Workshop on Coordinating Agents’ Plans and Schedules
CAPS-07

May 15th 2007

Michael Brenner
Brad Clement
Mathijs de Weerdt
Program committee

Organizers

- Michael Brenner, Albert-Ludwigs-Universität Freiburg
- Brad Clement, Jet Propulsion Laboratory, Pasadena
- Mathijs de Weerdt, Delft University of Technology

Program committee

- Anthony Barrett, Jet Propulsion Laboratory, Pasadena
- Keith Decker, University of Delaware
- Ed Durfee, University of Michigan
- Boi Faltings, Swiss Federal Institute of Technology
- Piotr Gmytrasiewicz, University of Illinois at Chicago
- Nick Hawes, University of Birmingham
- Sven Koenig, University of Southern California
- Roman van der Krogt, University College Cork
- Victor Lesser, University of Massachusetts
- Karen Myers, SRI International
- Jeff Rosenschein, Hebrew University of Jerusalem
- Reid Simmons, Carnegie Mellon University
- Steve Smith, Carnegie Mellon University
- Tom Wagner, DARPA
- Cees Witteveen, Delft University of Technology
- Shlomo Zilberstein, University of Massachusetts
Contents

Víctor Muñoz, Javier Murillo, Dídac Busquets and Beatriz López
   *Improving Water Quality by Coordinating Industries Schedules and Treatment Plants*  
   1

Keith Purrington and Edmund Durfee
   *Joint Goal-Setting for Self-Interested Agents Using CP-nets*  
   9

Roman van der Krogt
   *Privacy Loss in Multiagent Planning: A Classical Definition with Illustration*  
   17

James Atlas and Keith Decker
   *Task Scheduling using Constraint Optimization with Uncertainty*  
   25

Wei Chen, Renato Levy and Keith Decker
   *An Integrated Multi-Agent Coordination Including Planning, Scheduling, and Task Execution*  
   29

J. Renze Steenhuisen and Cees Witteveen
   *Coordinating Planning Agents for Moderately and Tightly-Coupled Tasks*  
   37
Improving Water Quality by Coordinating Industries Schedules and Treatment Plants

Víctor Muñoz, Javier Murillo, Dídac Busquets and Beatriz López
Institut d’Informàtica i Aplicacions
Universitat de Girona
Campus Montilivi
17071 Girona, Spain
{vmunozs,jmurillo,busquets,blopez}@iia.udg.es

ABSTRACT
Having a proper waste water treatment system is crucial for making a good use of water resources. Current regulations enforce some restrictions to the industries producing waste, according to the capacities of waste water treatment plants. However, these are usually not sufficient to ensure that these capacities are not exceeded. In this paper we present a coordination system that provides a more integrated view of the problem, taking into account all the elements involved in the treatment system. The goal of the system is to coordinate the individual industries' discharge schedules over time to help the treatment plant in the cleaning process of the water. The system we propose is based on an auction mechanism, in which the industries can bid for the right to perform a discharge. We have extended it with a priority mechanism in order to provide a fair solution to the problem. A prototype of the system has been implemented and tested on simulation, obtaining successful results.

Categories and Subject Descriptors

General Terms
Algorithms

Keywords
Schedule Coordination, Auctions, Multi-agent Systems, River-basin Management

1. INTRODUCTION
Water is a vital natural resource, not only for urban and industrial consumption, but also as the main element to maintain any natural environment. Thus, the need of having a good treatment system is basic in order to account for the high demand it suffers. Moreover, the increase in population growth and industrial activity results in the harmful effect of having more contaminated waters, therefore making the treatment task harder.

The most important element in the treatment system is the Waste Water Treatment Plant (WWTP). Its job is to remove contaminants from sewage and produce an (up to a certain degree) clean water that can be put back into the river. In order to ensure that the treatment process is correctly performed two conditions must hold:

- Keep the incoming water flow below the WWTP hydraulic capacity (that is, the amount of water the plant can absorb at any instant in time); otherwise, the overflowed water goes directly back to the river without receiving any treatment, increasing its contamination level.

- Keep the contamination level of the incoming water below the WWTP treatment capacity. The contamination level is defined by a set of quality variables (oxygen demand, nitrogen level, etc.). If the level of any of these variables is above the WWTP capacity, the water cannot be fully treated, and it increases the contamination of the river. Moreover, if the levels were too high, the microorganisms used to treat the water may be damaged and the whole process could be stopped until these were regenerated. During this time, the plant could not accept any incoming water and it would be redirected to the river without any treatment.

These capacities or thresholds, are usually referred to as the WWTP's design parameters and depend on the amount of water to be treated in relationship with the surrounding industries and cities.

The water entering the WWTP comes from three different sources: domestic use, rainfall and industries. Current regulations and legislations are in place so as to minimize the contaminating effects of industrial waste discharges. However these are not sufficient to guarantee the proper treatment of the water. The problem is that, although these regulations enforce industries to respect the WWTP capacity thresholds, they do not take into account that simultaneous discharges by different industries may exceed these thresholds. In such a case, no industry would be breaking the
rules, but the effect would be to have overflow or overcon-
taminated water going to the WWTP.

Thus, in order to ensure that the thresholds are not ex-
ceeded, some coordination between the industries and the
WWTP is needed. Through this coordination, the different
discharges would be temporally distributed so that the
WWTP is capable of processing all the incoming water. This
would be beneficial at an environmental and health level,
and is in the direction of having a more integrated manage-
ment of the river-basin and all the involved systems. More-
ever, it could also go together with economic incentives to
those industries collaborating the most (such as discounts
in the discharge fees).

In this paper we propose a coordination system to mediate
between the different industries willing to discharge waste
at the same period of time. More specifically, we have de-
veloped an auction mechanism to coordinate the individual
industries’ discharge schedules. The paper is organized
as follows. In Section 2 we present the coordination sys-
tem, describing in detail each of the steps involved. The
implementation of a first prototype is described in Section
3. In Section 4 we discuss the experimental results obtained
through simulation. Some related work is presented in Sec-
tion 5, and finally Section 6 concludes the paper.

2. COORDINATION SYSTEM

A typical water treatment system is depicted in Figure 1.
The industries discharge their wastes to a sewage system,
which directs the water to the WWTP. The plant, once the
water has been treated, puts it back to the river. The main
goal of our system is to ensure that the water flow enter-
ing the WWTP and its contamination levels are below some
given thresholds, so that it can be correctly treated. As men-
tioned previously, we propose to achieve this goal by coor-
dinating the discharges performed by the industries. In this
section we first describe the assumptions made about the
industries performing the discharges, and then we present
how the coordination system works.

2.1 Industry Model

Although there are two other sources of waste water (rainfall
and domestic use). In this first approach to the problem we
focus our attention to industries.

```
while not all discharges authorized do
    receive schedules from industries
    loop through discharges
        if no conflict then
            authorize current discharges
        else
            solve conflict
            inform industries about resolution
        end if
    end loop
end while
```

We assume that the industries have some kind of working plan
that allows them to foresee what discharges will be
necessary in the near future, according to their production
strategy. This knowledge would permit the industries to
inform the WWTP about the characteristics of their dis-
charges (starting time, duration, flow and contaminants lev-
els), so that coordination can be achieved.

We also assume that each industry has a tank where it can
store its waste in case a discharge is not authorized. With-
out this tank, an industry would be forced to perform a dis-
charge in the case it was not authorized, making the whole
coordination process useless. Obviously, if the industry is
denied to discharge and its tank is full, it will be forced to
realize the discharge anyway. This could seriously affect
the process in the WWTP, and therefore should be avoided at all
costs. The coordination mechanism that we propose tackles
this problem by distributing the authorizations among the
industries, trying to avoid that any of them has to perform
unauthorized discharges.

An industry may also reschedule its discharges when it is
denied the right to perform them at a given time. How-
ever, this would depend on the kind of production process
taking place in that industry, which may or may not allow
this kind of change in discharges. In our industry model we
assume that the industries can only delay their discharges.
Regardless of the rescheduling behavior of the industry, we
also assume that at most, an industry could perform two dis-
charges at the same time: one coming from the production
process, and another one coming from the retention tank.

In addition, the industries are equipped with sensors in their
outgoing pipes, which are used to control the amount of
waste being discharged. This information can then be used
for two purposes: to compute the fee each industry has to
pay (according to the volume and contamination levels of its
discharges) and to control whether the industries perform
any unauthorized discharge.

2.2 System Overview

As we can see, there are two main elements in the treatment
system: the WWTP and the industries. We have designed
our system as a Multiagent System to reflect the physical
separation between these elements and also to support pri-
vacy in the decision making process of each of the involved
agents. Thus, there is an agent for the WWTP and one agent for each industry.

The WWTP agent, upon reception of all the industries’ discharge schedules for one day (or any different predefined period of time), checks for conflicts between them. That is, it checks whether two or more simultaneous discharges would exceed the WWTP capacity thresholds. When a conflict is detected, the system has to select which industries are allowed to discharge and which should be delayed. The resolution is done in a sequential way, treating one conflict at a time in chronological order. Once the current conflict is solved, the involved industry agents are informed about the resolution, and then each agent updates its discharge schedule (depending on whether it has been authorized or denied to discharge). Those agents modifying their schedules inform the WWTP agent, and it can then check for the next conflict. This iterative process is performed until all discharges are authorized by the WWTP agent. A pseudocode of the coordination algorithm is depicted in Figure 2. The diagram of the communication between industry agents and the WWTP agent is shown in Figure 3.

It is important to note that discharge scheduling mirrors the planned production activity of the industries that come out after a very complex process (issues like just-in-time production, manufacturing parameter adaptation, etc.). So a centralized approach that changes the order of the discharges could imply some changes to the scheduled production activity. Conversely, solving conflicts one at a time based on the fact that each industry has a tank to deviate the waste water, and delaying the discharges, does not cause interferences to the production activity.

The coordination process is done offline, that is, the whole process is done before the discharges are actually performed. It could be done one or several days in advance, depending on the planning capabilities of the industries. The result of the coordination process is a set of new schedules for each industry, which will have no conflicts. In the next subsections we describe in detail each of the steps the WWTP and industry agents perform during the coordination process.

2.3 Conflict Detection

As mentioned previously, each industry agent informs the WWTP agent about its discharge schedule. Thus, this agent is provided with the following information:

\[\mathbf{D} = \{(s_1^k, q_1^k, r_1^k, d_1^k), (s_2^k, q_2^k, r_2^k, d_2^k), \ldots, (s_{N_1}^k, q_{N_1}^k, r_{N_1}^k, d_{N_1}^k),
\]

\[\ldots
\]

\[\ldots, (s_{N_1}^{k}, q_{N_1}^{k}, r_{N_1}^{k}, d_{N_1}^{k})\}\]

where:

- \(s_i^k\) is the start time of the \(k^{th}\) discharge of industry \(i\),
- \(q_i^k\) is the flow of the discharge,
- \(r_i^k\) is a vector containing the contaminants levels of the discharge per volume unit,
- \(d_i^k\) is its duration,
- \(n_i\) is the number of discharges of industry \(i\),
- and \(N_1\) is the total number of industries

A conflict arises when the set of discharges in a given instant exceeds the WWTP hydraulic capacity or the contamination levels. We consider that a conflict begins when an industry starts a discharge that causes any of these thresholds to be exceeded. The conflict ends when an industry finishes a discharge and the WWTP levels go back to be within the allowed limits. The industry that causes the beginning of the conflict can be different to the one that causes the end. All the industries that are discharging during the conflict are the ones involved in the conflict.

Figure 4 illustrates an example of a conflict. There are four industries discharging waste with different flows. For example, in timestep 0, the second industry begins to discharge with a flow of 100 m$^3$/d, finishing at timestep 4. Supposing that the maximum flow capacity of the WWTP is 300 m$^3$/d, a conflict arises in timestep 2, when industry 4 starts its discharge, because the sum of flows being discharged by the industries (370) exceeds this limit. The conflict ends at timestep 4, when industry 2 finishes its discharge and the sum of the remaining flows (270) falls below the capacity threshold. In this case, the involved industries in the conflict are 2, 3 and 4.
2.4 Conflict Resolution

Once the discharges involved in a conflict are detected, the corresponding industry agents are informed about it, and the coordination process begins. The WWTP agent has to select a subset of the conflicting discharges that will be authorized, while some others will be asked to be delayed. To perform this selection, we have chosen to use an auction mechanism, a well-known mechanism to distribute goods among competing agents when information privacy is a concern [1]. The winners of the auction are allowed to discharge, while the losers have to wait for another opportunity.

The WWTP agent calls for an auction in which the conflicting industry agents can place their bids for the right to perform their discharges. Each bid sent by the industry agents has the following form:

\[ b_i = (s_i, q_i, \tau_i, d_i, v_i) \]

The first component of this tuple contains the characteristics of the discharge (as explained in the previous section). This component is actually not sent with the bid, since the WWTP agent already has the information about discharges. The second component, \( v_i \in \mathbb{R}^+ \), is the value the corresponding industry agent gives to the discharge. A description of how this value is computed is given in the next section.

The WWTP agent will receive as many bids as conflicting discharges are. Note that an industry agent could have at most two discharges involved in a conflict (one coming from the industry’s production process and another one from the retention tank). Thus, the number of participating agents in the auction can be less than the number of bids. However, each bid is considered to be independent, so there is no restriction on the number of bids an agent can be awarded (it could either be both discharges, one of them or none).

The goal of the WWTP agent is then to select those discharges that maximize a given objective function, subject to the capacity restrictions (hydraulic and contaminants). Formally, to find the winners of the auction, the clearing algorithm must solve the following optimization problem:

\[
\max_{\omega} \sum_{i=1}^{ND} x_i \cdot g(\omega) \\
\text{s.t.} \quad \sum_{i=1}^{ND} x_i \cdot q_i \leq Q \\
\quad \mathcal{K} \leq \mathcal{C}
\]

where:

- \( ND \) is the number of conflicting discharges,
- \( x_i \in \{0, 1\} \) represents whether discharge \( i \) is denied (0) or authorized (1),
- \( g(\omega) \) is the contribution of discharge \( i \) to the objective function to be maximized. With \( \omega \) we refer to all information associated to bid \( i \) (start time, duration, flow...). This function can vary depending on the goal to achieve; possible candidates are:

\[
g(\omega) = \begin{cases} 
q_i & \text{maximize discharges' values} \\
q_i \cdot v_i & \text{maximize incoming flow} \\
q_i \cdot v_i & \text{tradeoff between previous criteria}
\end{cases}
\]

- \( Q \) is the maximum hydraulic capacity of the WWTP
- \( \mathcal{K} \) is a vector containing the contaminants levels given the authorized discharges:

\[
\mathcal{K} = \frac{\sum_{i=1}^{ND} x_i \cdot q_i \cdot \tau_i}{\sum_{i=1}^{ND} x_i \cdot q_i}
\]

- and \( \mathcal{C} \) is a vector containing the maximum contaminants levels accepted by the WWTP

This formulation is similar to a multi-unit combinatorial auction [9], in which the auctioner offers multiple (but limited) units of different goods and bidders submit bids for a certain number of units of each good. In our case, the goods would be the flow and the contamination levels entering the WWTP, and the available units would be defined by \( Q \) and \( \mathcal{C} \). Moreover, since in our case each bidder is allowed to place only one bid per discharge, it could also be seen as a multi-dimensional knapsack problem [10], the optimization problem of selecting a subset of valued objects that can fit into a bag with restrictions on its dimensions, with the goal of maximizing the stored value.

To solve the winner determination problem we have used a Linear Programming approach, since the size of our problem is not too large. However, if the size were to be intractable by linear programming, other algorithms could be used, such as Genetic Algorithms or any of the existing efficient combinatorial auctions algorithm, such as those presented in [5]. Note that the auction is repeated each time a conflict is detected, so we are dealing with a recurrent auction.

2.5 Bidding Policies

One of the key points in auctions (besides the winner determination algorithm) is the bidding policy of the bidders. This policy determines how an agent generates its bids, i.e., how it chooses the goods to bid for and the value (price) associated to each good (or set of goods).

In our case, the agents do not have to choose among different goods, since these are already defined by the discharge characteristics. However, the agent still has to compute the value \( v_i \) for each bid. We have considered two different alternatives for such computation. In the first one, the value represents the urgency the industry has for performing the discharge. This urgency would depend on the production process of the industry and the state of its resources (e.g., is the retention tank available? How much can I store there?). Thus, the more urgent a discharge is, the higher its associated value should be.

The other alternative is to give a more economical view to the value. In this case, it could represent the price the industry is willing to pay for performing a discharge at a given
time. This alternative has somehow the urgency degree embedded, since probably an industry needing to perform a discharge immediately is inclined to paying much more than when the urgency is low.

2.6 Adding Priorities

The auction mechanism presented previously has a major drawback: it can be unfair. The reason is that the auction is solving a one-shot problem, and does not take into account the history (who won and who lost) of previous auctions. However, the coordination process is repeated over time to solve all the conflicts that may arise. Thus, if an industry wanted never to be denied a discharge, it could always bid very high, pretending that its discharges are very urgent, guaranteeing its success in the auction. This behavior could prevent other industries from getting any discharge authorization, which could highly affect its production processes, or even force them to perform a discharge without having been authorized (an act that would be detected by the industry’s sensors and could be penalized by the regulatory authorities).

Although there are several auction protocols (such as the Vickrey-Clarke-Groves auction) that incentivize agents to bid truthfully (which would remove the problem of irrationally high bids), we are more interested in having a fair distribution of the authorizations. Even if all agents were bidding their true values, the problem of an uneven distribution would still be possible. Thus, a new mechanism to manage the recurrent aspect of the auctions is needed.

In order to find a fair solution for all the conflicts found over time, we have added a priority mechanism that takes into account the history of each agent having been authorized or denied to discharge over time. The mechanism assigns a priority to each agent, $W = \{w_1, w_2, \ldots, w_N\}$, that is used in the auction clearing algorithm to find the solution. High priority values indicate that the agent should be authorized to discharge, while low priority values indicate that it would not be unfair to deny a discharge to that agent. These values are updated after each auction, according to its outcome. If an agent wins an auction, its priority is increased, while if the agent loses, it is decreased.

These priorities could be used in very different ways, such as defining new constraints that should be satisfied by the solution, or directly designating some or all of the winning agents, among others. We have chosen to use these priorities as a modifier of the bids sent by the agents. Formally, given a bid $b_i$ submitted by agent $k$, a new bid is computed as:

$$b'_i = f(b_i, w_k)$$

This function $f$ is a parameter of the system. Note that the bid and agent indexes do not necessarily match, since as mentioned previously, an agent may have sent more than one bid.

The use of the priorities introduces an egalitarian view of the auction, in contrast with an utilitarian view, which is used more often but does not take into account how fair is the system. Thus, even if an agent bid very high, given it had a low priority, its bid should be decreased somehow. Similarly, an agent bidding low could still be the winner of the auction if it had a high priority.

3. IMPLEMENTATION

To evaluate the coordination mechanism we have implemented a prototype of the system. For programming it we have chosen Reaspi [14], a free open source software framework for creating agent based simulations using the Java language. The simulation reproduces the process and the communication between the WWTP and the industries performing waste discharges. We have created an agent to represent the WWTP and another one for each of the industries.

As a first evaluation of the system, we have only considered the hydraulic capacity and have supposed that the industries always obey the WWTP decisions, as long as they have enough tank capacity. In the near future we will consider different contaminant components and introduce different industry behaviors to have more realistic scenarios.

To calculate the bid, the industry agent takes into account the urgency for performing the discharge, based on the retention tank occupation of the industry:

$$w_i = \frac{\text{tank occupation}_i}{\text{total tank capacity}_i}$$

Recall that we assume that each industry has a retention tank used to retain any discharge that is not authorized at a given moment. In case an industry agent has to reschedule its discharges, its behavior is the following: it first tries to store the rejected discharge into the tank. The discharge of the tank is then scheduled as the first activity of the agent after the current conflict finishes. The remaining discharges are shifted so that they do not overlap.

