* and the task structures of the involving agents are shown in Figure 5[2]. If a police agent has been selected to respond to a police incident, after it arrives at the incident location, it finds that the incident develops and is not simply a police call any more—a victim of the incident needs immediate medical care and potentially needs to be transported to a hospital for treatment; thus the police agent sends a message to a local ambulance agent about the victim; the police agent needs to wait for the arrival of an ambulance; in the meantime the police agent can not carry out other tasks, e.g. to respond to another incident, until the arrival of the ambulance at the incident location; it is easy to figure out that the dependency is between the police agent's tasks and the arrival of the ambulance for immediate medical treatment for victims.

The third interdependency is named *EMS coordination point three*: When an ambulance arrives at the incident location, it finds out that a victim needs to be transported to a local hospital for further treatment; however, open beds

[2]Due to the paper length constraint, the figures for the other two interdependencies are not shown. See [2] for details.

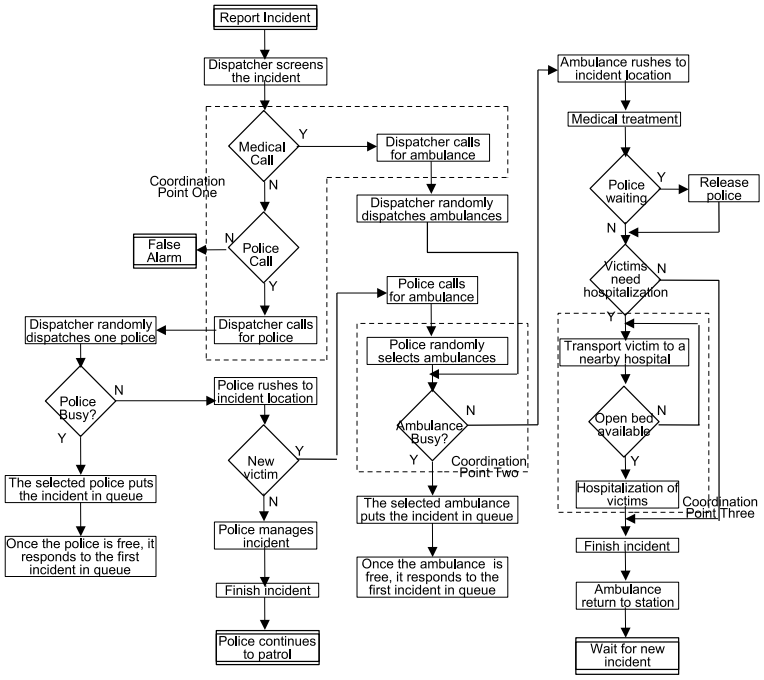

Figure 4: Diagram of EMS operation

may not be always available at hospitals; the hospitalization of a victim for further medical treatment depends on the availability of open beds in the hospital at the time of the victim's arrival; if an ambulance containing the victim arrives at the hospital and there are no any open beds available, the victim has to be transported to another hospital, which is a major delay for further treatment for the victim. In this case the dependency is between further medical treatment for victims by hospitals and the acknowledgment of available open beds from a hospital.

5. EXPERIMENTAL RESULTS

The EMS experimental environment is simplified to include two classes of agents: three police cars and two ambulances move around in local area map in response to incident calls and travel from their base stations (for ambulances) or their current patrol locations (for police agents) to incident locations. There are many other dependencies in an EMS environment. However, we will apply our extended set of GPGP coordination mechanisms to these three coordination points.

Different types of agents have different tasks; tasks for the same type of agents share the same structures. Initial task structures for ambulance, police car, and hospital are shown in Figure 5. These tree-style structures represent tasks for response agents. Top-level root nodes indicate agents' goal tasks. Leaf nodes represent the actions each agent has to execute. The cloud-shape nodes are communication interfaces for an agent to receive and send messages to remote agents. For coordination processes, these communication interfaces could represent non-local tasks as well. The task structures depicts the decomposition of corresponding root tasks into low level tasks until executable actions. The symbols, Input and OK, are the input and output of a task or an action. The links connecting the input and output represent information flow. The meanings of various types of nodes, the task structural information, and task decomposition are further discussed in [6]. Figure 5 only shows main tasks and actions of response agents as a simplified version. The

Table 1: Application of extended GPGP mechanisms to the three coordination points in emergency medical service (EMS) framework.