To calculate the priority of agent $k$, $w_k$, we take into account the number of lost and won auctions:

$$w_k = \frac{\text{lost auctions}_k + 1}{\text{total participated auctions}_k + 2}$$

The initial priority of each agent is 0.5. If an agent loses an auction its priority is increased, otherwise it is decreased.

The function chosen to modify the industry’s bid according to its priority changes the value sent by the industry. So, given $v_k$, the bid value by agent $k$ and its priority, $w_k$, the actual value used is $v'_k = v_k \cdot w_k$.

The objective function ($g(a)$) to maximize in the auction clearing is the sum of discharge values. The linear programming toolkit GLPK [8] has been used to solve the winner determination problem. Actually, it is a 0-1 integer programming problem, since we do not permit splitting discharges, so a discharge is either authorized or denied.
<table>
<thead>
<tr>
<th></th>
<th>NO</th>
<th>MFO</th>
<th>VO</th>
<th>MT</th>
<th>TDT</th>
<th>%A</th>
<th>%MWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>without coord</td>
<td>5.7</td>
<td>6515</td>
<td>138860</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>(1.16)</td>
<td>(2093.46)</td>
<td>(64854.43)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>coord w/o prios</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10492.1</td>
<td>3355</td>
<td>74</td>
<td>25.75</td>
</tr>
<tr>
<td></td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(28454.67)</td>
<td>(19348.41)</td>
<td>(6.2)</td>
<td>(18.22)</td>
</tr>
<tr>
<td>coord w/ prios</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7009.3</td>
<td>243.4</td>
<td>78</td>
<td>48.73</td>
</tr>
<tr>
<td></td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(2787.76)</td>
<td>(1763.6)</td>
<td>(5.8)</td>
<td>(18.42)</td>
</tr>
</tbody>
</table>

Table 1: Experimental results (average and standard deviation)

The experiments consisted of ten simulations using a set of real data of ten industries in ten different days. The industries can have the same discharge schedule each day (if they produce the same products every day) or different (if they have changes in the production process).

We have tested the system with three different scenarios. In the first scenario there is no coordination between the industries and the WWTP. The second has coordination, and the third has coordination and uses the priorities.

The results shown in Table 1 are the average and the standard deviation of the ten executions (dashes indicate that the measure is not applicable). The differences between TDT and %A when using priorities and without them are not statistically significant. However, the difference between MT and %MWA is statistically significant. The results show that without schedule coordination (no matter with or without priorities) we eliminate the overflows. Moreover, there were no forced discharges in any of the experiments. Figure 6 illustrates an example of the behavior of the system without any coordination, while Figure 7 shows the behavior of the system with the same example when using the coordination mechanism with priorities. In the first figure we can observe that the WWTP capacity is being exceeded seven times, while in the second the maximum capacity is never exceeded.

However, the final time of the execution (TDT) is increased by about 4 and 5 hours respectively (243.4 and 335.5 minutes as shown in Table 1). This could cause some problems with the scheduling of the following day. Although this delay is considerable, the effect of priorities and its fair distribution can be seen in the reduction of the average and standard deviation of this time. This is also the case with the modification time, for which the increase is lower with priorities than without. Looking at Figure 7 we can observe that sometimes the WWTP flow is underused. If industries were allowed to perform multiple discharges (from the buffer and from the production process) at the same time, the reschedule delays could be shortened. We need to deal with this possibility in future work.

Regarding the rate of authorized discharges (%A), the percentages obtained with or without priorities are similar. However, as shown in Figure 8, the standard deviation in each of the experiments is smaller when using priorities. Actually, the average of this standard deviation is 27% without priorities, and 19% with priorities. This means that the difference between the agents is reduced with the use of priorities, increasing the fairness of the system. Moreover, the results

Figure 5: User interface

Figure 5 shows the application user interface. The graphical representation shows the tank occupation levels of the industries and the occupation degree of the WWTP.

4. EXPERIMENTAL RESULTS

In order to evaluate the system we have considered some quality measures based on different characteristics of the solution. These characteristics are the following:

- **number of overflows (NO):** number of overflows occurred during the execution of the discharge schedules.
- **maximum flow overflown (MFO):** measured in m³/d.
- **volume overflown (VO):** total liters overflown.
- **modifications time (MT):** sum of differences between the initially proposed discharges times and the actual times after coordinating the schedules, measured in minutes.
- **total delay time (TDT):** difference in minutes between the final time of the execution and the final time when no coordination is used.
- **% authorized discharges (%A):** percentage of conflicting discharges that have been authorized.
- **minimum % won auctions (%MWA):** minimum percentage of won auctions among all agents.
Figure 6: Behavior without coordination

Figure 7: Behavior with coordination and priorities

Figure 8: Average and standard deviation of won auctions percentage for each experiment

also show that the minimum percentage of won auctions (%WMA) is significantly increased with priorities (48.73% against 25.75% when priorities are not used, as shown in Table 1). This indicates that an agent has more chances of being authorized to discharge when the priority mechanism is used.

5. RELATED WORK

Coordination in multi-agent systems is a very important issue since it directly affects the overall system performance. The coordination can be performed at execution time or at planning and scheduling time. In our work we have focused on the latter option, coordinating agents schedules. Schedule coordination can be further divided depending on whether the goal is to coordinate existing schedules or to create new schedules for each of the agents. Since our agents (industries) have their individual schedules, we are faced with the former problem of coordinating schedules that have already been generated.

There are many approaches to handle schedule coordination from a divide-and-conquer strategy [2, 3, 15], to solving it as a constraint optimization [4], or using auctions [6, 7], among others. We have followed this option, use a market approach to coordinate the agents’ schedules. The characteristics of our problem makes the auction to be continuously repeated, so we are dealing with a recurrent auction. This kind of auction is recently being used for e-services markets, such as assigning advertising time in public displays [12] or in networking markets [11].

This latter work is closely related to our problem, since it tackles the bidder drop problem. This problem arises when bidders are frustrated with the outcome of the auctions (usually because they are constantly losing) and decide to leave the marketplace. We are also very interested in this problem, since we need to incentivize the agents to participate in the coordination process. In [11] the problem is solved by defining a more flexible winner determination algorithm, which takes into account the bidder’s outcome history in past auctions. The goal of their work is to incentivize the bidders to stay in the market place, so that the prices do not collapse. We also use this history in order to compute the agents’ priorities, but our objective is not economic, but to obtain a fair distribution of the discharge authorizations.

Regarding work on water treatment systems, there has been much research on the internal treatment processes, but very little on coordinating the different systems involved. An example is [13], where a negotiation approach to coordinate different WWTPs treating the same river basin is presented. However, the elements being coordinated in this work are the WWTPs, leaving the industries aside.

6. CONCLUSIONS AND FUTURE WORK

The pressure exerted to water resources is rapidly increasing due to the high demand by industries and domestic consumers. To ensure a proper supply, water must be treated to remove contaminants so that it can be put back to the river. Current regulations prevent industries from discharging waste with high contamination load. However, these are deficient and do not fully attain the goal of maintaining a low contamination level in the water so that it can be successfully treated. Instead of focusing on individual industries, a more integrated view of the problem is needed. Taking into account all the elements involved in the treatment system and their interactions could help improving its performance.
In this paper we have presented a mechanism to coordinate the discharge schedules of a set of industries so that the WWTP is capable of treating all the waste they produce. Through the coordination, the individual schedules are refined whenever a conflicting situation is detected. The new schedules contain a sequence of discharges that are distributed over time and decrease the risk of possible treatment failures by the WWTP. The core of the coordination mechanism is a recurrent auction, in which industries can bid for the right to perform a discharge at a given time.

The auction has been extended with a priority mechanism to introduce fairness in the assignment of authorizations. With this mechanism, the authorizations to perform discharges are evenly distributed among the industries, meaning that their original schedules are modified the least possible. This is a very desirable property of a coordination mechanism, since it incentivizes agents to participate in it. Otherwise, if the mechanism were to drastically modify their schedules, agents would be reluctant to participate. This could lead to an overall failure of the system, since each agent would be acting on its own. This is specially important in environments where agents are self-interested and do not pursue a common goal. In these cases, coordination must offer an add-on so that agents are attracted. In our water treatment domain, this add-on would be discounted fees for those industries complying with the discharge authorizations.

The results obtained through simulation show that the coordination mechanism accomplishes the goal of maintaining the incoming flow below the WWTP hydraulic threshold. The results have also shown that the use of priorities provides a fairer solution of the auctions. However, we need to further study how to reduce the delay produced by rescheduling. We also need to incorporate the contaminants levels restriction, since we have taken into account only the hydraulic capacity of the plant.

The system presented in this paper does not yet capture the complexity of a real water treatment system. As a first step towards getting more realistic scenarios, we plan to introduce disobedience behavior to the industry agents. In the current prototype, disobedience only occurs when an industry has its retention tank full and it is not authorized to perform a discharge, which forces it to perform it anyway. However, an industry may be interested to empty its retention tank when it is not full, even if the discharge is not authorized. We plan to use a trust mechanism so that the WWTP agent may foresee this kind of situation and can react accordingly. Some other open issues include the introduction of a more economic view on the bid value, and study whether the method used to compute the priorities is the most appropriate.

7. ACKNOWLEDGMENTS
We thank the LEQUIA research group (Laboratory of Chemical and Environmental Engineering) for having introduced us to the water treatment problem and for the help given during our research. This research project has been partially funded by the Spanish MEC project TIN2004-03546-C02-02 and DURSI AGAUR SGR 00290 (AEDS).

8. REFERENCES
Joint Goal-Setting for Self-Interested Agents Using CP-nets
Keith Parringon and Edmund H. Durfee
Computer Science and Engineering
University of Michigan, Ann Arbor, MI 48109 USA
purting@umich.edu, durfee@umich.edu

ABSTRACT
In multi-agent planning, there is often a need for agents to negotiate shared goals, which requires some mechanism to mediate agents’ potentially conflicting preferences. In part because they enable efficient preferential optimization, CP-nets are an attractive model for representing agents’ preference information. Hoping to benefit from this same efficiency, we apply the CP-net model to the problem of negotiating joint outcomes. We introduce a way to compare outcomes across agents based on each outcome’s relative standing in the individuals’ spaces of possible outcomes. The ability to compare outcomes across agents allows us to guide a search through the outcome preference graphs that are induced by the agents’ CP-nets and find the optimal outcome. Generating these induced preference graphs is, unfortunately, a combinatorial problem, but exploiting an additional semantic constraint often incorporated into CP-nets allows us to construct near-optimal social outcomes without first generating the induced preference graphs.

General Terms
Algorithms

Keywords
Multiagent Systems :: Argumentation, negotiation, and conflict handling
Multiagent Systems :: Societal aspects :: Social and organizational structures

1. INTRODUCTION
Every day, groups of people need to make social choices: What movie should they see? What pizza should they order? Who should be the president? A large amount of work has been aimed at deciding on outcomes that are, in some way, best for a group. This problem has been approached from many directions, including Arrow’s original work on ordering relations [1], utility-based approaches such as [2], and numerous applications involving voting theory, as in [8].

Social choice enters into the area of collaborative planning as a necessary first step to determine the goals toward which the agents ought to direct their planning effort. For instance, individual agents might have preferences over the times that activities should be done, or even the specific activity to do at a particular time, and must reach some consensus to facilitate planning. This work addresses the problem of choosing outcomes (goals) in an offline setting based on individual agents’ preferences. While this work is very much focused on finding an efficient answer to the social choice problem, we discuss how making these choices might be incorporated into the planning process in Section 8.

We assume, as a starting point, that each agent’s preferences are encoded in an acyclic CP-net. In the original CP-net paper [4], Boutilier et al. argue that a CP-net representation offers several benefits. An important strength of the representation is the ease with which it allows an agent to represent how its preference for the value of one feature depends on the values that other features take. One example of this used by Boutilier et al. is a diner whose preference for wine depends on the entrée served, so that she prefers the variable “wine” assume the value “red” if the variable “dinner” takes on a value “meat”, and that “wine” have the value “white” if “dinner” takes on the value “fish”. Also, because the network only explicitly represents preferences about the values of individual variables, a user’s CP-net can be elicited via statements that are very natural for users to make [4]. The ceteris paribus (“all else being equal”) semantics allows the user to answer questions about single variables in isolation (“If everything else the same, do you prefer red or white wine with meat?”)

In [8], Rossi, et al. adapt CP-nets to answer questions about social choice. They extend the pure CP-net semantics to allow agents’ preferences to interact, and combine the resulting networks into an mCP-net. This multi-agent representation allows outcomes to be evaluated using well-understood voting semantics. As with single-agent CP-nets, outcomes can be compared for dominance from the collective perspective, and optimal outcomes identified.

We take a very different approach, working directly with a pure CP-net representation. In this paper, we present a novel method of assessing an outcome based on its position in the space of all possible outcomes. This allows us to compare outcomes in a multi-agent setting and define an algorithm for finding optimal social outcomes by a directed search through the outcome space. Finally, we define a second way of assessing outcomes that is a good approximation of our original definition, and that can be done based purely on the information in the original CP-nets. We are able to use this definition to construct outcomes via a directed search in the variable space (rather than the outcome space), where these outcomes are optimal under the approximation we adopt. Like the single-agent forward sweep, this construction can be done in linear time. This compares favorably to [8], which requires exponential time to identify optimal outcomes, although the mCP-nets offer additional flexibility in return. We provide analytical evaluation of both the outcomes-space and CP-net-space algorithms, proving their optimality under the stated assumptions.

The remainder of the document is arranged as follows. Section 2 describes CP-nets in greater detail and summarizes the concepts that will be most important for our work. Section 3 introduces our new metric for assessing competing outcomes, even across multiple agents. Section 4 discusses optimality in the context of social choice to arrive at the criteria that we will use for comparing outcomes. Section 5 presents our algorithm for finding the optimal social outcome by searching in the outcome space. Section 6 discusses the assumption necessary to search in variable space instead of outcome space, and describes the resulting algorithm for very simple network topologies. Section 7 extends the ideas developed for very simple networks to the case of two arbitrary networks. Section 8 discusses our results, related work, and directions for further work.
2. CP-NETS OVERVIEW

In [4], Boullier et al. present a model (called a CP-net) that describes an agent’s conditional ceteris paribus preferences. Briefly, a CP-net is a directed graph, in which each node is annotated with a conditional preference (CP) table. Each node’s CP table indicates the preferred value for that node’s variable, conditioned on the values of its parents (the nodes that point to it).

\[
\begin{align*}
\text{d} & \succ \text{d}' \\
\text{D} & \rightarrow \\
\text{w} & \succ \text{w}' \\
\text{D} & \rightarrow \\
\text{d}' & \succ \text{w} \\
\end{align*}
\]

![Figure 1. The CP-net for the dinner example.](image)

Figure 1 shows a simple CP-net for Boutilier’s example diner. We have two variables, and name them with capital letters—D for Dinner is Meat and W for Wine is White. Lowercase letters (and their negations) denote values that are assumed by variables. Thus, d and d’ represent respectively, meat and non-meat (fish) dinner, while w and w’ stand for white and non-white (red) wine. For simplicity, we assume throughout that variables are binary.

2.1 Comparing Outcomes with CP-nets

Every CP-net induces an associated preference graph that describes the relationship between similar outcomes. In the induced preference graph for a CP-net, there is a directed edge between every pair of outcomes that differ on the value of only a single variable. For any pair of outcomes O₁ and O₂, that differ only on the value of a single variable V, the CP-table for V indicates which of the competing values of V is more preferable. Under the ceteris paribus semantics of the CP-net, the outcome in which V assumes a more-preferred value is the more preferable outcome. By convention, we draw the edge from the more preferable outcome to the less preferable.

\[
\begin{align*}
\text{d}, \text{w}' & \rightarrow \\
\text{d}, \text{w} & \rightarrow \\
\text{d}', \text{w} & \rightarrow \\
\text{d}', \text{w}' & \rightarrow \\
\end{align*}
\]

![Figure 2. Induced preference graph for the Figure 1 CP-net.](image)

The induced preference graph for the diner example (Figure 2) shows that \((d, w') > (d, w) > (d', w) > (d', w')\). We rank \((d, w')\) over \((d, w)\) since \(w' > w\) when D has the value d. Similarly, we rank \((d, w)\) over \((d', w)\) because \(w\) has the value \(w\) when \(d\) is preferable to \(d\). Finally, \((d', w') > (d', w')\), because \(w > w'\) when \(D\) takes the value \(d'\). The final relationship, between \((d, w')\) and \((d', w')\), can be directly established based on the differing values of \(D\), and also can be inferred as a transitive relationship.

Induced preference graphs are constructed using dominance relations, which apply when one outcome is strictly preferable to another. Like strict inequality, dominance is transitive. Thus, in the previous example, we infer that \((d, w')\) is better than \((d', w)\), even though the two outcomes are not themselves connected by a directed edge. Answering dominance queries requires searching for a sequence of single variable “flips” that produces a string of intermediate outcomes linking one outcome to another, each more preferred than the last. Boutilier et al. show that this search is combinatorial in the number of variables for all but the simplest network topologies.

The second way in which two outcomes can be related is called an ordering relation. Ordering is a weaker relationship than dominance, establishing only the non-dominance of one outcome over another. Outcome O₁ is defined to be consistently orderable over outcome O₂ if the information in the CP-net is insufficient to indicate that O₁ dominates O₂.

Unlike dominance queries, a consistent ordering between two outcomes can be found quickly. Boutilier et al. show that a pair of outcomes O₁ and O₂ may be ordered (in one direction) by considering the variables in topological order based on the CP-net. The first variable (say V) in the topological order that takes a less-preferred value in one of the outcomes (w.l.o.g., say V’s value is less-preferred in O₁) indicates that the outcome in which V takes the more-preferred value (O₂) is orderable over the other (O₁).

\[
\begin{align*}
\text{a} & \succ \text{a}' \\
\text{a} : \text{b} & \succ \text{b}' : \text{b} \\
\text{b} : \text{c} & \succ \text{c}' : \text{c} \\
\end{align*}
\]

![Figure 3. A more interesting CP-net.](image)

Although dominance implies ordering, the reverse is not also true. In some cases we find both that O₁ is orderable over O₂ and O₂ over O₁ (as in Figure 4, above). In such cases O₁ and O₂ are incomparable because the network has abstracted away the information necessary to decide whether O₁ is preferable to O₂, or whether the agent is indifferent between O₁ and O₂. If forced to choose between two such outcomes, the efficient approach is to use an ordering query. Such a query will find an outcome guaranteed to be no worse than the other.

2.2 Outcome Optimization

We have seen in constructing the preference graph in Figure 2 that the CP-net semantics make a strong claim about the relative importance of variables at different levels of a CP-net, where levels are defined as follows.

Definition 1 (Variable Level): The level of a variable \(V\), \(\text{level}(V)\), is equal to the length of the longest path in the CP-net from any root variable to \(V\). “Level \(N\)” of a CP-net is \(\langle V: \text{level}(V) = N \rangle\). We say that variables in level \(N\) are “higher level” than those in level \(K\) when \(N < K\).
In Figure 2, we infer the relationship \((d, w) > (d', w)\) because \(d > d'\); even though flipping the value of \(D\) from \(d\) to \(d'\) changes the value of \(w\) that is assigned to \(W\) from a non-preferred to a preferred choice. This remains true regardless of the number of children of \(D\). Under the CP-net semantics, an agent is at least as satisfied getting its way about a higher-level variable \(D\) as it would be to get its choice about any number of lower-level variables \(D\)'s descendents.

The importance of higher level variables allows Boutilier et al., to show in [4] that given some evidence (an assignment to a subset of the feature variables), the most-preferred outcome that is consistent with that evidence can be found in time that is proportional to the number of variables in the CP-net. The algorithm for doing this, which Boutilier et al. call forward sweep, is very straightforward: begin from the top level of the CP-net and assign to each unassigned variable the value that is most preferable given the existing partial assignment. After assigning all variables in one level, continue by assigning values to variables in the next, until every variable has been assigned a value. The semantics of the CP-net representation guarantee that the forward sweep constructs an outcome that dominates any other outcome that is consistent with the evidence.

It is simple to apply this procedure to the diner example. If some evidence sets \(W\) to \(w\), the algorithm assigns \(d\) to \(D\), returning the outcome \((d, w)\), which is, indeed preferable to \((d', w)\). If the evidence for \(w\) didn't exist, after assigning \(d\) to \(D\), the algorithm would assign to \(W\) the value most preferred given \(d\), namely \(w'\). In this case, the outcome \((d, w')\) found by the algorithm matches the outcome that we placed first in our total ordering, as desired.

3. OUTCOME COMPARISON BEYOND DOMINANCE AND ORDERING

When using CP-nets, outcomes can only be compared by dominance and ordering queries. These relations are defined only for a single agent, and cannot be directly applied in a multi-agent setting. In finding good joint outcomes, we want to be able to compare how good one outcome is for agent \(A\), compared to how good a different outcome is for agent \(A\). This sort of calculus requires that we account for two factors. First, we need to be able to make meaningful statements about how good an outcome is for an agent relative to all outcomes. For example, we want to express that outcome \((a, b, c')\) in Figure 4 is the second-best outcome for an agent with the CP-net shown. Second, we observe that the total number of outcomes for an agent affects the interpretation of this ranking: second-best of eight outcomes is better than second-best of three, for example. In this section, we develop a way of describing the degree to which an outcome is good for an agent.

3.1 Single-Agent Refinements

We can use the dominance information provided by a CP-network to infer the degree to which an outcome is good (or bad) for a single agent by measuring the position of an outcome relative to the entire "outcome space". In Figure 4, for instance, the long path from \((a, b, c')\) to \((a', b, c)\) indicates that a large number of outcomes dominate \((a', b, c)\), while a much smaller number dominate \((a, b, c')\), which implies that \((a', b, c)\) is much worse than \((a, b, c')\).

To formalize this intuition, we observe that given the partial order represented by the outcome graph induced by a CP-net, outcomes may be divided into equivalence classes, or tiers, so that all the outcomes in a class are incomparable. We assume that each outcome is better than any in a lower tier. Thus, the introduction of tiers is analogous to the introduction of additional dominance edges into the outcome graph. For a given CP-net, there are many such ways to perform this division, including any total order consistent with the partial order encoded in the outcome graph. The CP-net itself contains no information to inform the choice of division, but we offer one candidate approach here.

Definition 2 (Minimal Tier Division). The outcome space is said to be minimally divided into tiers if each outcome is placed to tier \(N\) IFF it is dominated by some outcome in tier \(N-I\), and is dominated by no outcome in tier \(K\), for all \(K \geq N\). Non-dominated outcomes are placed in tier \(1\).

We observe that this division results in a minimal number of tiers, because for a division to have fewer tiers, some outcome would need to be in the same tier as an outcome that dominated it, which violates the principle that any pair of outcomes in a tier is incomparable.

Choosing to divide outcomes into the minimal number of tiers is a reasonable default, in that it introduces the smallest number of assumptions about the ordering of incomparable outcomes. Furthermore, this choice of tiering is attractive because it respects the relationships that are found by ordering queries. That is, none of the additional relationships implied by such a tiering order a pair of outcomes differently than an ordering query would.

Finally, we note that dividing the outcomes into tiers this way has a bit of the same flavor as the idea of k-optimality [6]. Outcomes in higher tiers are better, but there is nothing to say by how much one tier is an improvement over another.

3.2 Multi-Agent Outcome Comparison

Once outcomes have been divided to tiers, any two outcomes can be compared, for a single agent, solely on the basis of their tiers, because they are being evaluated in the context of the same overall outcome space. As we observed above, the meaning of being in tier \(N\) is dependent on both the number of tiers, as well as the "shape" of the outcome space. To normalize for agents' different outcome spaces, we consider for each agent the percentage of outcomes that are definitely better and worse than a particular outcome of interest, given the chosen tiering.

Given some tier division, we define a quantitative metric that is normalized to eliminate the effects of differing outcome spaces. Based on an outcome's position in a given tier structure, we define a satisfaction interval or just interval for that outcome in the context of that preference graph, as follows.

Definition 3 (Satisfaction Intervals). Given some division of outcomes into tiers, we can describe O's position in the space of all outcomes with a satisfaction interval. We write intervals as [\(\min, \max\)], where \(\min\) is the proportion of outcomes in a lower tier than \(O\), and \(\max\) is 1 minus the proportion in a higher tier.

We introduce additional notation for describing these satisfaction intervals. Any such interval describes the quality of some
outcome $O$ from the perspective of a particular agent $A$. We write $\text{interval}(O_A)$ to refer to this satisfaction interval. We additionally write $\text{max}(O_A)$ and $\text{min}(O_A)$ to refer to the high and low endpoints of $\text{interval}(O_A)$, respectively.

Using these tiers and associated satisfaction intervals to compare outcomes across agents relies on assumptions of some degree of commensurability between agents' underlying utility functions. In particular, these functions are assumed to have about the same shape, and especially the same concavity. If we have only CP-net representations, such an assumption is necessary so make any progress. We discuss in the conclusion some of the ways that this assumption might be relaxed.

4. MAKING SOCIAL CHOICES

To evaluate outcomes from a social perspective, we need some criterion for evaluation. An argument has been made (by John Rawls in [7], for one) that rational, self-interested agents ought to favor a maximin criterion for any mechanism used to make one-time social choices. If agents have no a priori knowledge of which agent will end up being the worst off, it is in every agent's interest to ensure that whichever agent ultimately ends up the worst off is as happy as possible. Although a number of different social welfare metrics might reasonably be used, the decision to use any particular metric is of minor importance. Our work can be adapted to work with any metric that can be evaluated by comparing satisfaction intervals. For example, future work might attempt to find outcomes that, in addition to being maximin, are also Pareto optimal.

We can decide which agent is worse-off under a particular outcome by comparing the agents' satisfaction intervals (Section 3). For example, consider agents $A_1$ and $A_2$ with the outcome graphs shown in Figures 4 and 5. Given a choice between outcomes $O_1$ ($a, b, c$), and $O_2$ ($a', b', c'$), we must consider four intervals: $\text{interval}(O_1, A_1)$, $\text{interval}(O_1, A_2)$, $\text{interval}(O_2, A_1)$, and $\text{interval}(O_2, A_2)$. We see that $A_1$ has an interval of $[.625, .75]$ for $O_1$ and $O_2$ since the outcomes are in the same tier. $A_1$ has an interval of $[.625, .75]$ for $O_1$ and $[.75, .875]$ for $O_2$. Thus, $A_1$ is worse off under outcome $O_2$, while $A_2$ is worse off under $O_1$. We finally observe that $A_2$ is worse off with the outcome $(a, b, c')$ than $A_1$ is with $(a', b', c')$, so we choose to disappoint $A_1$. Thus, $O_2$ is the better of these two outcomes from a maximin perspective.

To find the "better" or "worse" of a pair of intervals, we define a way to compare two intervals. When comparing two intervals, we write $\text{interval}(O_1, A) > \text{interval}(O_2, A)$ to indicate that agent $A$ is happier with $O_1$ than agent $A_2$ is with $O_2$. Since we are looking for maximin outcomes, the comparison of intervals reduces to comparing their lower endpoints, using the max to break ties. This observation forms the basis for our algorithm in the next section for finding optimal social outcomes.

5. FINDING OPTIMAL OUTCOMES

If efficiency is of no concern, we can use the induced preference graph for each agent to find the best social outcome. Generating, and even representing the induced preference graph for a CP-net turns out to be combinatorial in the number of variables, which makes it impractical as the basis of an algorithm. If, however, we assume that this information is available for each agent, finding the best social outcome turns out to be extremely simple.

Given the induced preference graph for each agent, we can, as described, divide the outcomes into tiers, and identify the interval that describes the outcomes in each tier. Any division into tiers may be used, provided that it respects the constraint that outcomes in the same tier are incomparable. For example, in Figure 5, the minimal tiers have associated intervals of $[1, .875]$, $[.875, .5]$, $[.5, .125]$, and $[.125, 0]$. Note that all of the outcomes in a tier have identical intervals. Once we have identified the intervals for each tier of outcomes for each agent, finding the socially optimal outcome is straightforward.

![Figure 5. CP-net and induced preference graph for three unconditioned variables.](image)

![Figure 6. Algorithm 1 finds an optimal social outcome.](image)

Figure 6 describes the algorithm for finding optimal social outcomes. Although for the sake of clarity it is presented for only two agents, the generalization to $N$ agents is trivial. The data structures outcomes1 and outcomes2 list the tiers associated with each agent's outcome space. Each tier (list entry) contains the set of outcomes in that tier that are consistent with any external evidence, and the min for the interval associated with those outcomes. The algorithm maintains a set of candidate outcomes for each agent, as well as a counter. This counter slowly decreases from 1, and whenever it falls below the min of the interval for an agent's top remaining tier, the outcomes in that tier are added to the set of candidate outcomes for that agent. Note that value of .01 for the counter decrement is arbitrary but unimportant, provided that the decrement is smaller than the difference in mins for successive tiers. Once the intersection between the sets of
candidate outcomes is non-empty, we need only to check the max values of the outcomes in the intersection.

We briefly describe the execution of the algorithm for agents $A_i$ and $A_j$ again using Figures 4 and 5. Initially, both agents have empty candidate lists, and the counter is at 1. The counter ticks down until it drops below 375, at which point $(a, b, c)$ is added to the candidates for $A_i$ and $(a', b', c')$ is added for $A_j$. The intersection is empty, so we continue. When the counter reaches 375, $(a, b, c)$ is added for $A_i$. Then, at 625, $A_i$ adds $(a', b', c')$. Finally, when the counter reaches 5, $A_i$ adds outcomes $(a', b', c'), (a', b, c'),$ and $(a, b, c')$. At this point, $(a, b, c')$ is in the intersection, it is returned as the best joint outcome.

**Theorem 1:** The outcome returned by Algorithm 1 is maximal.

**Proof:** Consider the final iteration of the whole loop. There are two cases to consider. In the first case, only one of the IF statements evaluate to true. Assume w.l.o.g., it was the first (line 3). A set of outcomes which we'll call O, is added to cand1, which produces a non-empty intersection in the WHILE condition. Because they are in the same tier, every outcome in $O$ has the same interval, and so they all have the same max, and the final step for loop might choose any of them. Consider a particular outcome $o$ in $O$. We know from the assumption that only the first IF statement was true in the final iteration that $o$ was in cand2 before the final iteration, and so, interval($o, A_j < interval(o, A_i)$ and $A_i$ is worse-off under outcome $o$. Consider any other outcome $o'$ not in $O$. Either min($o', A_j) < min(o, A_j)$ or min($o', A_j) < min(o, A_j)$, because otherwise $o'$ would have occurred in the intersection of cand1 and cand2 during some earlier iteration of the loop body. Thus, any outcome $o'$ that is not in the initial non-empty intersection is worse for one agent than $o$ is for $A_i$.

In the second case, both IF conditions are true during the final iteration, and so outcomes $O_1$ are added to cand1 and $O_2$ to cand2. If their intersection contains only outcomes from one of $O_1$ or $O_2$, we are back in the first case. If, however, the intersection contains some outcome from both $O_1$ and $O_2$, we need to make the additional check. The final step of the algorithm computes max($o, A_j$) for the outcomes in $O_1$, and max($o, A_j$) for outcomes in $O_2$. The algorithm returns an outcome $o$ for which this max calculation is higher (say w.l.o.g., $o$ is in $O_2$), which must be maximin. Any outcome $o'$ in $O_1$ is, by the same argument we used for case 1, worse for $A_j$ than for $A_2$. But $o'$ has min($o', A_j) = min(o, A_j)$ and max($o', A_j) < max(o, A_j)$, so that interval($o', A_j$) < interval($o, A_j$). Similarly, any outcome $o'$ not in $O_1$ or $O_2$ is, by the same argument as in case 1, worse for one agent than $o$ is for $A_1$. Thus, outcome $o$ is maximin.

If we have more agents, the algorithm is optimal for exactly the reasons given in the two-agent proof. The outcomes $O$ that first appear in the intersection are all the same min, and any outcome $o'$ not in $O$ has a smaller min for some agent, because for some $k$, min($o', A_j) < min(o, A_j)$, for any $o$ in $O$ and all $j$. As in the two-agent case, we only need to check the max values of the outcomes in the intersection, and only when the last iteration adds outcomes to the candidate sets of more than one agent.

This algorithm is problematic in its reliance on the availability of the induced outcome preference graph for every agent, which is a combinatorial problem. Boutilier et al. resolve the single-agent case by working directly with the CP-net in "variable space" rather than first building the preference graph and working in "outcome space". Under the assumption that there is no additional preference information that the CP-net is unable to capture (that would invalidate a minimal tiering), their algorithm finds an outcome in the highest tier that contains any outcomes consistent with the evidence. The remainder of this paper investigates the conditions that are necessary for us to find optimal social outcomes while working directly with CP-nets. By remaining in variable space in the multi-agent case, and performing what amounts to a parallel forward sweep, we avoid the need to construct each agent's induced preference graph.

6. **MULTI-AGENT FORWARD SWEEP**

To construct an optimal outcome by working directly with agents' CP-networks, our basic approach is similar to the single-agent forward sweep. Agents assign variables in topological order, setting them to the values that they prefer. By keeping track of agents' intervals as more variables are assigned, we can ensure that the outcome constructed is maximin.

Unfortunately, computing agents' satisfaction intervals relies on the availability of the induced preference graphs to determine which outcomes are better or worse than a particular outcome. In order to compute satisfaction intervals for outcomes or partial assignments while considering only CP-nets, we need rely on an additional assumption.

The problem of finding good joint outcomes is analogous to the single agent problem of deciding which of two sets of assignments to evidence variables an agent would prefer. (We can think of these evidence sets as being the assignments made by the other agent). Unlike outcome optimization, the single-agent CP-net formalism is unable to efficiently answer this question. The answer can be found by first finding the best outcome consistent with each candidate evidence set and then comparing them, but choosing the better of these outcomes is NP-hard. If we need to be more efficient, we must, instead, use ordering queries.

We observed in Section 2 that, in comparing two outcomes, if variable $V$ takes on different values in the two outcomes, but all of $V$'s ancestors have the same values in both outcomes, then the outcome in which $V$ gets a more preferable value is orderable over the outcome in which $V$'s value is less-preferred, regardless of the values of $V$'s descendents in the competing outcomes. The consequence of this is that the variables in one level of a CP-net must collectively be at least as important as all the variables in lower levels, combined. It is this same principle that we apply to the multi-agent forward sweep.

Numerically, this implies at least 50% of an agent's total satisfaction is determined by the top-level variables, 25% by the second level, and so on. As a result, we can compute the satisfaction interval for any outcome by beginning at [1, 0], and iteratively refining this interval based on the values of the variables in each level.

We first define functions to count the assignments that are better and worse than a particular assignment to a level's variables. We use these counts to define the local satisfaction interval for some assignment to a level's variables, by converting the straight counts to proportions of the number of possible assignments.
Definition 4: For an assignment $A$ to the $n$ variables in a single CP-net level $L$, in which $k$ variables are assigned their less-preferred value, we define NumBetter($n$, $k$) = $\sum_{i=1}^{k} \frac{n!}{i!(n-i)!}$ and NumWorse($n$, $k$) = $\sum_{i=k+1}^{n} \frac{n!}{i!(n-i)!}$. Where the $\frac{n!}{i!(n-i)!}$ are binomial coefficients. By definition, NumBetter($n$, 0) = NumWorse($n$, $n$) = 0.

Definition 5: For an assignment $A$ to the $n$ variables in level $L$, in which $k$ variables are assigned their less-preferred value, we define localMax($A$, $L$) = 1 - NumBetter($n$, $k$)/NumWorse($n$, $k$). We similarly define localMax($A$, $L$) and localMax($A$, $L$) as the endpoints of this interval.

Because of the assumption about the relative importance of different levels, we can compute an agent’s satisfaction interval for an assignment to all the variables in the highest $N$ levels of a CP-net by iteratively narrowing an interval based on the assignment to each level, beginning with [1, 0].

Definition 6: Assume level $L$ of the CP-net $C$ contains $N$ variables, and that an assignment $A$ to all higher-level variables has interval($AC$)=[oldMax, oldMin]. Further assume that an assignment $A'$ to the variables $L$ produces $k$ bad assignments. Then, the new satisfaction interval for the assignment $A \cup A'$ is [newMax, newMin] where 

newMax = oldMax - NumBetter($n$, $k$)/oldMax $-$ oldMin, 
newMin = oldMin + NumWorse($n$, $k$)/oldMax $-$ oldMin

Because the quantities (NumBetter($n$, $k$)/oldMax $-$ oldMin) and (NumWorse($n$, $k$)/oldMax $-$ oldMin) are always at least zero, we know that newMax $\geq$ oldMax and newMin $\geq$ oldMin. Thus, as an agent considers more levels in its network and iteratively refines its interval, its max is non-increasing, while its min is non-decreasing.

These definitions enable us to derive the following useful result.

Theorem 2: If every variable in the first $j$ levels of a CP-net takes on its preferred value, the interval for the partial assignment to those $n$ variables is [1, 1/2$^j$], independent of the network topology for those variables.

Proof: The max is trivial. At every level, NumBetter = 0, and thus newMax = oldMax = 1. We prove the min by induction. If $n=1$, Theorem 2 specifies newMin = 0 + (1/2)$^1$ = 1 = (1/2)$^1$, as desired. Now, we assume that Theorem 2 is true for $n=q$ and consider $n=q+1$. There are two cases depending on the network’s topology. If the $k$th variable $V$ is alone in level $L$, the first $k$ variables fit the conditions of the theorem. Then, by the induction hypothesis, the interval after considering the first $k$ variables is [1, 1/2$^k$]. Then, by Theorem 2, the interval after considering level $L$ is [1, (1-1/2$^k$)/(1-1/2$^k$)] = [1, 1/2$^{k+1}$].

Finally, we consider the case where $V$ is in a level $L$ with another variable or variables. Assume that levels 1 through $l-1$ contain $x$ variables, while $L$ contains $y$, with $x+y=q+1$. By the inductive hypothesis, the interval after level $L$ is [1, 1/2$^j$]. Every variable in $L$ is assigned a preferred value, so by our formula, newMin = $1 - 1/2^j + (1/2)^j$ numerBetter($y$, 0). We know that newMin = $1 - 1/2^j + (1/2)^j$ numerWorse($y$, 0). So newMin = $1 - 1/2^j + (1/2)^j$ numerWorse($y$, 0). NewMax = $1 - 1/2^j + (1/2)^j$ numerBetter($y$, 0). So the interval is [1, 1/2$^j$].

With this framework in place, we can formalize what it means to perform a forward sweep with multiple agents.

6.1 Forward Sweep in Simple Networks

We begin by considering the case of two agents, each of which has a "linear" CP-net with a branching factor of 1. The basic idea is to proceed level-by-level down the CP-nets, allowing agents to assign variables in parallel. The key point in the algorithm occurs at the first level in which one agent $A_1$ assigns a value to its variable (or finds that the variable already has a preferred value), while the second agent $A_2$ cannot, because its variable was already set to a value $A_1$ dislikes. Say that this occurs in level $n$.

By Theorem 2, under the current assignment $CA$, min($CA$, $A_2$) = 1 - 1/2$^n$, while max($CA$, $A_2$) = 1 + 1/2$^n$. Because max is non-increasing and min is non-decreasing, interval($O$, $A_2$) $>$ interval($O$, $A_1$) for any outcome $O$ that is consistent with $CA$. To find a maximin outcome, $A_2$ should get its choice of every remaining variable.