Index	Mechanisms	P1	P2	P3
1	Avoidance		Y	
2	Sacrifice Avoidance		Y	
3	Reservation		Y	Y
4	Predecessor EST Commitment		Y	
5	Predecessor Deadline Commitment		Y	
6	Predecessor Notice at Start		Y	
7	Predecessor Sending Result		Y	
8	Successor Deadline Commitment	Y	Y	Y
9	Successor EST Commitment	Y	Y	Y
10	Demotion		Y	
11	Promotion			
12	Third Party Execution	Y	Y	Y
13	Third Party Coordinator	Y		
14	Polling for Result	Y	Y	Y
15	Polling for Schedule	Y	Y	Y
16	Constant Headway / Timetabling			Y
17	Bidding	Y	Y	Y

seventeen GPGP coordination mechanisms can be suitably applied to the aforementioned corresponding coordination points as shown in table 5. P1 means Coordination Point One. A "Y" cell in the table shows that the mechanism (to the left of the row) can be applied to the correspond-

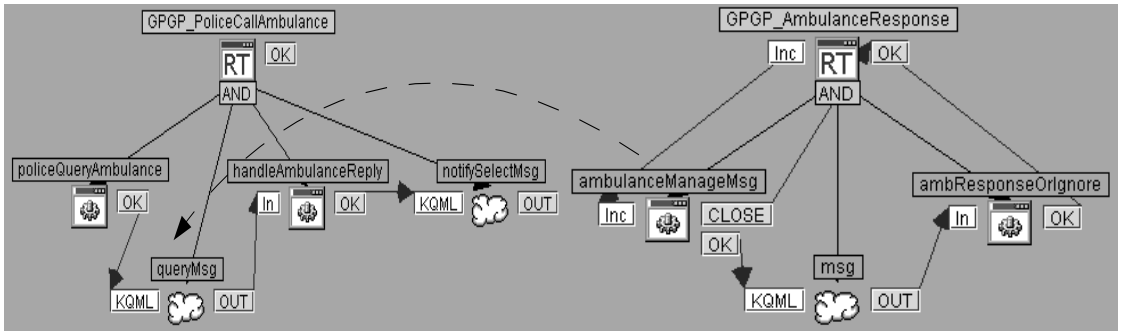

Figure 5: EMS coordination point two: between police and ambulance.

ing coordination point (indicated by the coordination point number to the above of the column); a blank cell indicates the mechanism is not applicable to the corresponding coordination points. For example, the first mechanism, *avoidance*, can be suitably applied to coordination point two, but not to coordination point one and three, simply because there is no alternative local task available; the second mechanism, *reservation*, can not be suitably applied to coordination point one, but can be applied to point two and three; the seventeenth mechanism, *bidding*, can be applied to all three coordination points safely; while *promotion* can not be applied to any of the coordination points.

We define an event, or an accident, that causes a 911 emergency call as an *incident*. There are various types of incidents requiring different response organizations. Coordination during response processes may involve all types of participating agents. An incident is represented as *incident(time, type, severity, location, duration, number, victims[number].(life, life_function))*.