**Algorithm 2**

1. init a1Int = [1, 0], a2Int = [1, 0], level = 1
2. if (a1Int.max > a2Int.min & a1Int.max > a1Int.min) then
3. a1Int = a1Int.min & a2Int = a1Int.min
4. else
5. a2Int = a2Int.min & a1Int = a1Int.min
6. a1Int = a1Int.min & a2Int = a2Int.min
7. a1Intupdate = a1Int
8. a2Intupdate = a2Int
9. level += 1
10. if (interval1.min > interval2.max) then
11. a2Int = a2Int & RemainingVars
12. else
13. a1Int = a1Int & RemainingVars

Figure 7. Algorithm 2 performs a parallel forward sweep for two agents.

Pseudocode for the two-agent forward sweep is provided in Figure 7. Each agent has an array of variables called "vars" sorted in decreasing topological order. The min($O$, $A_2$) functions in line 3 use the min function we've defined to return the smaller endpoint of an interval. The $O_2$ and $O_1$ used here are special outcomes. $O_1$ is the outcome found by assigning $a1.vars[level]$ to the value preferred by $A_1$, and assigning every remaining variable by using an ordinary single-agent forward sweep for $A_2$. $O_2$ is found similarly. Thus, line 3 is deciding which agent ultimately does better if it fails to get its choice for the variable at hand.

Theorem 3: The outcome constructed by Algorithm 2 is maximin given our definition of agents' satisfaction intervals.

Proof: Consider the outcome $O^*$ returned by the algorithm, and assume $interval(O^*, A_1) > interval(O^*, A_2)$. Furthermore, assume that $V$ is the highest-level variable in $A_1$'s CP-net whose value is not preferred by $A_2$ and that $V$ is at level $N$. Then, by Theorem 2, $min(O^*, A_2) \leq 1 - 1/2^N$. Consider the case where $V$ was set by $A_1$ in some level $L$ higher than level $N$, so that $L < N$. Consider any outcome $O$ in which $V$ takes the value preferred by $A_2$ (and not $A_1$). By Theorem 2, $max(O, A_1) = 1 - 1/2^N \leq min(O, A_2)$, so $A_1$ is less happy under $O$ than $A_2$ is under $O^*$. 

14
Now consider any variable \( W \) at a lower level for \( A_2 \) than \( V \) that takes a value under \( O^* \) that is not preferred by \( V \). To take a value that is not preferred by \( A_N \), \( W \) must have either been set as evidence, or assigned by \( A_j \) in some level above level \( N \). In the first case, \( W \)'s value is fixed. In the second, any outcome \( O' \) constructed by flipping \( W \)'s value is worse for \( A_1 \) than \( O^* \) is for \( A_2 \) by the argument used above.

Finally, if \( V \) is in the same level for both agents, the algorithm explicitly constructs the maximin outcome. Line 3 compares the mins of two hypothetical intervals: the interval for \( A_S \) given the outcome that \( A_2 \) would eventually construct, assuming \( A_1 \) gets its choice \( V \), and the analogous interval for \( A_1 \). Say that this min is larger for \( A_2 \). Then, \( A_1 \) gets its choice for the value of \( V \), and \( A_2 \) realizes this min. And, by our construction, this min is larger than the min for \( A_1 \) if \( A_2 \) instead received its choice for \( V \). Thus, the outcome constructed by our algorithm is maximin. ■

![Figure 8: A second linear CP-net](image)

As an example, suppose one agent has the network from Figure 3, and the other has the network shown here as Figure 8. In the first level, the agents have different variables, so each assigns its most preferred value to that variable. The current assignment is updated to include \((a, b')\). Then, each agent updates its interval from \([1, 0]\) to \([1, S]\). At the next levels, the agents again attempt to assign values to their variables. \( A_2 \) moves up to \([1, .75]\), while \( A_1 \) drops to \([.75, S]\), since it is unhappy about the value in the second level. At this point, \( A_2 \)'s min is as large as \( A_1 \)'s max, so \( A_1 \) assigns values to its remaining variables. In this case, there is nothing to do, since \( C \) has already been assigned, so the final outcome is \((a, b', c')\), and \( A_1 \)'s final interval is \([.75, .625]\), while \( A_2 \)'s is \([.875, .75]\).

7. SEARCHING IN ARBITRARY CP-NETS

When the CP-nets are linear, the forward sweep reaches a critical point the first time it encounters a variable that is set to a value that is less-preferable for some agent \( A \). Once this occurs, the outcome ultimately constructed by the algorithm is guaranteed to be worse for \( A \), so the algorithm insures that \( A \) does as well as it can the rest of the way. In moving to more complicated networks, our approach is similar. We identify the point at which one agent is guaranteed to do worse for any possible outcome, and then award that agent its remaining choices.

7.1 Networks Of Identical Topology

Before moving to truly arbitrary networks, we consider the case of networks with identical topologies. As the agents sweep down their respective networks, unless the agents are in complete agreement, there again must be some variable \( V \) that is the first variable encountered by an agent that is already set to a bad value. Say that in \( N_1 \), \( V \) is at level \( j \), which contains \( n \) variables and that in \( N_2 \), \( V \) is at level \( j_k \), which contains \( n \) variables and \( i < j_k \). Furthermore, say that levels 1 through \( i-1 \) contain \( x \) variables, and levels 1 through \( j-1 \) contain \( y \) variables. The key consequence of the networks having identical topologies is that \( x+n \leq y \).

Assume that in outcome \( O \), \( V \)'s value is bad for \( A_2 \). If, however, \( V \) isn't assigned a value by the evidence, its value was earlier set by \( A_1 \). By flipping the value of \( V \) to construct outcome \( O' \), we could give \( V \) the value preferred by \( A_2 \) at the expense of \( A_1 \). To evaluate this proposal, we compute the satisfaction intervals \( interval(O, A_j) \) and \( interval(O', A_j) \). We find that \( min(O, A_j) = 1 - 1/2 + \text{(minWorse(m1, b-2)} \times (1/2^2) \), which we re-write as \( 1 - 1/2^2 + \text{b-2} \), where \( b > 0 \). Similarly, we know \( \max(O', A_j) = 1 - 1/2^2 + \text{b-2} \).

Because \( y \leq x+n \), this implies that \( \min(O, A_j) > \max(O', A_j) \).

Because of agents' intervals are computed, after the first level \( L \) where an agent finds some variables that have less-preferred values, if \( A_2 \) has more such variables than \( A_1 \), then \( \min(O, A_j) > \max(O', A_j) \) for any outcome \( O \) consistent with the current assignment. As in the linear case, \( A_2 \) assigns every remaining variable. We showed above that any variable that \( A_2 \) is unhappy about due to an earlier assignment made by \( A_1 \) ought not to be re-assigned. If, on the other hand, \( A_1 \) and \( A_2 \) find an equal number of variables with less-preferred values, their updated intervals are equal after level \( L \), and the sweep continues.

We again need to be careful in assigning the variables that occur in level \( L \) for both agents. If possible, they should be assigned so that each agent ends up unhappy about the same number of variables in \( L \). Otherwise, assignments are made so that the agent unhappy about more variables in \( L \) is happy about as many as possible. Finally, it is possible that we must choose to make either \( A_1 \) or \( A_2 \) less happy with the variable assignment. As in the linear case, we choose which agent to disassociate by testing the intervals of the worse-off agent under both competing assignments.

7.2 Relaxing The Identical-Topology Criterion

When we relax the assumption that agents' networks share a uniform topology, the same general principles apply, but there are many more subtleties to be aware of. When the networks are of the same topology, the forward sweep can proceed level-by-level and be sure that the agents' intervals shrink at the same rate. When the networks might have different topologies, the forward sweep algorithm needs to keep track of each agent's interval, and calibrate the rate at which each agent can assign variables so that the intervals again change at the same rate.

We can use some of the same techniques as we do in the identical topology case to show that a greedy assignment is optimal. The earlier proof relies on the fact that a variable occurs strictly earlier in any order for one agent than the other (this is the \( x+n \leq y \) observation in the first paragraph of 7.1). By counting how many variables each agent has assigned and looking ahead to the size of the next level, we can decide when an agent can assign to the variables in that next level.

This approach works up until the point that an agent looks ahead and sees that assigning preferred values to the variables in its next level will bring it into conflict with variables in the other agent's network. When this happens, it is unclear at what point the agent should be allowed to begin assigning variables in that next level. In particular, the order in which variables are considered remains very important. If an agent \( A_j \) has its choice of four variables in a level to next assign, it should assign the variable that is at the
lowest level in $A_2$'s CP-net. The reason for this is that $A_2$ may suddenly find that it is worse off under any possible outcome (as in the simpler cases) and needs to assign the rest of the variables. In this case, $A_2$ should have created potential conflicts with the lowest-level variables in $A_2$'s network that it could. We are still working out the final details of the general case.

8. DIRECTIONS FOR FUTURE WORK
In this paper we have made several assumptions. Some of these, like the assumption of binary variables, and focusing our attention on the two-agent case in Sections 6 and 7, serve to simplify the description of our approach. Adapting the approach to relax these assumptions would be valuable, if somewhat trivial next step.

Other assumptions cannot be so easily relaxed. For example, to allow for comparison of outcome quality across multiple agents, we assume, for lack of information to the contrary, that the agents are similar in the shape of their utility functions. If some evidence were available to contradict this, the algorithms could be adapted to accommodate the discrepancy. For instance, if one agent has weaker preferences relative to the other, we might give the second agent a "head start" in the search, allowing it to assign more variables faster. In general, however, such issues are more easily addressed in a more quantitative framework.

We purposely worked with the foundational version of CP-nets. Although the representation can be adapted to be more powerful by introducing richer semantics, pure CP-nets have their own advantages. In particular, they provide the most natural semantics for agents to supply conditional preference information. If we consider richer CP-net representations, several augmented models might be profitably adapted to guide social choices, especially UCP-networks [3]. By attaching utilities to the values for variables, this representation provides much more power to evaluate outcomes socially, while at the same time offering some of the same advantages of CP-nets.

The algorithms presented here are offline algorithms, operating under the paradigm that agents have some preferences and some set of evidence, from which they produce outcomes—full assignments to all feature variables. We are beginning to consider problems with added temporal dynamics. We envision a model in which agents must make decisions—typically about variable bindings—subject to some temporal constraints, while receiving evidence in a manner also subject to some temporal constraints.

In practice, it is likely that agents have many local variables and only a few held in common. This particular framework provides ample opportunity for the simplification of the problem. One approach is preferential decoupling, similar in spirit to temporal decoupling [5]. If agents first resolve the joint variables, the problem decomposes into many single-agent problems, which can be solved very efficiently. The difficulty here lies in making good decisions about the joint variables even though the values of their parents may not all be known.

9. CONCLUDING REMARKS
By assessing outcomes based on their positions in agents' overall preference spaces, we are able to distinguish between outcomes that are impossible using just dominance and ordering queries. This allows us compare the quality of outcomes from the perspectives of different agents. This comparison method, in turn, provides the basis for an efficient algorithm to find an optimal (maximin) social outcome. Although our particular definition of tiers and intervals is only one of several similar metrics that can be chosen, our algorithm could operate similarly regardless of the precise definition of agents' satisfaction intervals.

Searching agents' outcome spaces requires the creation and maintenance of preference graphs that are exponential in the number of variables. To avoid this problem, we exploit an additional semantic assumption that allows us to construct maximin outcomes by a traversal of agents' CP-nets, using an approach that is very similar to the single-agent forward sweep. Analytically evaluating our algorithm, we prove that the outcome found is socially optimal under the additional semantics.

ACKNOWLEDGEMENTS
The work reported in this paper was supported, in part, by the National Science Foundation under grant IIS-0534280. We would like to thank Martha Pollack and Erin Rhode for valuable feedback on this work.

10. REFERENCES
Privacy Loss in Multiagent Planning

A Classical Definition with Illustration

Roman van der Krogt*
Cork Constraint Computation Centre
Department of Computer Science
University College Cork, Cork, Ireland
roman@4c.ucc.ie

ABSTRACT
Privacy is often cited as the main reason to adopt a multiagent approach for a certain problem. This also holds true for multiagent planning. Still, papers on multiagent planning hardly ever make explicit in what ways their systems protect their users' privacy, nor do they give a quantitative analysis. The reason for this is that a theory of privacy loss in multiagent planning is virtually non-existent so far.

This paper proposes a measure for privacy loss based on Shannon's theory of information. To illustrate our approach, we apply this metric to an existing multiagent planning system to assess its merits when it comes to privacy on two domains. For this, we compare its plans to centrally generated solutions (by a trusted third party) for the same problems. The results clearly establish the need for such an analysis; even though the multiagent planner seemingly exchanges little information, its overall performance with respect to privacy is less than that of the centralised system (not taking into account the privacy loss with respect to the central planner, of course).

1. INTRODUCTION
The literature names a great many reasons why to pursue multiagent planning. One of the reasons that often comes up is that of privacy. Especially in circumstances where the agents represent (possibly competing) companies, sharing data with other parties is considered undesirable. At the same time, it is well recognised that cooperation may be mutually beneficial to all parties. An example of this is the reduction of empty rides in transportation. Transportation companies are often faced with deliveries of cargo from point A to point B, but find themselves without a matching order for the return. Often, another company might be in the same situation in the opposite direction: it has a load from B to A, which are close to respectively points A and B. By cooperating on the two tasks, they can spare themselves the empty return rides, which is beneficial for both parties (and the environment). However, the companies are also each others competitor. As such, they have a natural tendency to distrust one another. By revealing too much information, a company may find itself in a situation where the competitor has such knowledge of the order books and cost structures that it can consistently undercut bids on tenders.

Multiagent planning is one of the tools that can help agents in a situation such as described above. It offers a way to cooperate while being in control of which information is shared and with whom. Yet, it is impossible to cooperate without sharing any information. At the very least, a pair of cooperating agents has to agree on which subtasks are being carried out by one agent on behalf of the other [13]. Several approaches, such as (Generalised) Partial Global Planning (GPSP; see, for example, [2]), go even further and share detailed parts of their plans. In the latter approach, more information is exchanged. Clearly, this must lead to a poorer performance when it comes to privacy. This raises the obvious question of how to measure this performance. Just how much privacy is lost by exchanging certain information? How can we evaluate which method is better then another when privacy is concerned?

This paper introduces a measure for privacy loss in multiagent planning based on Shannon's Theory of Information [8]. We show how the concept of "uncertainty" that underpins Shannon's work can be interpreted in the context of plans, and how we can derive a measure from this for the information that is gained during negotiations on plan construction or coordination. We then apply this measure to an existing multiagent planner, called MPPOPR [13], and show its privacy

*Roman van der Krogt is supported by an Irish Research Council for Science, Engineering and Technology (IRCSET) Postdoctoral Fellowship.
behaviour on two domains, compared to a centralised system using a trusted third party. In the latter case, agents extract information about the other agents’ plans from the plan they are given from the central planner.

The remainder of the paper is organised as follows. In the next section we will introduce our measure for privacy loss. Then we introduce the multiagent planning system that we applied our work to and the two domains for which we did our analysis. After examining our findings, we discuss some related work from the field of distributed constraint satisfaction. Finally, we draw some initial conclusions and expand on future work.

2 CHARACTERISING PRIVACY LOSS IN MULTIAGENT PLANNING

If we want to speak about a loss in privacy, we have to obtain a measure to compare different cases. Information theory can give us this measure. The key ideas that we introduce here come from Shannon [6]. Whereas Shannon followed a rigorous route in deriving his famous function, we follow the more intuitive explanation given by Schneider [7] in setting out the background.

2.1 Information and Uncertainty

Information is closely linked to uncertainty. Suppose we have \( M \) differently coloured balls in a hat, and we intend to randomly draw one. Now we have a certain degree of uncertainty regarding the colour of the ball we will draw. When we draw a ball, we get some information (on the colour of this ball) and our uncertainty (regarding the colour of this ball) decreases. Shannon’s work gives an answer to the question of how to measure this uncertainty. If we assume an equal probability for all of the balls, we would like to say that we have an “uncertainty of \( M \) colours”. However, we would like our measure of uncertainty to be additive, which leads to the following formula for the uncertainty \( H \):

\[
H = \log_2(M)
\]

(1)

If we intend to draw a second ball there are two situations: either we put the drawn ball back or not:

1. If we return the ball that was drawn, again we have an uncertainty of \( \log_2(M) \). Thus, the total uncertainty that we have in drawing two balls is \( 2 \log_2(M) = \log_2(M^2) \) in this case.

2. If we do not return the ball, we obtain an uncertainty for the second ball of \( \log_2(M - 1) \). Hence, the reduction in uncertainty, or the information gained, by drawing a ball in this situation is \( \log_2(M) - \log_2(M - 1) = \log_2\left(\frac{M}{M - 1}\right) \).

So far, we have considered the situation where each outcome (i.e., each colour) had equal probability. But what if there are fewer colours than balls, with some colours more likely than others? First, let us rearrange Equation 1 in terms of the probability \( P \) that any colour is drawn:

\[
H = \log_2(M) = -\log_2\left(\frac{1}{M}\right) = -\log_2(P)
\]

(2)

Now, let \( P_i \) be the probability of drawing colour \( i \), with \( \sum_{i=1}^{\infty} P_i = 1 \). The “surprisal” [11] of drawing the \( i \)th colour is defined by analogy with \( -\log_2(P) \) to be

\[
u_i = -\log_2(P_i)
\]

(3)

In the generalised case, uncertainty is the average surprisal for the infinite string of colours drawn (returning each ball before drawing a new one). For a string of finite length \( N \), with colour \( i \) appearing \( N_i \) times, this average is

\[
\sum_{i=1}^{M} \frac{N_i}{N} v_i
\]

(4)

For an infinite string, the frequency \( \frac{N_i}{N} \) approaches \( P_i \), the probability of drawing this colour. The average surprisal would now be:

\[
\sum_{i=1}^{\infty} P_i v_i
\]

(5)

Substituting for the surprisal (cf. Equation 3), we get Shannon’s general formula for uncertainty:

\[
H = -\sum_{i=1}^{\infty} P_i \log_2(P_i)
\]

(6)

Notice that the unit for uncertainty is bits per symbol. The \( H \) function forms a symmetrical (multidimensional) curve that peaks when all symbols are equally likely and falls towards zero when one of the symbols becomes dominant.

At the start of this section, we said that information can be considered to be the decrease in uncertainty. Using Equation 6, we can express information. Information relates to communication and uncertainty as follows. Suppose we have an uncertainty \( H_{\text{before}} \) before an event (such as the transmission of a message) and that uncertainty after the event is \( H_{\text{after}} \). Then the information that was gained in the event equals

\[
R = H_{\text{before}} - H_{\text{after}}
\]

(7)

This is what Shannon calls the rate of information transmission. Thus, information always relates two points in time, and the uncertainties at those times.

2.2 Uncertainty with respect to Plans

Having established a general measure of information and uncertainty in the previous subsection, how are we to apply this to planning? If we go back to the classical definition of planning, we consider a plan \( \Delta \) to be a (totally-ordered) sequence of actions from a set \( O = \{o_1, o_2, \ldots, o_m\} \) of possible actions that brings about a state change from the current state \( I \) to some goal state \( G \). Hence, a plan is a sequence \( o_1 \cdot o_2 \cdot \cdots \cdot o_n \) where \( o_i \in O \) is the action executed at time point \( i \). Under the assumption that the occurrence of each \( o_i \) is independent from the sequence \( o_1 \cdot o_2 \cdot \cdots \cdot o_{i-1} \) of previous
actions, the uncertainty about which action \(a_i\) is executed at time step \(j\), follows straightforward from Equation 6:

\[
\sum_{i=1}^{[\Omega]} P_i \log_2(P_i)
\]

Here, analogously to Equation 6, \(P_i\) is the probability of action \(a_i\) being executed. Unfortunately, we cannot establish the probability of a given action. In some cases, however, we have access to previous plans, from which we can estimate the probability by the frequency \(F_i\) that an action \(a_i\) occurred in the past. This leads to the following formula for the uncertainty \(H_\Omega\) regarding the execution of an action at a certain time step:

\[
H_\Omega = -\sum_{i=1}^{[\Omega]} F_i \log_2(F_i)
\]  

(8)

Notice that this is the average surprise for encountering a certain action in any timestep. A plan consists of a number of actions, however, for successive timesteps. Because of the additive property of the uncertainty measure, we can extend Equation 8 to the uncertainty \(H_{\Delta,T}\) of a plan \(\Delta\) over a certain time window \([1 \ldots T]\) as follows:

\[
H_{\Delta,T} = T \times H_\Omega = -T \sum_{i=1}^{[\Omega]} F_i \log_2(F_i)
\]

(9)

2.3 Privacy Loss

Now that we have defined the concept of uncertainty in the context of planning, we can define the privacy loss that agents incur by exchanging information with other agents while constructing a multiagent plan. Privacy relates to the amount of information other entities have about you. In the case of multiagent planning, it relates to the information other agents have about your plan.

Consider the case of two agents, entering into negotiations for some aspect of their planning problems. Before the negotiations, the uncertainty with regard to the other agent’s plan is governed by Equation 9. During their negotiations, agents may learn about certain aspects of the other agent’s plan. In particular, they may receive information about certain actions (not) being executed at a certain timepoint. Let \(O_t \subseteq \Omega\) be the set of actions that are known to be possible at timestep \(t\), and let \(F_i \mid O_t\) be the restriction of \(F_i\) to this set \(O_t\). Then the uncertainty after the negotiations equals

\[
H_{\Delta,T} = -\sum_{i=1}^{[\Omega]} \sum_{t=1}^{T} F_i \mid O_t \log_2(F_i \mid O_t)
\]

(10)

Notice how this reduces to Equation 9 when \(O_t = \Omega\) for all \(1 \leq t \leq T\).

A few issues arise, however, that prevent us from simply combining Equation 7 with Equations 9 and 10. Firstly,

\[1\] This assumption is, of course, too strong in practice. See the discussion on how to relax it.

\[2\] Thus, \(F_i \mid O_t = \frac{N_i}{\sum_{j \in O_t} N_j}\), where \(N_i\) is the number of times action \(a_i\) was encountered in the plans sampled to estimate \(P_i\).

the uncertainty with respect to a plan \(H_{\Delta,T}\) depends on the length \(T\) of that plan. However, in general agents do not know the length of the other agent’s plan. This gives rise to a number of different measures of privacy loss for different choices of estimating the length of the plan, as we see below. Second, because of synchronisation issues, agents may have to wait in their plan, not executing any actions at all. We can resolve this by introducing an additional \(idle\) action to the set \(O\) of allowed actions. This action has neither preconditions nor effects and is used to pad agents’ plans so that all plans of length \(T\) have \(T\) actions. We define \(O^+ = O \cup \{idle\}\) to denote the expanded set of actions.

As indicated, different estimates of the plan length result in different measures of privacy loss. The basic equation from which these different measures derive is the following:

\[
R(T) = -T \sum_{i=1}^{[\Omega]} F_i \log_2(P_i) + \sum_{t=1}^{T} \sum_{\alpha \in O_t} F_i \mid O_t \log_2(F_i \mid O_t)\]

(11)

This equation is derived from Equation 7, substituting Equation 9 for \(H_{\text{before}}\) and Equation 10 for \(H_{\text{after}}\) and taking into account the \(idle\) action.

To define the different measures of privacy loss, consider a group of agents \(A = \{a_1, a_2, \ldots, a_N\}\) with respective plans \(\Delta_{a_i}\) for \(1 \leq i \leq N\). If we assume that the agents do not share their knowledge about other agents’ plans, a loss of privacy always occurs between a pair of agents. Let the target agent \(\tau\) be the owner of the plan \(\Delta_{\tau}\) under consideration, and the invading agent \(\iota\), \(\iota \neq \tau\), be the agent that has gained some knowledge about \(\Delta_{\tau}\). We can now distinguish the following measures of privacy loss:

- **Internal** The internal privacy loss can only be measured by the target agent. It takes into account the true length of \(\Delta_{\tau}\): \(R_{\text{internal}} = R(\Delta_{\tau})\).
- **Maximum** The maximum privacy loss assumes that \(\Delta_{\tau}\) ends at the final timestep about which some information was gained: \(R_{\text{max}} = R(\alpha_{\text{current}} \cap \{\alpha \mid O_{\alpha} \neq O^+\})\).
- **Estimated** The estimated privacy loss assumes that \(|\Delta_{\tau}| = |\Delta_{\tau}|\), unless information was gained about a later timestep: \(R_{\text{estimated}} = R(\alpha_{\text{current}} \cap \{\alpha \mid O_{\alpha} \neq O^+\})\).
- **Theoretical** The theoretical privacy loss assumes that the plan is as long as that of the longest plan in the group of agents: \(R_{\text{theoretical}} = R(\alpha_{\text{max}} \cap \{\alpha \mid a \in A\})\). It can only be computed by an omniscient entity.

The difference between these measures lies in the estimation of the length of the target plan. Obviously, this is an important aspect, as it has a profound impact. However, the invading agent has no way to determine this exact length, and has to make an estimate. The internal privacy loss is, in some sense, the true privacy loss, as it takes into account the actual length of the plan \(\Delta_{\tau}\). The target agent can use this
measure to evaluate its own plan with respect to privacy. At the other end of the scale, we have the theoretical loss. This is the loss over the entire horizon of the planning episode. It cannot be computed by any of the agents (as none of them know the exact lengths of the other agents' plans), but it can be used by developers to assess the privacy implications of their systems. The maximum and estimated privacy losses can be used by invading agents to estimate what information they have gained. The maximum loss takes an optimistic view, and assumes that the last timestep of which information was gained is also the last step of the target plan. The estimated loss takes a more moderate view, and uses the length of the invading agent's plan instead, if it is longer. Hence, it is an estimate of the theoretical loss, and is a more accurate measure of privacy loss over the whole episode than the maximum loss. In our next section, we show some examples of the different measures.

3. EXPERIMENTAL SETUP

To illustrate the metric that we propose, we apply it to evaluate the privacy aspect of a multiagent planner. For this analysis, we have used the results for the benchmark set of the MPOP planning system [12, 13]. This set consists of multiagent versions of the logistics domain from the plan repair benchmark set of the GPOP system [3], as well as a multiagent version of the gripper domain. We focus our attention on the latter domain, as it was constructed to investigate scalability issues and hence has a wider range of problems than the former set.

3.1 The System and the Datasets

The idea behind the MPOP system is to combine a dynamic planning method for each agent with an auction for delegating (sub)tasks. The system consists of a number of agents that first concurrently plan for a single goal. As some goals may involve subgoals that the agent cannot achieve itself, each agent is equipped with some high-level information about the services of others. They can use this information to reason about which subgoals they should auction. After the first planning phase, the agents take part in an auction (if there is any) to exchange some of these unattainable subgoals. Then, they apply a plan repair technique to add another goal to their plan, and take part in the next auction. They continue to alternatingly perform these steps of adapting a plan using plan repair and taking part in an auction until a complete and valid plan is computed. When an agent gets a task assigned on which others depend, a heuristic is employed that lets the agent schedule it early in its plan to prevent cyclic dependencies.

Notice that the information exchanged is very little: the auctioneer only specifies a fact that it wants to see achieved, the bidders issue a single value as their bid, and the winner communicates when it starts working on the goal and at which point it is finished. No information is exchanged about actions that are being undertaken, as is for example done in the partial global planning approach [2]. This may lead one to assume that the system has favourable privacy properties. Indeed, this is what the authors suggest [13]: "However, it allows us to create valid multiagent plans without exchanging details about the plans, which is very important for self-interested agents."

Figure 1: The multiagent logistics domain.

Figure 2: Graphical representation of the robots domain. Depicted are the 25 rooms, as well as the areas in which each agent may travel.

The system is evaluated on two sets of benchmark problems. In [13], the authors present a multiagent variant on the well-known logistics domain. As can be seen in Figure 1, the standard problem is divided over a number of agents: one for each city, responsible for the intra-city transport of goods, and an additional agent that handles the inter-city transport. For our analysis, we have used the B benchmark set, which consists of 45 plan repair problems.3

In [12], an additional domain is introduced called robots, cf. Figure 2. This is a multiagent variant on the gripper domain, in which a moving robot has to deliver objects. In the multiagent variant, each agent is confined to a single corridor and the agents have to coordinate to bring objects between corridors. This benchmark set consists of a number of random problems generated for different problem sizes (measured by the total number of goals). For our analysis, we have averaged the results over 3 problems of each size.

3Note, however, that we did not employ a plan repair method to solve these problems. Rather we computed solutions from scratch, without taking into consideration a previously existing plan.
3.2 Evaluation

We evaluate the privacy aspect of the MPOP system by comparing it to a centralised generation of plans by the VHPOP [16] system. To do so, we compute the information that an agent can gather from the final plan that is computed. This information is present in the form of inter-agent dependencies. Such dependencies exist when one agent achieves a subgoal for another agent. Moreover, information can be gained by relaying this information. For example, in the logistics domain, if an agent is informed that particular subgoal of moving a package within a city is started at time \( i \) and finishes at time \( i + 2 \), we can not only infer that a load action takes place at time \( i \) and an unload takes place at time \( i + 2 \), but due to the workings of the domain, we can also deduce that a move action is executed at step \( i + 1 \). In this case, the uncertainty for three timesteps of the plan is removed, which is the information we gained. For one agent, this does not hold, however: the inter-city transport agent in the logistics domain controls two airplanes. Hence, for every timestep, there is uncertainty about two actions. And if we look at the dependencies, we see that still a large degree of uncertainty remains. For example, if we can infer a load action for a certain timestep, we do not know which of the two airplanes is involved. Thus, the uncertainty in this case is only slightly reduced. A more elaborate example is given in the sidebar.

For the purpose of this evaluation, we have assumed that the probabilities of each action is equal. While this is not completely accurate (move actions occur slightly more often in both domains) the analysis still gives a good feeling for the privacy loss.

Logistics

Notice that in the logistics domain, interactions only occur between the inter-city agents and the intra-city agent. We therefore look at the average privacy loss that the inter-city agent incurs with respect to each of the other agents, and the average privacy loss by the inter-city agents to the intra-city agent. Figure 3 shows the theoretical privacy loss that the inter-city agents incur with respect to the intra-city agent for the 45 different problems in this set. As we can see, the privacy loss is worse for the MPOP system when compared to the centralised result. The main reason for this is that the centralised approach combines more orders. The anti-clockwise heuristic employed by MPOP has the effect that its plans usually contain consecutive load, drive, and unload actions which is what it does for another agent. As discussed before, this leads to a decrease in uncertainty for three timesteps. Compare this to the central approach, which often has (also more efficient) sequences as load, drive, unload, drive. Depending on the order of the loads and unloads, the drive action may or may not be deduced, which on average leads to a lower loss in privacy.

Figure 4 shows a quite different picture for the inter-city agent. As we noted earlier in this section, the information that agents can gain from interacting with this agent is lower, because of the two airplanes this agent can use. As a result, the loss in privacy is very little for both methods. The reason for choosing VHPOP is that MPOP is ultimately derived from this planner.

Example

Consider the following problem in the logistics domain: There are two cities, \( A \) and \( B \), each with a post office \( p_A \) and an airport \( a_A \). Agent \( a_1 \) handles the truck in city \( A \) and has a package \( p_1 \) to transport between \( p_A \) and \( a_A \). Agent \( a_2 \) is responsible for the truck in \( B \). It has an order to transport \( p_2 \) from \( p_B \) to \( a_B \). A third agent \( a_{AB} \) owns the airplane to transport goods between the airports in cities. Notice that the transportation of \( p_2 \) requires all three agents to cooperate. For sake of simplicity, assume that the truck in \( A \) is located at the post office, the one in \( B \) is at the airport, and the airplane is in city \( A \).

Central case

The central planner could produce the following plan (each action is preceded by its timestep and the executing agent in parentheses):

1. \((a_A)\): load \( p_1 \) 7. \((a_{AB})\): fly from \( A \) to \( B \)
2. \((a_A)\): load \( p_2 \) 8. \((a_{AB})\): unload \( p_2 \)
3. \((a_A)\): drive \( p_A \) to \( a_A \) 9. \((a_B)\): load \( p_2 \)
4. \((a_A)\): unload \( p_1 \) 10. \((a_B)\): drive \( a_B \) to \( p_B \)
5. \((a_A)\): unload \( p_2 \) 11. \((a_B)\): unload \( p_2 \)
6. \((a_{AB})\): load \( p_2 \)

Agent \( A \) is informed of its plan (i.e., the final 3 actions), as well as the fact that \( a_1 \) undertakes its part of the task between timesteps 2 and 5, and \( a_{AB} \) between time steps 6 and 8. From this, \( a_2 \) can infer the load and unload actions in the plan of \( a_1 \). The minimal privacy loss is computed over the first 5 steps, as step 5 is the latest time step of which \( a_B \) learns information. The minimal privacy loss of \( a_1 \) with regard to \( a_{AB} \) is therefore \(-5 \log_2(\frac{1}{8}) + \frac{3}{8} \log_2(\frac{1}{4})\), or 40% (\(-5 \log_2(\frac{1}{8}) + \frac{3}{8} \log_2(\frac{1}{4})\)). The estimated loss of \( a_{AB} \) with respect to \( a_B \) is \(-11 \log_2(\frac{1}{8}) + \frac{3}{8} \log_2(\frac{1}{4}) + 8 \log_2(\frac{1}{3})\) or 18%. For 3 steps (the load, fly and unload actions), we now know the type of action, but not which plane. This reduces the uncertainty for these three steps from 8 possible actions (4 for each plane), to just two.

Distributed case

MPOP produces the following plan:

1. \((a_A)\): load \( p_2 \) 8. \((a_{AB})\): load \( p_2 \)
2. \((a_A)\): drive \( p_A \) to \( a_A \) 9. \((a_B)\): fly from \( A \) to \( B \)
3. \((a_A)\): unload \( p_2 \) 10. \((a_{AB})\): unload \( p_2 \)
4. \((a_A)\): drive \( a_A \) to \( p_B \) 11. \((a_B)\): load \( p_2 \)
5. \((a_A)\): load \( p_1 \) 12. \((a_B)\): drive \( a_B \) to \( p_B \)
6. \((a_A)\): drive \( p_A \) to \( a_A \) 13. \((a_B)\): unload \( p_2 \)
7. \((a_A)\): unload \( p_1 \)

The privacy losses now are as follows. Agent \( a_1 \) now knows 3 out of 3 actions of agent \( a_A \) when computing the minimal privacy loss (it can infer the move action from the time steps of the load and unload actions): \(-3 \log_2(\frac{1}{8}) + 0\), or 100%. The estimated loss of \( a_{AB} \) is slightly less in this scenario, as the length of the plan has increased to 13: \(-13 \log_2(\frac{1}{8}) + (3 \log_2(\frac{1}{4}) + 10 \log_2(\frac{1}{3})\)), or 15%.
Figure 3: Average (theoretical) privacy loss of the intra-city (trucking) agents

Figure 6: Average estimated privacy loss agents 1-4

Figure 4: Average (theoretical) privacy loss of the inter-city (flying) agent

Figure 7: Average theoretical privacy loss agents 1-4

Figure 5: Average internal privacy loss agents 1-4

Figure 8: Average estimated privacy loss of agent 5
Here, too, MOPPR fares worse than the centralised method on most problems.

**Robots**

In the robots domain, we also have a central agent that interacts with all other agents. The corridors all intersect with the central corridor, and with no others. Hence, we look at the agent controlling the central corridor and the average of the other agents. However, we not only look at the theoretical privacy loss, we shall also look into the internal privacy loss and the estimated loss. For this domain, the graphs display the average values obtained over 3 different problems instances of a given size (in number of goals to be achieved)

The average loss that the agents incur from interacting with the agent in the central corridor is shown in Figure 5 (internal privacy loss), Figure 6 (estimated loss) and Figure 7 (theoretical loss). Again, we observe that MOPPR often gives a greater loss in privacy than the centralised method. Interestingly, if we look at the average theoretical privacy loss (and to a lesser extent the estimated loss), this seems to remain relatively stable. We conjecture that this is due to the fact that while the plans do get bigger (as there are more goals to achieve), there are also more opportunities for learning about the other agent’s actions. Together these two effects counter each other, leading to a stable behaviour we see. This behaviour is not noticeable in the internal privacy loss. Often one of the agents only has one or two packages to handle. If this involves a coordination with the central agent, almost all of the plan is exposed. However, since the central agent does not know the lengths of the plans, the estimate does not reflect this. The observer simply doesn’t know that it has learned so much information.

Figures 8 and 9 show the privacy loss for the central agent. As one can see, the privacy loss for this agent is comparable to that of the other agents, although slightly lower. Since the agent interacts with many other agents they gain less information about its plan then *vice versa*. This is a matter of its plan being bigger on average.

### 3.3 Preliminary Conclusions

While the aim of this section is mainly to illustrate the metrics that we propose, we can draw some initial conclusions from the results. First of all, it underlines the necessity for a metric for privacy loss. A system that was designed to leak little information, turns out to be actually worse when it comes to privacy than a system in which a plan is computed centrally and then distributed. Even though we singled out the MOPPR planning system, other multiagent systems have neither underlined a privacy analysis. Rather, they rely on enforcing certain properties that are thought to provide privacy for their users. Unfortunately, good intentions do not make for a secure system, as we have seen from this analysis.

A recommendation that can be drawn from these experiments for a privacy aware planner would be to allow for a certain degree of slack and randomisation. Another option would be the interleaving of actions. The MOPPR system enforces neither of these aspects, whereas the central planning has many tasks interleaved, which improves privacy. However, in light of the previous paragraph, we cannot simply assume that adhering to these properties would make for a secure system. A privacy analysis remains indispensable.

### 4. RELATED WORK

Recently, researchers in the field of Distributed Constraint Optimisation (DCOP) and Distributed Constraint Satisfaction (DisCSPs) have started to propose metrics for analysis of privacy loss in such systems. Most of the work in this area focuses on distributed meeting scheduling. In this type of problems, a number of agents has to schedule a number of meetings. Each meeting requires a certain set of agents to be present, and each agent has preferences or costs attached to timeslots and locations. Silaghi and Faltings [9] use a measure of privacy to drive their algorithm. Each agent has certain costs associated with the revelation of whether some tuple of values is feasible. During the exchange of messages, agents have to pay this cost if some tuple is fully determined by the other agents. Negotiations are terminated if the cost of revealing a certain tuple is greater than the potential reward for collaborating. A similar model is the basis of Silaghi and Mitra [10]. However, the privacy metric here is the size of the smallest coalition necessary to deduce an agent’s costs for certain tuples. Wallace and Freuder [15] consider a measure of privacy loss that is very close to ours. Their work, like ours, is based on information entropy. However, the application of information theory is more straightforward as they consider the uncertainty of each of the variables in the constraint satisfaction problem, rather than having to apply it to an additional datastructure (i.e. the plan) as we do. Recent work by Maheswaran et al. [4] proposes a general quantitative framework to analyse privacy loss. The three earlier approaches can be seen as specific instances of this framework.