Time means the start time of an EMS process. There are three types of incidents in the model: *medical calls*, *police calls*, and *mixed calls*. A medical call requires ambulances to provide incident victims with primitive treatment and potential transportation to a nearby hospital if needed. A police call requires police agents; it is modeled as a traffic accident or a criminal accident and requires the police to manage the situations, e.g. clearing road or catching criminals. A mixed call starts as a police call and requires police agents at the beginning; upon the arrival of a police agent, the police agent finds out that there are new victims associated with the police call; thus, the police agent calls for ambulances to take care of the victims, as if it was an a medical call. *Severity* indicates whether there are any injured victims associated with the incident and the level of the injury: $(severity = 0)$ means low severity injury and an ambulance is needed for primitive treatment, e.g., CPR or other first aid assistance; $(severity = 1)$ means high severity and an ambulance is not only needed to provide the above basic treatment, but also to transport the victims to a nearby hospital for further medical treatment. An *incident location* means the spot where the incident happens; it is represented with coordinate $Incident.Location(x, y)$; *Duration* indicates how long an incident lasts, e.g how long a response agent is kept on the incident location; in other words, duration indicates the working time for a response agent to manage the incident; an actual duration in experiments follows exponential distribution and will be described later. *Number* represents the number of the victims of an incident; candidate values are (0, 1, 2); if the number is greater than a single ambulance's capacity, multiple ambulances may be dispatched. Notably, the capacity of an ambulance is one, i.e. each ambulance is able to transport one victim at a time. The compound variable, *victim*, is one of the major resources in the EMS framework. *Victim[number].life* represents the victim's initial life value immediately following the incident and this value drops according to the victim's *life function*. A life function indicates how a victim's life value drops as time goes on after an incident. If *life* of a victim drops to zero before entering an appropriate medical facility, it indicates the death of this victim. There are two kinds of life functions defined: sharp linear reduction and flat linear reduction according to time change:(1) $f_1 = (\text{Initial Value} - a_{sharp} * t)$ and (2) $f_2 = (\text{Initial Value} - a_{flat} * t)$, where a_{sharp} and a_{flat} are coefficients indicating sharp and less sharp life value reductions. Based on different values of an incident, different response agents will participate and different operations will be carried out.

In EMS systems, the occurrence of incidents follows a Poisson distribution; and the distribution of time intervals between successive incidents follows an exponential distribution. The time interval, t, from the occurrence of a current incident until the occurrence of a next incident is calculated as the following equation shows.

$$t = \frac{\log(1 - rand())}{-\lambda},$$

where rand() is a uniformly generated random number within the range of $[0, 1)$, $\lambda = \frac{1}{\text{mean}}$, mean is the average value of a series of expected time intervals for next incidents.

There are three main evaluation factors for EMS at a system level: *response time* for quality control, *survival rate* for effectiveness control, and *coordination cost* for efficiency control. In our EMS model, the start of a response time is *when an emergency call is received*; and the end time is *when any appropriate response agent first arrives at the incident location*. Survival rate evaluates the effectiveness of EMS and reflects the overall system performance; survival rate is defined as the rate of victims surviving from an incident reported by a particular EMS system. Coordination cost is another important factor, which reflects system resource consumption by coordination processes, e.g. extra communication for coordination purpose, extra computation for coordination mechanism execution.

The experiments was carried out in a local area network,

agents may reside in different Sun workstations. Each round of experiment consists of one hundred incidents, which have been generated randomly following an exponential distribution at locations in a local area map.

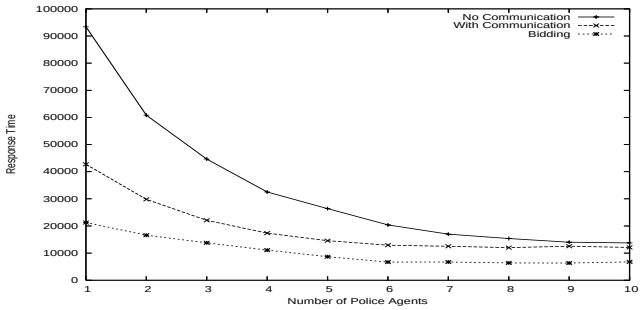

Figure 6: Response time based on changing number of agents (mean=20s).

The first experiment is to estimate the response time of three selected mechanisms, No Communication, With Communication, and Bidding, based on changing number of agents. Different numbers (from one to ten) of agents have been used in each round of experiment. Each response time is an average value of the one hundred response times according to the one hundred incidents for each run. The experimental result is shown in Figure 6. The result indicates that when the number of response agents is limited, e.g., below four, *Bidding* mechanism achieves the shortest response time; coordination without communication has the worse response time; the availability of communication improves response time significantly. As the number of agents increases, the agents are distributed across a local area and the response times approximates with each other when the number is above eight. This experiment leads us to the most suitable number of limited resources (agents) for a certain EMS environment.