Beyond DCOP and DisCSP, research on privacy is undertaken in the agent community. This includes work on cryptographic techniques, secure auctions and randomisation (see e.g. work by Brandt [1], Van Otterloo [14] and Naor [5]). Of particular interest to planning is the work on randomisation (e.g. Paruchi et al. [6] and Van Otterloo [14]). These approaches assume that actions and behaviours can be observed. By choosing actions in a randomised fashion (e.g. using policies with a high entropy) agents can try to provide minimal information on their preferences, while still attaining their goals.
5. DISCUSSION AND EXTENSIONS

Although privacy is an issue that is often mentioned in work on multiagent planning (as well as multiagent approaches to different problems), heretofore this notion was neither made explicit, nor analysed. The present work shows how Shannon’s Information Theory can be applied to (classical) planning to derive meaningful definitions of concepts such as uncertainty, information and privacy loss. As an illustration of our work, we applied it to an existing multiagent planning system to evaluate its performance with respect to privacy, compared to a central solution with a trusted third party. This clearly established the need for such an analysis: even though the multiagent planner seemingly exchanged little information, its overall performance with respect to privacy was less than that of the centralised system.

A number of extensions to this work seem obvious. Firstly, it seems reasonable to consider past actions when it comes to establishing the uncertainty of a certain timepoint. For example, in the logistics domain, we often see the following sequence of actions: load (a package into a truck), move (the truck to the package’s destination) and unload (the package from the truck). Hence, if we knew that at time $t$ a load action takes place, the probability (or estimated frequency) of a move action at time $t+1$ is slightly higher. Work on the entropy of Markov processes can be of help here. For example, for a first-order Markov source $S$ (where the probability of a symbol is dependent only upon the immediately preceding one), the entropy rate is

$$H(S) = - \sum_i P_i \sum_j P_i(j) \log_2 P_i(j)$$

where $P_i(j)$ is the probability of $j$ given that $i$ was the preceding symbol. Such an extension would bring our work closer to the model of planning in conditional and conformant planning approaches.

Another extension is to consider more than just the type of action. Our model is limited in the sense that we do not distinguish between knowing that a move action takes place, and knowing the precise locations. A step in this direction was made in the logistics domain, where we allowed uncertainty to exist with regard to which plane was used by the mercy transport agent. However, there may be domains where such a distinction is vital and requires a more rigorous assessment then our current model allows for.

Thirdly, our work is based on the classical definition of plans. Over the years, more advanced definitions have emerged, allowing for parallel execution of actions and durative actions. Obviously, to assess the privacy issues of modern planners built on these richer formalisms, the definitions of uncertainty, information and privacy loss will have to be extended. It is not immediately clear how our work can be generalised to these situations.

Finally, this work allows new multiagent planning systems to be designed. Whereas before, privacy was mentioned as a driving force but not explicitly taken into account, the existence of a metric for privacy loss can direct research to new algorithms that are optimized for privacy. This would also entail research into the relation between privacy loss and other factors such as optimality and search efficiency.

6. REFERENCES


Task Scheduling using Constraint Optimization with Uncertainty

James Atlas
Computer and Information Sciences
University of Delaware
Newark, DE 19716
atlas@cis.udel.edu

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

1. INTRODUCTION

Multiagent task scheduling encompasses diverse domains of problems that require complex models and robust solutions. C-T/EMS [1] is a new specification, based on T/EMS [2], for multiagent task scheduling problems that represents the complex relationships necessary to model these domains.

Some recent work has been done in the area of mapping the C-T/EMS scheduling problem into a Distributed Constraint Optimization Problem (DCOP) [6]. Distributed constraint optimization is a direct extension to the traditional AI approach of constraint satisfaction for multi-valued constraints in a distributed system [7, 3]. Typical DCOP algorithms define the optimal solution as the optimal sum of local utilities.

Currently the mapping from C-T/EMS to a DCOP allows only for certain combinations of quality accumulation functions (QAFs), and works only for deterministic outcomes. The C-T/EMS scheduling problem contains uncertain information describing possible outcome distributions over the qualities of methods. The combination of these possible outcomes distributions creates uncertainty in the global utility of a task schedule. Using an evaluation function for comparisons, the optimal schedule may not be equal to the one with the optimal sum of local utilities.

In this paper we extend the original DCOP formalization for uncertainty information in the form of utility distributions. Additionally, we extend the C-T/EMS mapping to include additional QAF combinations using only binary constraints. We then show how the C-T/EMS mappings can take advantage of the extended DCOP formalization with some sample evaluation functions. This research is ongoing, and comprehensive results on general classes of C-T/EMS scheduling problems are pending.

2. C-T/EMS SCHEDULING PROBLEM

The multiagent task planning and scheduling problem requires a rich language for domain representation. The original T/EMS (Task Analysis, Environment Modeling, and Simulation) language was developed to provide a domain independent, quantitative representation of the complex coordination problem [2]. A C-T/EMS problem instance contains a set of agents and a hierarchically decomposed task structure. Nodes in the graph are either complex tasks (internal nodes) or primitive methods (leaf nodes). Each node may have temporal constraints on the earliest start time and the deadline. Nodes may also have non-local effect (NLE) constraints that represent hard and soft precedence. Methods have probabilistic outcomes for duration, quality, and cost. Tasks have a quality accumulation function (QAF) that describes the quantitative combination of quality outcomes of subtasks and methods. Some basic QAFs include sum, min, max, and sync.sum. In the sample C-T/EMS problem instance in Figure 1, the node T1 represents a task with M1 and M2 representing a decomposition of this task into submethods. T1 has constraints for earliest start time of 1 and deadline of 31. The accumulated quality at T1 is a sum of the qualities of the executed submethods. In this case if both M1 and M2 executed within the temporal constraints, T1 would have an accumulated quality of 5 + 10 = 15.

3. DCOP FORMALIZATION

DCOP has been formalized in slightly different ways in recent literature, so we will adopt the definition as presented in [5]. A Distributed Constraint Optimization Problem with n nodes and m constraints consists of the tuple \(<X,D,U>\) where:

- \(X = \{x_1, ..., x_n\}\) is a set of variables, each one assigned to a unique agent
- \(D = \{d_1, ..., d_n\}\) is a set of finite domains for each variable
- \(U = \{u_1, ..., u_m\}\) is a set of utility functions such that each function involves a subset of variables in \(X\) and

Figure 1: An example C-T/EMS problem instance.
defines a utility for each combination of values among these variables.

An optimal solution to a DCOP instance consists of an assignment of values in $D$ to $X$ such that the sum of utilities in $U$ is maximal. Problem domains that require minimum cost instead of maximum utility can map costs into negative utilities. The utility functions represent soft constraints but can also represent hard constraints by using arbitrarily large negative values.

4. EXISTING MAPPINGS FOR C-TÆMS SCHEDULING

A mapping for a subset of C-TÆMS to DCOP is proposed in [6]. The mapping using our formalization is:

- $X =$ Each method is assigned to a unique variable.
- $D =$ Unique domains for each variable containing all possible start times for the method assigned to the variable.
- $U =$ Three types of utility functions:
  - Mutex constraints on all pairs of methods that share the same agent
  - For an NLE between two nodes, $N_1$ and $N_2$, all methods in the subtree of $N_1$ have a precedent constraint with all methods in the subtree of $N_2$
  - Unary soft constraints on each method that apply a cost if the method is not scheduled

This mapping allows only specific QAFs, enables NLE, and deterministic task outcomes. The complexity of the mapping, where $M$ is the number of methods in the original C-TÆMS problem, involves $O(M)$ variables, $O(M^2)$ utility functions, and the size of each domain is $O(|T|)$ where $T$ is the range of all possible start times.

5. PROPOSED MAPPINGS FOR C-TÆMS SCHEDULING

We can observe that DCOPs naturally optimize global sums of utility, so the mapping of the sum QAF can be achieved with relatively simple binary constraints. It is possible to map min and max QAFs using the existing mapping if they are the only QAF in a specified instance [6]. This is achieved by changing the DCOP aggregation function to a min or max function. The authors in [6] used ADAPT [4] to verify these mappings.

To date it has proven very difficult to combine QAFs in a hierarchical fashion. Although all types can be mapped using binary constraints over entire subtrees of tasks/methods, that increases the computational complexity. We propose the following mappings to binary constraints for non-sum QAFs that are correct for problem instances where non-sum QAFs apply only to methods and not to tasks or task groups.

5.1 sync_sum

This QAF produces quality equal to the sum of the methods that start at the same time slot. To start, we create a new variable that represents the synchronized start time of the methods involved in the sync_sum. This variable has the domain of all possible start times for any of the methods. Instead of the unary constraint producing a cost if the method is not scheduled, we create binary constraints between every method and the special sync_sum variable. If the method is not scheduled for the same time slot as the special sync_sum variable, the constraint returns a cost equal to the quality of that method.

5.2 min

This QAF produces quality equal to the minimum quality of any submethod. An important note is that the min QAF produces no quality if any of the methods are not scheduled. To map this QAF, we again create a special variable. This variable has the domain of true and false. We first determine which of the submethods has the lowest potential quality. A special binary constraint is created between this method, $a$, and the special variable, $v$. The cost function for this constraint is:

$$\text{cost}(a) = \begin{cases} a_{\text{qual}} & \text{if } v = \text{false} \\ \infty & \text{if } v = \text{true} \text{ and } a \text{ not scheduled} \\ 0 & \text{otherwise} \end{cases}$$

Next, we create a binary constraint between each other method, $b$, and the special variable, $v$, with the cost function:

$$\text{cost}(b) = \begin{cases} \infty & \text{if } v = \text{true} \text{ and } b \text{ not scheduled} \\ 0 & \text{otherwise} \end{cases}$$

5.3 max

This QAF produces quality equal to the maximum quality of any submethod. To map this QAF, we create another special variable. This variable's domain values are the set of all possible qualities produced by any submethod (and a special tag that marks that quality's submethod) and not scheduled value. We first determine which of the submethods has the highest potential quality, $q_{\text{max}}$. We create a binary constraint between each of the methods, $c$ and the special variable, $x$. If the method, $c$, is scheduled we use the cost function:

$$\text{cost}(c) = \begin{cases} \infty & \text{if } c_{\text{qual}} > x \\ q_{\text{max}} - c_{\text{qual}} & \text{if } c_{\text{qual}} = x \text{ and } x_{\text{tag}} = c \\ 0 & \text{otherwise} \end{cases}$$

If the method, $c$, is not scheduled we use the cost function:

$$\text{cost}(c) = \begin{cases} 0 & \text{if } x_{\text{tag}} \neq c \\ q_{\text{max}} & \text{if } x \text{ not scheduled and } q_{\text{max}} = c_{\text{qual}} \\ \infty & \text{otherwise} \end{cases}$$

6. UNCERTAINTY IN THE C-TÆMS SCHEDULING PROBLEM

Uncertainty of various task characteristics, such as completion time, solution quality, and total cost, is one of the major complexities underlying the C-TÆMS scheduling problem. In our previous discussion of mappings, we do not attempt to include this concept. All the mappings so far discussed are based on deterministic outcomes.

Uncertainty can be represented in an agent system as a statistical distribution of values. We may incorporate a crude mapping of this uncertainty into the existing DCOP...
framework by using average expected value for the uncertainty (a sum product of the probabilities and values). However, this offers little help in domains where the global utility over the uncertainty is not an average expected value. For many domains including C-TEMS scheduling, it is valuable to incorporate risk aversion functions into the global utility, or enforce minimum confidence level utility.

6.1 DCOP with Utility Distributions

We can extend the DCOP problem formulation to include uncertainty by allowing constraint evaluation functions to return a distribution instead of a single value. A global optimum is now an optimal distribution instead of a maximum (or minimum) sum. To evaluate the optimality of a distribution, evaluation criteria must be formalized. The optimal evaluation function may not be the same for all problems for all agents. Thus we must include the evaluation function as part of the extended DCOP problem. We extend our previous DCOP formalization for this:

- \( U = \{ u_1, \ldots, u_n \} \) is a set of utility functions such that each function involves a subset of variables in \( X \) and defines a utility distribution for each combination of values among these variables
- \( \text{e} = \{ (u_1, u_1), \ldots, (u_n, u_n) \} \) is a distribution of probabilities and values such that \( \sum_{r=1}^{n} u_r = 1 \)
- \( E = \{ e_1, \ldots, e_n \} \) is a set of evaluation functions for each variable that reduce a utility distribution to a single utility value \( e(u) = v \) where \( v \) is a single utility value.

6.2 Re-mapping C-TEMS to DCOP

The C-TEMS scheduling problem mapping presented in the previous section can easily be extended to include this concept of uncertainty. We maintain the current mappings for all of the hard constraints (ones that return either zero or infinite cost). For each soft constraint we return a set of utility distributions instead of the single value. This allows a method to specify that it produces a distribution of quality; for example in Figure 1 method M1 may now produce quality = 5 for 80% of executions, quality = 10 for 10% of executions, and quality = 100 for 10% of executions. This would be represented as a utility distribution of \( \{ (0.8, 5), (0.1, 10), (0.1, 100) \} \). Also, for the C-TEMS scheduling problem we use a single evaluation function, \( E \), for all agents.

We illustrate three sample evaluation functions among many that express the effectiveness of our model. The first function is a risk neutral expected value function that simply computes the product of the utility distribution. For the prior example distribution, this function would evaluate as:

\[
e(u) = 0.8 \cdot 5 + 0.1 \cdot 10 + 0.1 \cdot 100 = 15
\]

The second is a risk averse evaluation function based on quadratically decreasing utility, where:

\[
e(u) = \left( \sum_{r=1}^{n} u_r \cdot \sqrt{u_r} \right)^2
\]

For the example this would evaluate as:

\[
e(u) = (0.8 \cdot \sqrt{5} + 0.1 \cdot \sqrt{10} + 0.1 \cdot \sqrt{100})^2 = 9.64
\]

The third is a minimum confidence evaluation function, such that utility equals the highest value \( v \) such that \( c \% \) of the values are greater than or equal to \( v \). For the example this would evaluate as 100 for \( c = 5 \), 10 for \( c = 20 \), and 5 for \( c = 90 \). Many other evaluation functions are available, but these simple functions will allow our global solution to incorporate things such as risk aversion and minimum confidence.

It is straightforward to aggregate utility distributions using the typical summation function. Thus a global aggregated distribution for a specific variable assignment is possible. Applying the evaluation function to the aggregated distribution allows for a choice of an optimal variable assignment. However, many DCOP algorithms also require intermediate comparisons between the values, which are distributions in our model. Appropriate comparison mechanisms that correctly identify the optimal global assignment are a topic of current research. This research and implementation is ongoing, and comprehensive results on general classes of C-TEMS scheduling problems are pending.

7. CONCLUSION AND FUTURE WORK

We have introduced a new formalization for DCOP that includes uncertainty characteristics. We showed how a problem domain that includes uncertain outcomes, the C-TEMS scheduling problem, can be mapped into our new formalization. We also introduced some additional mappings for C-TEMS QAFs to binary constraints. Using our new formalization we illustrated how evaluation functions can express concepts such as global risk aversion and minimum confidence.

We are currently testing our formalization in several domains to see how easily it can be applied to various problems that involve uncertainty. Additionally we plan to develop comparison tests to calculate the effectiveness of the model. We will also continue to improve the mapping of the C-TEMS scheduling problem to the DCOP formalization.

8. REFERENCES

An Integrated Multi-Agent Coordination Including Planning, Scheduling, and Task Execution

Dr. Wei Chen and Dr. Renato Levy
Intelligent Automation Inc.
15400 Calhoun Dr, Suite 400
Rockville, MD 20855, USA
wchen,levy@i-a-i.com

Dr. Keith S. Decker
Dept. of Computer Information Sci.
University of Delaware
Newark, DE 19716, USA
decher@cis.udel.edu

ABSTRACT
Traditional planning has been explored in depth. Most research is concentrated at a high level, e.g., developing coordination interaction protocols to be imposed on agents. There has been less concern about how the internal task structures of individual agents affect these higher-level coordination behaviors. When involving multiple participants, collaborative planning has been facing problems like uncertainty, contingencies, and evaluating worth-oriented goals. In particular, agent planning, scheduling and execution are inextricably linked to coordination behaviors. This paper presents extensions and restrictions, as extended hierarchical task networks (EHTN), to the expressiveness of traditional plan and schedule representations that allow the formal definition of the multi-agent coordination problem. Based on EHTNs, this paper discusses some open issues in multi-agent coordination (e.g. what to coordinate among agents, how much information to be exchanged, how to evaluate a planning approach) and proposes a generalized solution towards successful distributed goal achievement by analyzing the task performance of participating agents.

1. INTRODUCTION
Much work in multi-agent systems focuses on coordinating the activities of agents so that the end result approximates the solutions possible if one were to centralize the activities being carried out by these agents. Research into coordination has taken different forms: negotiation, scheduling, planning, organizational approaches, etc. Multiagent coordination, defined as managing inter-dependencies between activities [16], addresses the special issues arising from the interdependency relationships between multiple agents' tasks. We define an interdependency as a relationship between a local and non-local task where the execution of one changes some performance-related characteristics associated with the other. This definition gives rise to the following questions: (1) how are the dependencies represented, and (2) how to manage these dependencies? Previous approaches, including GGP (Generalized Partial Global Planning) [8], have concentrated on high level solutions: to keep semi-consistency between agents' activities, represent the relationships between actions and goals, and develop coordination interaction protocols. There has been less concern about how an individual agent's internal task structures and reasoning capabilities affect its coordination behaviors. In particular, in order to understand the underlying assumptions, capabilities, and limitations of our GGP approach, we need to formalize these linkages in addition to the earlier approach described in [6].

Based on the definition and the (later) analysis of interdependency, the proposed answer to the question “which applications require decentralized planning” is that it is NOT about how many agents present in a problem solving process, but whether interdependencies exist among the agents’ activities, i.e. interdependencies require decentralized planning, while the number of participating agents does not necessarily demand decentralized planning.

This paper focuses first on the representation of dependencies using extended hierarchical task networks (EHTN)[7] and the restrictions we impose on them in practice to keep the planning problem decidable. This extension induces a second problem, one of resource scheduling. Previously, planning and scheduling were often studied separately from coordination. For example, papers might analyze the use of certain market mechanisms to coordinate the use of certain resources. Such mechanisms will have particular, sometimes guaranteed performance characteristics. However, most of this high level mechanism design work ignores the point of view of the individual agent wondering which market to participate in (planning) or how to set pricing expectations (related to scheduling of the agent’s local resources) or indeed whether to participate in this mechanism at all (unless simply forced to by the programmer) or how to react to dynamic situations (re-planning due to changed environmental variables or updated execution results). This paper explains how planning, scheduling and execution are integrated to help agents improve their coordination behaviors.

Traditional HTNs, e.g. [9], are not expressive enough to represent worth-oriented goals, contingencies, or the uncertainties that arise when such plans are in fact distributed over multiple agents. 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. Collaborative planning [12] is proposed to coordinate multiple agents’ activities by exploiting a revised and expanded version of SharedPlans to handle cases in which a single agent has only partial knowledge; but the task representation is still not expressive enough to analyze arbitrary coordination mechanisms. The modular structured approach [17] uses schematic Bayes net fragments and programming language constructs to represent actions, domain relations and plan space; although it provides some quantitative representation and time factors for the task structures, the characteristics of the tasks still remain vague. It has been stated that by annotating plan choices with conditions, a sophisticated coordination process may become more efficient [13]. The modern approach to general reasoning about worth-oriented goals, contingencies, and uncertainties is the POMDP approach [3]; but even Boutilier states that in practice the standard AI representations and algorithms survive because real “problems commonly possess structure … [therefore] specialized representations, and algorithms employing these representations, can achieve computational lever-
age by exploiting these various forms of structure.” In another approach [15], execution resource constraints are defined to remove uncertainties about the plans of other agents and communicated among agents based on a convergence protocol; since the architecture is based on STRIPS, hierarchical structure information is lost; often coordination only requires commitments at an intermediate level of abstraction.

In this paper we present a brief introduction of formalized EHTNs to represent agent task features by annotating them like T.AEMS (Task Analysis, Environment Modeling, and Simulation) [8], an abstract modeling representation that represents interdependencies quantitatively, as shown in Section 2. Given EHTNs for representing tasks and environments, the remaining question is how to reason about the interdependencies between tasks. We show that these EHTNs generate a resource scheduling problem and recast our GPGP coordination mechanisms [7, 8] within this formalism in Section 3. At the end of this section we introduce the schedule coordination problem and a corresponding solution. Implementation of previous research is sketched in Section 6. Finally, discussions will be presented for the following concerns: the implementation of the integrated coordination problem covering entire reasoning and execution procedures, a performance-based solution for evaluating planning approaches, and challenges posed by human involvement in Section 7.

2. EXTENDING HTNS TO REPRESENT COORDINATION PROBLEMS

Following the definition of coordination [16], interdependencies among multiple agents introduce the problems of contingency, uncertainty and worth-oriented goal tradesoffs. We concentrate on the agent behaviors of plan/subplan selection and action scheduling. We assume that if there is perfect information, the involved agents would behave in a cooperative way. A local scheduler cannot make appropriate schedules with unknown information about an agent’s possible tasks. Such information includes how the execution of some action or subplan at a particular time will effect the agent’s performance, and what other non-local actions or subplans will change these effects. Thus a more expressive representation for agent task features is needed than classic precondition and effect information. We use T.AEMS [8] as the source of this additional expressiveness, as it provides a way of precisely representing just such local and non-local effects on agent performance.

Classical planning was oriented toward a notion of a plan as a (partially ordered) sequence of primitive actions. Williamson et al. [20] review the raft of more recent planning work and the push towards supporting a more sophisticated model of control flow, e.g., parallel execution, conditional branching, contingencies based on information gathering actions, etc. Thus we can think of classical plans with explicit control flow as a specialized case of contingency-filled plans based on information flows (and these, then, begin to look like a useful, tractable simplification of POMDP policies, as observed by Bottouler [3]). Similar to the way the explicit control flow in a classical plan is dictated by the plan step precedence sets of provisions attached to an action indicate where it is enabled for execution. Another important feature of the traditional HTN representation was that (unlike [9]) a reduced task is not replaced with its subtasks, but instead the task network is given a tree-like structure. In the multi-agent context, this is important both for providing a way to send partial, abstract plans to other agents that omit some details, and to allow the specification of explicit task relationships such as enablement in complex cases.

There are two main features that T.AEMS has that traditional HTNs do not have. First, a quantitative definition of a vector of measurable, utility-influencing characteristics, such as quality, cost, and task duration, and how these characteristics accumulate as actions are executed, called characteristic accumulation functions (CAFs). Secondly, explicit task relationships indicating how task progress affects primitive action characteristics elsewhere in the task structure. These extra features both move the planning problem from a goal-achievement view to a worth-oriented view, and more importantly provide a basic unit over which all coordination reasoning, no matter how complex, is ultimately concerned with (the task relationship).

We will base our formal definition of the coordination problem on the well known work of Erol, et al. [9], and focus primarily on the differences. Note that we are only concentrating on an agent’s (partially) local view—primitive or compound tasks at other agents are represented locally as non-local tasks (NLTs). The vocabulary $\mathcal{L}$ is a tuple $\langle V, C, P, F, T, N, NLT, I, O, A, CAF \rangle$, where $V$, $C$, $P$, $F$, $T$, $N$, $NLT$, $I$, $O$, and $A$, and $CAF$ are the $i^{th}$ attribute of a task/action (e.g., quality, duration, cost, completeness, reputation, etc.), but also in some work characteristics like duration-mean, duration-standard-deviation, etc.). An instance of attribute $A_i$ is shown as $a_i$. For example, the duration of a searching task at some certain search engine is 1 minute and the cost is $100$. $CAF$ is a set of characteristic accumulation functions; 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, qsum, qsumall as presented in [11, 18]. So the cost of a task is the sum of the costs of its children; the duration of a task is the max duration in parallel execution and sum in sequential execution; tasks who only need one child completed ("OK" tasks) have max as quality-CAF, etc.

We modify Erol’s representation to include explicitly named input provisions and outcomes (representing possible contingencies). A Primitive Task (informally, an action), is a syntactic construct of the form $\textit{do}(f(\pi_1, ..., \pi_k) \rightarrow (\omega_1, ..., \omega_l))$, where $f \in F$ is a primitive task symbol, the $\pi_i$ are unique provision symbols in $I$, and $\omega_i$ are unique outcome symbols in $O$. The function $A : A \rightarrow \mathcal{R}$, is a mapping from an attribute to the default value for that attribute. In some work, $A$ is specified at the operator level (different for each outcome/contingency) [9]. A Non-Local Task, is a syntactic construct of the form $\textit{NLTDo}(\textit{nlt}(\pi_1, ..., \pi_k) \rightarrow (\omega_1, ..., \omega_l), A)$, where $nlt \in NLT$ is a compound task symbol, and the other symbols as before, except that $A$ may be partial or missing—this is one of the prime uncertainties associated with non-local tasks. A Goal Task is a syntactic construct of the form $\textit{achieve}(l, \phi)$, where $l$ is a literal and $\phi$ is a boolean formula constructed from characteristic vector constraints such as $a_i < c$. An instance of this boolean formula could be $(a_{\textit{deadline}} < \textit{1:00 PM}) \wedge (a_{\textit{cost}} \leq 850)$. A Compound Task, is a syntactic construct of the form $\textit{perform}(l(\pi_1, ..., \pi_k) \rightarrow (\omega_1, ..., \omega_l), A)$, where $l \in T$ is a compound task symbol, $\pi$ and $\omega$ are unique provision and outcome symbols. The $\delta : A \rightarrow \textit{CAF}$ maps attributes to the CAFs that determine the value of the attribute from the attributes of the child tasks of the compound task.

Task networks are modified to explicitly include links from out-
comes to provisions, and unlike some HTN formalisms the hierarchy from compound tasks to the associated task network is kept intact. This is extremely useful in multi-agent situations for creating abstractions of a task, or for making commitments to achieve a goal or perform compound task without committing to a particular method. A Task Network, \( T \), is a syntactic construct of the form
\[
[(\alpha: n_1), (\alpha: n_2), ..., (\alpha: n_m), \psi, L_{prot}, L_{inherit}, L_{disinherit}],
\]
where each task in the net has a unique label \( n \) and \( \alpha \) is root task of \( T \). \( \psi \) is a boolean formula containing variable binding constraints, temporal ordering constraints, and truth constraints (such as the “protection intervals” that appear in POP-style planners). The \( L \) sets are sets of explicit links between provisions and outcomes as in Figure 1. \((n_i, \omega_j, n_j, \pi_j) \in L_{prot} \) link outcomes to input provisions of network siblings. \((n_i, \pi_k, n_i, \pi_i) \in L_{inherit} \) link the provisions of the parent non-primitive task to the provisions of its subtasks. \((n_j, \omega_i, n_i, \omega_i) \in L_{disinherit} \) link the outcomes of the subtasks to the outcomes of the parent non-primitive task.

Erol’s operator definition is modified to allow specification of any contingency separately. An operator is of the form
\[
\text{operator}(f(v_1, ..., v_k, \omega_1, ..., \omega_m))
\]
where the literals denote the primitive task’s effects under the given outcome, or perform\((n|f(v_1, ..., v_k, \omega_1, ..., \omega_m))\) where the literals denote the non-local task’s effects under the given outcome. A method \((\alpha, d)\) still maps from a goal or compound task \( \alpha \) to a task network \( d \), representing one possible task reduction. A planning domain remains as a pair \( D = (Op, Me) \) where \( Op \) is a set of operators, and \( Me \) a set of methods, and a planning problem is still a triple \( P = (D, S_i, D) \), where \( D \) is a planning domain, \( S_i \) an initial state, and \( d \) is the task network we need to plan for (usually a goal task). Finally we slightly modify the definition of a primitive task network; it will contain only primitive tasks, non-local tasks, or non-primitive tasks that have been reduced to primitive or non-local tasks. All provisions must be linked to, and all outcomes have links from them. Based on the definition of expressiveness by Baudrand [11] and the similar ideas in Erol et. al. [9], we present the expressiveness theorem as follows:

**Theorem 1.** \( \mathcal{L} \) is strictly more expressive than the traditional HTNs. The expressive power of the extended HTNs comes from the additional representation of information flow and control flow: the explicit definition of non-local tasks, input/output provisions and the computing ability for task characteristics. Without these new language constructs, \( \mathcal{L} \) reduces to the equal expressiveness of traditional HTNs. The straightforward proof is omitted here.

3. **GPGP MECHANISMS AND THE SCHEDULE COORDINATION PROBLEM**

![Figure 2: Relationships among the planner, the scheduler and the coordination module.](image)

With the extended HTN representation, one can describe the management of the interdependencies among agents’ activities using a set of GPGP coordination mechanisms. The key problem is that there exists uncertainty in the attribute vector values \( A \) of non-local tasks, which can lead to bad schedules, low efficiency, and resource misuse. We state a base assumption first: Each agent is capable of reasoning locally about its schedule of activities and possible alternatives. This assumption guarantees an appropriate local scheduling without uncertainty within task executions. A schedule \( S \) is a list of primitive tasks along with an earliest start time (EST) and latest finish time for each, indicating a specific execution path from a primitive task network \( T \).

The scheduling problem is to select, from a set of possible execution paths, the “best” execution path[11]. Two kinds of information are needed: (1) an estimate of the cumulative performance of the paths, and (2) a utility function. Recall that a vector of Characteristic Accumulation Functions (CAF), \( \delta \), is estimated characteristically of compound tasks, given scheduling information about primitive tasks. Recall that any task network \( T \) has a single named root, and that the attributes associated with that root node are defined purely recursively in the attributes of its subtasks. Thus for a primitive task network \( T \) that includes no non-local tasks we can simply compute the attributes associated with \( T \) given a schedule \( S \) that says what primitive tasks are executed and when. A utility function, \( U \), is the reference from the user about the desirability of some set of attributes on a final plan and schedule, i.e. \( U(S, T) \rightarrow \mathcal{R} \). Thus, a local scheduler is defined as a function: \( \text{Schedule}(T, U) = S \), that attempts to maximize \( U(S, T) \). Solutions to this problem include [10, 11, 19].

Besides the local scheduling problem there is a schedule coordination problem to provide information to the local scheduler, removing uncertainty and thus allowing construction of better schedules. The general idea is to find a way to fill the uncertain information in the uncoordinated task structures. If \( A_i(T, S, n) \) is the value of attribute \( i \) in task network \( T \) and schedule \( S \) of named task \( n \), then how to determine these values in the case of plans that contain non-local tasks? An example is like this: a subtask, SubTask1, in task network, \( T_A \), of Agent A is enabled by a provision from a remote task in another agent and SubTask1 is uncertain about the duration of the enabling task. Our solution is that a specific GPGP coordination mechanism is applied and a result sent from the remote agent that the enabling task will take 10 seconds to finish; a local scheduler for Agent A may now calculate the cumulative duration of SubTask1 based on the answer from the remote agent and the other local branches with known estimated durations.

Figure 2 shows the relationship and the information flow among the planner, the scheduler and the coordination module in our agents’ internal structure: The planner provides uncoordinated plans (with uncertainty) to the coordination module; the coordination module takes the uncoordinated plans as input, applies one or a combination of appropriate GPGP coordination mechanisms to the uncoordinated plans and produces coordinated plans, which are the input
to the scheduler; the scheduler uses the coordinated plans to make better schedules. The arrow from the scheduler to the coordination module indicates that the coordination module takes advantage of the local scheduler’s scheduling ability to evaluate/estimate the features of actions of the remote agents by asking “What-IF” questions.

Generally speaking, there are three ways to coordinate multi-agent schedules: arrange the problem so as to avoid the need for coordination (by avoiding non-local tasks (NLTs), or using NLTs with static, known attributes), construct a special agent that functions as a centralized coordinator or associate coordination mechanisms with each individual agent. All of these solutions have good and bad points and many possible realizations in different environments. GPGP (Generalized Partial Global Planning) [7, 8] is a domain-independent, extensible approach meant to encompass all of these alternative coordination mechanisms. GPGP views inter-agent relationships like NLTs as possible coordination points and then instantiates general coordination mechanisms at those points.

A GPGP mechanism is composed of two parts: (1) a set of protocols (represented by task structures) specific to the mechanism, and (2) a pattern-directed re-writing of the extended HTNs by analyzing the task structure, especially the NLTs, to acquire scheduling information.

The actual schedule coordination process has two steps: (1) locate the coordination points in the task structure, and (2) apply the coordination mechanisms at those points. The detection step is straightforward, except that the coordination point is the parent compound task \( T_i \) of a NLT (to enable structural rewriting during the application of a mechanism).

The application of GPGP is formalized as

\[
\text{GPGP}_{\text{Apply}}(T, T_i, M, \text{role}(T_i)), \text{where } T \text{ is the task structure to be coordinated; } T_i \text{ is the compound task that contains NLT; } M = \{ M_i | i = 1, 2, 3, ... \} \text{ is a set of GPGP coordination mechanisms; role determines the role of the agent (enabler or enabler).}
\]

It has been shown that GPGP}_{\text{Apply}} for a large set of mechanisms is deterministic through carefully defined processes[7]. We will present the formal representation of the application of one complex GPGP coordination mechanism in Section 5.

**Theorem 2.** The GPGP processes for those coordination problems that can be represented with the extended HTNs are deterministic. The proof of this theorem is in [5] but give the idea here. A GPGP coordination mechanism is composed of two parts: the rewriting of the task structures and the coordination communication protocol. Based on the definition and explanation of these mechanisms in Section 5, it is easy to conclude that the coordination process is deterministic, i.e., once a coordination mechanism is selected and the coordination participants are ready, the coordination process is predetermined, and of course, not random. The detection of the coordination points has the same time complexity as traversing a tree - the hierarchical task network. The actual re-writing of the task structure is bounded based on which mechanism is chosen. The coordination communication is the only process that may introduce nondeterministic results because of uncontrollable factors, e.g., the reliability of the communication channel. The coordination protocols are implemented with timeouts, which means that a “TIME-OUT” message will be collected in order to continue the coordination process, if a certain amount of time has passed after a coordination message is sent out. With this time-out mechanism, the largest communication time per message is also a fixed time period. Based on the above description, it is easy to understand that the GPGP processes for those coordination problems that can be represented with our extended HTNs are deterministic.

## 4. COORDINATION MECHANISM SELECTION

There is an important observation that different mechanisms have correspondingly different performance characteristics, and that these can change dramatically in different environments (i.e., no one mechanism is best for all domains). Thus, it is key for the agents to autonomously select the best coordination mechanisms based on some rules/strategies. If we did this, each agent would be able to learn over time to choose the best coordination mechanism in various situations according to its knowledge of its own capabilities, its belief about the other agents, and the dynamic environmental features. Since this process is also based on the planning and scheduling processes described in the previous two sections, we call this, the autonomous selection of appropriate coordination mechanisms under various environments, the coordination strategy problem, which is presented as Equation 1.

\[
\text{Given } k \text{ NLTs } \in NLT \text{ in } E \text{ & } M, \text{ find } m_i \in M \text{ for each NLT, that maximizes } EV(E, m_1, ..., m_k).
\]

where \( E \) is a problem environment (including not just the formal planning problem \( P \) but also other characteristics explained later); the \( k \) NLTs are the related coordination points in the current solution to that problem; \( M \) is our set of seventeen GPGP coordination mechanisms; \( m_i \) is the selected \( i \)-th mechanism in \( M \) to be applied at a particular coordination point. NLT_i; \( EV \) is a performance Evaluation function under environment instance \( E \) and the coordination mechanisms \( m_1, ..., m_k \) selected in response to the \( k \) NLTs. The evaluation function provides feedback about whether a certain mechanism selection at a coordination point is good or not in various environment settings; \( m_1, ..., m_k \) are also parameters of \( EV \) because the selection of mechanisms at different coordination points may be related.

Before we propose a general solution to the coordination strategy problem, a new term coordination feature vector needs to be defined first: a coordination feature vector consists of task attributes and system attributes. Task attributes were defined early in Section 2 as the task characteristic vector \( A \): Quality, Duration, Cost, Reputation, etc.; in contrast system attributes represent characteristics of the application environment. There are two kinds of system attributes: environment-specific attributes and situation-specific attributes. Environment-specific attributes represent the physical backbones of agent coordination systems: the number of agents in the domain, whether communication is available or not, agent failure rate, communication bandwidth, etc. Situation-specific attributes represent the abstract features about the coordination situations: coordination task turn rate, local load (how busy the agent is locally), interleaved NLTs, etc. The general steps to analyze the relationship between the GPGP coordination mechanisms and different environments are as follows[7]:

1. Vectorize the coordination universe by filling in the information for the coordination feature vector;
2. Normalize the vector so that the vector elements can be formatted as input for learning algorithms;
3. Apply coordination mechanisms under various combinations of environmental values in the coordination framework to produce training data;
4. Apply learning over the training data to get the mappings, or rules, between environmental factors and the coordination mechanisms.

32
This general process, by solving the coordination strategy problem, proposes a practical method in turn to tackle the schedule coordination problem:

Given: \((T, M)\)
\(T\), an uncoordinated task network; \(M\), the set of GPQP coordination mechanisms;
1. Apply the function \(\text{GPQP}^{\text{dec}}\) to \(T\) to get a set of \(k\) coordination points: \(\text{CP} = \{\text{nlt}_i | i = 1, \ldots, k\}\);
2. While \((\text{CP} \neq \phi)\)
   - Select \(\text{nlt}_i; \text{CP} \leftarrow \text{CP} - \{\text{nlt}_i\}\);
   - Select \(m_j \in M\) according to Equation 1;
   \(T' \leftarrow \text{GPQP}_{\text{Apph}}(T, \text{nlt}_i, M, \text{role(\text{nlt}_i)})\)
3. Generate better schedules: \(\text{Schedule}(T, U) = S\);
4. Return \(S\).

5. EXTENDED SET OF COORDINATION MECHANISMS

We briefly introduced the GPQP coordination mechanisms in Section 3. Each GPQP coordination mechanism consists of a coordination protocol and a pattern-directed re-writing of the extended HTNs. Because of space constraints, we simply list these mechanisms and demonstrate the application of one particular mechanism—demotion shift dependency.

Seventeen coordination mechanisms [7] have been catalogued for the enable relationship. They are avoidance (with or without some sacrifice of quality), reservation schemes, simple predecessor-side commitments (to do a task sometime, to do it by a deadline, to an earliest-start-time, to notify or send directly when complete), simple successor-side commitments (to do a task with or without a specific Earliest Start Time), polling approaches (busy querying, timetabling, or constant headway), shifting task dependencies by learning or mobile code (promotion or demotion), various third-party mechanisms, or more complex multi-stage negotiation strategies. The input to these mechanisms are agents’ task structures with structural information and annotations on the tasks. The result of the application of a mechanism might be either a structural alteration (i.e. removing or adding tasks) or an update of the annotations on the task structure, or both. Considering the scheduling problem that a local agent may have no knowledge about the characteristics of non-local tasks, our approach removes this uncertainty and allows the local agent to make better scheduling decisions. Next we demonstrate the application of a particular GPQP coordination mechanism—demotion shift dependency; exploration of other mechanisms are in [5].