Now let us introduce the experiments on the performance of various coordination mechanisms applied to different coordination points in our EMS system. The selection of each kind of response agents is: three police agents, two ambulance agents, two hospitals, one dispatcher. These facts are randomly generated and follow an exponential distribution: next incident arrival time (average arrival time is ten seconds), service time for response agents (average time for a police or an ambulance agent to work at an incident location is two seconds), hospital service schedule time (the average schedule cycle, the arrival of a next victim not from this EMS framework, occupying the open beds, is ten seconds; the service time for a victim is four seconds.). Initial life value of a victim is twenty seconds. The coefficient of life function with sharp reduction is simply one; the coefficient of life function for less sharp reduction (life value decrease slowly.) is 0.5. The EMS evaluation factors of GPGP coordination mechanisms applied to coordination points based on Table 5 will be presented in detail.

As the tasks from Figure 5, Figure 7 shows the sorted average EMS response time of the selected thirteen mechanisms. The *base case* is the situation where there is no GPGP coordination mechanism deployed. If there is an alternative local task, the coordination task can be removed;

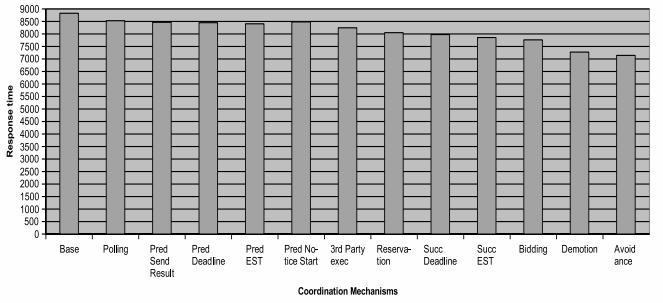

Figure 7: Average response time of selected GPGP Coordination Mechanisms on EMS Coordination Point Two.

thus the *Avoidance* mechanism has the shortest response time. When there is no alternative tasks, *Demotion* mechanism is the best, while *Polling* takes the longest time. Other mechanisms have difference performance in between.

In response to the performance of the selected coordination mechanisms, the shortest response time results in the highest victim survival rate, because a response agent may reach the incident location and provide treatment to the victim immediately. The survival rates of different coordination mechanisms maintain the sequence as in Figure 7 and are shown in Figure 8. The survival rate is affected by the response time: *Avoidance* produces the highest survival rate; *Demotion* is the best if coordination cannot be avoided; *Polling* has the lowest survival rate; other mechanisms fall in between.

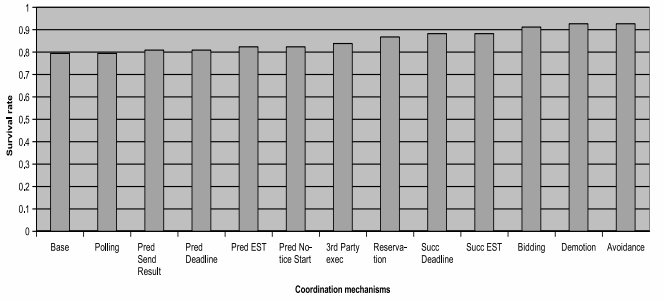

Figure 8: Average survival rate of selected GPGP Coordination Mechanisms on EMS Coordination Point Two.

Figure 9 shows EMS coordination cost versus selected coordination mechanisms applied to coordination point two. Given that *Polling* is carrying out persistent queries, it does results in the highest coordination cost. *Avoidance* has little coordination cost (the only cost is the detection of interdependency and its local alternative task). Other coordination costs by the rest of the mechanisms are as shown.

Due to the distinctive structural features of the three coordination points described earlier, different coordination mechanisms generate different results for the three EMS evaluation factors. We have shown coordination point two in detail as an example. The figures of the experimental results of the mechanisms applied to coordination point one

and three are omitted here. But, the most important factor, EMS response time, of the selected mechanisms for coordination point one and three will be listed as follows. Seven coordination mechanisms have been applied to coordination point one, the order from the longest response time to the shortest is: Base Case, Polling, Successor Deadline Commitment, Success EST Commitment, Third-Party Execution, Bidding, and Third-Party Coordinator. Similarly, eight coordination mechanisms have been applied to point three, and the order from the longest response time to the shortest is: Base Case, Polling, Third-Party Execution, Success EST Commitment, Successor Deadline Commitment, Constant Headway, Bidding, and Reservation.

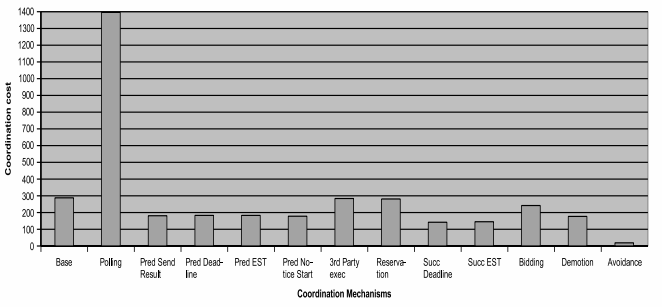

Figure 9: Average coordination cost of selected GPGP Coordination Mechanisms on EMS Coordination Point Two.

The above experiments have shown that different coordination mechanisms may perform best at different coordination points. It is important to see how the overall system performances be improved by a combination of the best mechanisms. In this experiment, we select a *combination* environment: *bidding* mechanism applied to coordination point one, *demotion* applied to point two, *reservation* applied to point three. The base case is that there is no any coordination mechanism applied to any coordination point. The next table shows the result and concludes that a careful

Table 2: System performances caused by a combination of the best mechanisms applied to the corresponding three coordination points .

Environments	ResponseTime	SurvivalRate	TotalCost
Base Case	9523	80%	1010 msec
Combination	3598	100%	624 msec
Improvement	37.8%	125%	61.8%

selection of our extended set of GPGP coordination mechanisms at different coordination points significantly improves the overall system performances.

6. CONCLUSIONS AND FUTURE WORK

We have briefly introduced a highly expressive representation, EHTNs, to explicitly denote the characteristics of agents' task structures. Given EHTNs, interdependencies can be detected and analyzed. An extended set of domain-independent GPGP coordination mechanisms has been developed to address the coordination problem. We also introduced a simulated EMS system for demonstrating the performance of various mechanisms applied to selected interdependencies. We have experimentally demonstrated that the application of our mechanisms improves system performance and different mechanisms may outperform others in different environments (or coordination points within an environment). We have demonstrated that the development of coordination mechanisms based on the analysis and alteration of agents' task structures is a promising approach.

The final experiment about overall system performance is a greedy approach, meaning that selecting the best mechanism for each coordination point without considering potential relationships among these coordination points, i.e. it is not determined whether the result shown in table 5 is the best possible improvement, since other combinations of mechanisms may be better. The experiments about other potential combinations of mechanisms, and of course learning appropriate sets of mechanisms, are part of future work.

7. REFERENCES

[1] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11(1):1–94, 1999.

[2] W. Chen. *Designing an Extended Set of Coordination Mechanisms for Multi-Agent Systems*. PhD thesis, Computer and Information Sciences, University of Delaware, 2005.

[3] W. Chen and K. Decker. The analysis of coordination in an information system application - emergency medical services. In *AOIS05*, LNCS-3508, P36–51. Springer, 2005.

[4] K. Decker and J. Li. Coordinating mutually exclusive resources using gpgp. *JAAMAS*, 3(2):133–157, 2000.

[5] K. Erol, D. Nau, and J. Hendler. A critical look at critics in htn planning. In *Proceedings of the 4th IJCAI*, Montreal, Aug. 1995.

[6] J. Graham and K. Decker. DECAF a flexible multi-agent system architecture. *JAAMAS*, 7(1), 2003.

[7] B. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[8] G.Weiss. *Multiagent Systems, A Modern Approach to DAI*, pages 151–157. MIT Press, 1999.

[9] T. Malone and K. Crowston. The interdisciplinary study of coordination. In *ACM Computing Surveys*, pages 87–119, Mar. 1994.

[10] L. Ngo, P. Haddawy, and H. Nguyen. A modular structured approach to conditional decision-theoretic planning. In *AIPS98*, P111–118, 1998.

[11] R. Parr. Flexible decomposition algorithms for weakly coupled markov decision problems. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, Morgan Kaufmann, 1998.

[12] T. W. V. Lesser, K. Decker. Evolution of the gpgp/t?ms domain-independent coordination framework. *JAAMAS*, 1(2):87–143, 2004.

[13] Q. Yang. Formalizing planning knowledge for hierarchical planning. *Computational Intelligence*, 6(1):12–24, Feb. 1990.