Under the assumption that \(\text{AgentA}'s\) task \(\text{Sub2}\), is dependent on the execution of \(\text{AgentB}'s\) \(\text{TaskB}\), an example structure showing this dependency is as in Figure 3: the coordination point for this dependency is the non-local task, \(\text{NLT}\), within the task branch \(\text{Sub2}\) for \(\text{AgentA}\). \(\text{TaskA}\) and \(\text{TaskB}\) are the two agents’ domain tasks; \(\text{GPQP}_{\text{DemotionB}}\) and \(\text{GPQP}_{\text{DemotionB}}\) are the coordination tasks initiated by the Demotion Shift Dependency mechanism: \(\text{TaskB}\) within the dashed box represents a modified mobile code sent from \(\text{AgentB}\) to \(\text{AgentA}\). As shown in Figure 3, we name \(\text{AgentB}\) as Predecessor and \(\text{AgentA}\) as Successor. Demotion Shift gets the name because the task structure is transferred from the predecessor to the successor. Demotion shift dependency works as follows: \(\text{AgentA}\) detects the dependency relationship with \(\text{AgentB}'s\) task, \(\text{TaskB}\); \(\text{AgentA}\) sends a coordination request message to \(\text{AgentB}\) indicating task demotion; \(\text{AgentB}\) receives the message and sends back the task information and the modified object code (removing \(\text{NLT}\)) of \(\text{TaskB}\); \(\text{AgentA}\) gets the returning message and dispatches it to its GPQP Module again: the GPQP module in \(\text{AgentA}\) unwraps the message and executes the object code to get the result and at the same time modifies the task structure so that the result is directed to the next execution task. The inserted task structures are shown in Figure 3 and the coordination protocol for Demotion is shown in Figure 4. The application of demotion shift dependency mechanism for the enable agent, \(\text{AgentA}\), is formally represented with our extended HTNs as follows:

Given \(P = (d, I, D)\); suppose \(d = (\{(d_{\text{Task}} : T)\}, \{(d_{\text{Task}} : \text{NLT})\}, \phi_{\text{cpro}}, \phi_{\text{inher}}, \phi_{\text{disinher}}, \delta_{\text{A}(T)}\) \(\Rightarrow\)

\(T' = (\{\text{NLT}_{\text{Sub2}}\} + T'_{\text{TaskB}})\)
\(L'_{\text{cpro}} = L_{\text{cpro}} - \{(\text{Ok}, \text{Ok}, \text{Sub2}_{\text{Task2}})\}\)
\(L'_{\text{inher}} = L_{\text{inher}} + \{(\text{Ok}, \text{Sub2}_{\text{Task2}})\}\)
\(L'_{\text{disinher}} = L_{\text{disinher}} + \{(\text{Ok}, \text{Sub2}_{\text{Task2}})\}\)

where \(\phi_{\text{cpro}}\) is unchanged, and \(T'_{\text{TaskB}} = T_{\text{TaskB}} - \{\text{NLT}_{\text{TaskB}}\}\) is the mobile code transferred from \(\text{AgentB}\) to \(\text{AgentA}\) to remove the \(\text{NLT}\).

For the enable agent, \(\text{AgentB}\), the coordination is to transfer modified mobile code of \(\text{TaskB}\) to \(\text{AgentA}\) as shown in Figure 3.

Demotion Shift Dependency performs efficiently when the same tasks are requested for execution repeatedly. For example, in information gathering applications, very complex wrapper agent code
can be demoted to the requester agent at the user side, especially when the wrapper is heavily loaded. *Promotion Shift Dependency*—object code transferred to the opposite direction, is also applicable in information gathering applications, which is to upload tasks from a handheld agent or thin client to a large server.

6. IMPLEMENTATION

As introduced above in this paper and as shown in the previous work [6, 7], a novel set of seventeen GPGP coordination mechanisms have been implemented and applied following the general procedure in Section 3 to reveal the potential relationships between these mechanisms and the environmental characteristics. Experimentation has been carried out in a simulated Emergency Medical Service (EMS) system. This novel set of coordination mechanisms has been theoretically proven to be effective upon various environmental conditions in the EMS domain. Specifically, the applications of selected (by greedy strategy) coordination mechanisms to the three coordination points (i.e., interdependencies) within emergency medical service procedures improve the following three evaluation methods at the same time: reducing the average first response time to 38% of the base cases without coordination, increasing the average survival rate to 12% of the base cases, and reducing the total coordination cost (communication - message transfer amount cost and task modification cost) to 61% of the base cases (details in [5, 7]).

General analytical rules (not from machine learning) about applying which mechanisms to which particular environments have been presented, e.g., a couple of sample rules are displayed below.

**Rule 1** If (true), apply *Avoidable* and *Sacrifice Avoidable* mechanisms.

**Rule 2** If (deadline is the user's main concern), apply *Coordination by Reservation*.

7. DISCUSSIONS

We will not delve into implementation details about the GPGP coordination mechanisms as shown in [7]; instead we will discuss the key questions relevant to this workshop, e.g., what to coordinate among agents, how much information be exchanged, and how to evaluate a planning approach, etc. The brief answers are simply stated below (details explored later). The task structures and task achievement status represented using EHTNs are the actual information to be coordinated: the amount of information to be exchanged among the agents depends on the relevance to the target goals, i.e., only those task structures (and their status information) related to the designated dependent goals/sub-goals will be exchanged among involving agents; performance base analysis could be employed to evaluate whether a planning approach is successful or not.

The internal structure of a DECAF agent with GPGP (coordination) Module is shown in Figure 5. These internal components work together to keep track of an agent’s current status and to carry out designated tasks for potential team goals: plan selection, scheduling, and execution. The relationship among the GPGP Module, the planner and the scheduler, as Figure 5, is a concrete implementation of the model shown in Figure 2 and explained in [7]. Here we need to point out that a complex intelligent agent's internal structure requires the integration of the underlying components as follows. A “plan file” is created by a user (agent system architect) in pre-planning procedure; the “plan file” is then fed into planner

![Figure 5: DECAF agent architecture.](image)

7.1 Look-ahead Planning

We have stated earlier that the distributed nature of multi-agent systems can afford neither centralized coordination nor the distribution of constantly updated global view across all team participants and it is difficult to coordinate agents’ activities with uncertainty, contingencies, and evaluating worth-oriented goals. The root problem is that whatever plans and schedules generated at one time (e.g., the start of a goal task) may not reflect the updated situations at other times due to uncertainties, etc. This paper has discussed an integrated coordination approach uniting pre-planning, plan generation, scheduling, planning execution, and re-planning, in the scope of multi-agent collaboration. Next, a special planning strategy, look-ahead planning, will be proposed as an applicable solution to multi-agent coordination problems. Specifically, a performance analysis will be employed to assist look-ahead planning approach at both short-term and long-term scales.

Look ahead planning has been employed as one of the decision functions that constitute production control systems in construction industry[2]. A master schedule is usually issued at the beginning of the construction phase (i.e., execution phase in agent research) specifying the entire project schedule ranging from long term coordination to terms of payments. However, such initial, total schedules

---

1Notably, *avoidable, sacrifice avoidable, and coordination by reservation* are three of the seventeen GPGP coordination mechanisms developed in earlier work [5, 7].

2The performance based planning assessment approach is applicable to self-interested agents' planning efforts given that both collaborated and self-interested coordination among a team of agents are based on objective performance analysis, i.e., whether expected goals and sub-goals have been achieved—an objective assessment.
cannot be accurately detailed too far into the future because of lack of information about actual task durations and deliverables. Thus short-term schedules are generated to coordinate and direct various trades and crews ahead several weeks into the future called look ahead schedules. The key assumption is that thinking ahead accurately in the near future would be beneficial to long-term goals.

One feature of look ahead planning is its “conceptual framework, which posits planning as a process of reducing uncertainty and maximizing throughput, counter-posing plan reliability to resource redundancy as alternative strategies for managing in conditions of uncertain workflows” [2]. This feature surprisingly approximates the author’s research idea about multi-agent coordination, which is to reduce uncertainty within uncoordinated plans for generating better schedules. The idea is to generate best accuracy for short-term planning while in order to achieve overall long-term goals.

Notably this approach is NOT bounded by local maximum issues. There are two cases: (1) no interdependency between the look-ahead plans and the future plans, and (2) there are interdependencies between the look-ahead plan and later plans. In the former case, it is obvious that the achievements of look-ahead plans will improve the overall master plan for goal tasks. In the latter case, the coordination solution (applying appropriate coordination mechanisms to resolve interdependencies) presented in early part of this paper could be employed to manage the interdependencies by removing the uncertainties.

7.2 Team Performance

Look-ahead planning is a candidate solution to multi-agent coordination, it is still a question how to evaluate both look-ahead planning in a short term and the overall plan in a long run. The solution is to analyze the team performance to figure out whether the planning is suitable or not. The performance of an agent team is an objective measurement that enables a stable analysis method versus subjective measurements. It is intuitive that better collaboration leads to better performance and better performance reflects successful task planning. Performance measurement is paramount to various applications, e.g. learning and training. Performance measurement can be viewed as an “investment” that pays dividends in information upon which decisions can be made and action can be taken. Whether dealing with team or individual performance, the ability to accurately measure is a necessity — feedback, remediation, and ultimately learning depends on it.

Cannon-Bowers and Salas [4] advanced a set of requirements for performance measurement in team training systems which have been modified by other researchers. Their research sheds a bright light about what and how to keep track of performance in order to evaluate planning approaches. First, performance measurement systems must consider multiple levels of measurement. In the simplest cases, performance can be measured at the individual and team level. Adding complexity, team performance can be measured at the holistic level. A measure of individual level properties. Execution feedbacks for improving performance determine the level of analysis for measurement. Second, performance measurement systems should consider processes in addition to outcomes. Processes are represented using EHTEH-style task structures and the associated behaviors used by participating agents to accomplish the task; whereas outcomes are the end results of these processes. Outcome measures do not provide information about how a particular outcome was reached and therefore do not provide enough information for performance diagnosis. Without process measures, it is impossible to determine whether the levels of performance outcomes observed have been reached through erroneous or correct processes. For example, a team of agents may reach some outcomes through chance alone and not through the correct reasoning and behavioral processes. In this case, assessing the outcomes in isolation from processes yields a misleading representation of the team’s true performance. Process measures are more descriptive of performance because they define the desired means of achieving a particular end. Outcome performance measures can alert related agents that a problem is present, but they provide little direction as to its behavioral causes. These must be inferred, usually from a number of possibilities. EHTNs are expressive enough to specify the processes. Third, performance measurement systems should describe, evaluate, and of course diagnose performance. By addressing these issues, a more robust view of performance can be provided. The description of performance is conceptually simple, although in practice is often very difficult due to the complex dynamic nature of tasks and the conceptual frame in which they are described. The evaluation of performance is the comparison of observed behaviors to a model or standard and judgment concerning their relative value. Performance diagnosis is the determination of the underlying causes of both effective and ineffective performance. Performance diagnosis feeds into remediation, the next requirement. Fourth, performance measurement systems should provide a basis for remediation to support the task process. Measures should inform the agents with progress, provide results and direction for improving performance, and provide a basis for selecting future training.

As noted, performance measurement is vital to understanding and evaluating team behaviors, but the degree to which performance measurement can meet this aim is capped by the ability of the measures to diagnose performance.

In summary, a set of suitable metrics for evaluating team performance could be: (1) the level of measurements (individual or team level), (2) process content (how a task network is created as a goal task decomposition) and task outcomes (task achievement status and domain results), (3) the ability to describe, evaluate and diagnose performance, and (4) a remediation mechanism to provide results and instructions for agents to improve performance.

7.3 Human Agent Collaboration

As unmanned systems have been rushed in real world applications, e.g., anti-terrorist activities and fielded emergency response systems, it is necessary to incorporate humans and agents as peers for collaborative mutual goal achievements. However, human involvement is unpredictable compared with agents’ computation based (e.g., utility-based) decision making. Humans may simply reject a task assignment simply because “I [he] don’t feel like to work today”, or “I have a family emergency”, which is impossible to model. Agents are more predictable—an agent can not reject a task assignment if being capable and available. The availability of human expertise is less predictable, but the actual expertise content is generally more dependable, which is the reason why we want to reuse human expertise as much as possible.

With human participation, one key problem is that the performance evaluation should be objective, i.e., can be calculated according to objective standards, e.g., communication load on task structure exchange, efficiency, etc., but in real life different humans with different roles may have different evaluations, opinions, and interpretations towards the same task. Executions of an agent, e.g., managers caring about the overall goal task completion before deadlines, trainers concerning with individual agent’s

3The author has clear solutions to the performance evaluation metrics and intends to share and discuss the applicability of them at the workshop based on the research of three ongoing projects subject to the approval of the funding agencies.
skill improvement, resource suppliers (self-interested) maximizing resource utilization and profit. Thus, a practical way for team performance evaluation is to identify the objective metrics as base in certain domains and then to provide flexible subjective evaluation metrics for different humans in various roles. The selection of subjective metrics set is based on human's preferences and not in the scope of this paper.

8. CONCLUSION
The introduction of EHTNs provides a general answer to the three reasons of coordination presented in [14]: (1) interdependencies are represented by the task structures—although we concentrate on enablers in this paper; other dependencies, e.g., facilitates and hindars, can also be represented; because our approach models the structural, not the semantic, information of the dependencies; the structural features of agent tasks for various interdependencies are specified in the same way based on a unified representation—EHTNs; (2) resource constraint problems—using a new plan candidate, sensing action, to discover the availability of the resources, or simply regard the resource as a special task controlled by an agent that enables other tasks by consuming/reducing a required amount of resources; (3) the partial view problem—the filling of uncertainty within agent tasks provides different views from remote agents to local agents.

Most of the previous approaches about multi-agent coordination ignored the interleaved effects of planning, scheduling, and execution on internal agent reasoning. This paper projects an integrated coordination problem and suggests an applicable general solution and a standard algorithm in Section 3 for other researchers' future explorations. Finally discussions have been focused on key questions, e.g., what to coordinate (EHTN-based task structures and task achievement status), how much info to exchange among agents (only the information relevant to interdependencies), and how to evaluate a planning approach (using team performance objective metrics and adjustments with subjective metrics for human involvement).

Additionally we briefly mention the scaling-up concern. Previous research and ideas consider HTN-based approaches to be difficult to scale: the main reason is that HTN-based coordination is usually based on small number of agents' collaborations. However, there is an effective solution that leverages the scaling-up issue if the amount of communication (of EHTN task structure) to be exchanged is exactly as needed by suitable negotiation processes (to be discussed in the workshop).

9. REFERENCES
Coordinating Planning Agents for Moderately and Tightly-Coupled Tasks

J. Renze Steenhuisen
Delft University of Technology
P.O. Box 5031, 2600 GA
Delft, The Netherlands
J.R.Steenhuisen@tudelft.nl

Cees Witteveen
Delft University of Technology
P.O. Box 5031, 2600 GA
Delft, The Netherlands
C.Witteveen@tudelft.nl

ABSTRACT
In many task-planning domains, dynamic assemblies of autonomous agents are replacing hierarchical organisations because they promise more agility. In such assemblies, interdependent tasks might be given to different agents that each make a plan for their set of tasks. The feasibility of a joint plan for the total set of tasks, however, is likely to be endangered. Autonomous planning behaviour might result in individually constructed plans that are not jointly feasible. Therefore, (plan) coordination mechanisms have to be introduced to guarantee that even if each individual agent plans its part of the tasks independently from the others, the result will be a feasible joint plan for the complete set of tasks. In previous work we have addressed a coordination mechanism for moderately-coupled tasks, i.e., tasks that are partially ordered by some precedence relation expressing that some task $t$ has to be completed before another task $t'$ can be started.

Often, however, we have to deal with more complex qualitative temporal relations between tasks. We, therefore, first concentrate on the analysis of qualitative temporal relations between tasks and, using Allen’s analysis of temporal interval relations, we show that while some of them can be expressed by precedence relations, others require the addition of a synchronisation relation. We call a set of tasks requiring both precedence relations and synchronisation relations a tightly-coupled set of tasks.

The associated problem of coordinating tightly-coupled task systems then concerns the design of a coordination mechanism ensuring the existence of a feasible plan and plan execution process even if each agent is allowed to plan its part of the set of tasks independently from the others. Although we show the associated decision problem to be intractable (3X-complete) we provide a polynomial-time approximation algorithm that produces a set of additional constraints for each of the agents involved and ensures a feasible joint solution, while the agents keep their planning autonomy.

Categories and Subject Descriptors
I.2.11 [Artificial Intelligence; Distributed Artificial Intelligence—Coherence and coordination, Multiagent systems

General Terms
Algorithms, Theory

Keywords
Planning, qualitative temporal constraints, coordination

1. INTRODUCTION
Autonomous agents are being introduced in a wide variety of domains, because they promise to increase agility. Example domains where these systems are emerging are such diverse domains as multi-modal transportation [3], crisis response [7], firefighting with unmanned aerial vehicles and air traffic control [10]. In general, the tasks that need to be performed in these domains are interdependent, require more than one agent to execute them, and often require a careful task-planning process for each individual agent. Obviously, due to the task interdependencies, some form of coordination is needed between the individual agents to ensure that the individually constructed plans are jointly feasible. However, if we assume the participating agents to be self-interested and requiring planning autonomy, it is often not desirable or feasible to use approaches that either severely restrict the autonomy of the participating agents or require intensive negotiation and plan revision in order to reach a joint solution.

In these cases, coordination mechanisms in the sense of [4] provide a framework to ensure feasibility of any joint solution obtained by enabling the individual agents to choose their preferred way to solve their part of the task. This promise of agility, however, does not come for free. To guarantee feasibility, a coordination mechanism has to (minimally) reduce the initial autonomy of the agents. The quality then of a coordination mechanism is dependent on both the severity of the restrictions imposed and the overall performance quality it ensures.\footnote{In most cases, agents are willing to reduce their autonomy when they are compensated or get a guarantee of effectiveness in return.}

In task-based multi-agent systems, at least four different phases can be distinguished in or before which some form of coordination between the agents might be required. First,
in the allocation phase, each task is assigned to some agent that is capable of completing it. Second, the order in which the tasks are to be executed is determined in the planning phase. Third, a time schedule is constructed for the tasks in the scheduling phase that is compatible with the plan. Finally, we have the execution phase in which the tasks are executed according to the constructed schedule.

Task instances are often classified with respect to whether or not they require coordination in these phases [9, 15]. First, a set $T$ of tasks is called loosely coupled when the tasks occurring in $T$ can be assigned to the agents independently. Here, each agent is able to construct an independent plan for its subset of tasks, and coordination reduces to solving the task allocation problem (i.e., who will do which tasks). Therefore, we do not need to bother about coordination for the planning or execution phase when dealing with loosely coupled sets of tasks. Examples of this category are tasks that are totally unrelated to each other, such as searching for casualties in different parts of a city.

Second, if the set $T$ of tasks is partially ordered, $T$ is said to be moderately coupled. Obviously, for these task instances, coordination is required before or during planning in order to ensure that the partial-order relation between tasks allocated to different agents is not broken. There is, however, no need to coordinate the scheduling phase if plan coordination can be guaranteed. Typical problems in this category are monitoring tasks, patient scheduling, and multi-modal transportation tasks.

Finally, if the set $T$ of tasks is not only partially ordered, but also requires the satisfaction of constraints when executing the tasks such as simultaneity constraints, the set of tasks is said to be tightly coupled. Here, not only plan coordination, but also schedule coordination is required. Examples of tightly coupled tasks are (i) extinguishing a large fire that requires simultaneous action of multiple firefighters from different angles, and (ii) simultaneously lifting a patient onto a bed.

In our approach to designing coordination mechanisms, we do not address coordination issues that have to do with either task allocation or task scheduling. We assume the task allocation phase to have been completed and we concentrate exclusively on coordination issues that have to be dealt with before the agents start to plan and schedule the set of tasks allocated to them. The reason is that we want to deal with coordination mechanisms for self-interested and autonomous agents that are assumed (i) to require planning autonomy, and (ii) are not willing to adapt their previously constructed plan in order to ensure feasibility of the joint plan and the joint schedule. The coordination mechanisms we are aiming at will provide such planning autonomy and ensure that the joint plan composed of the individually constructed plans is feasible, and also ensure that there exists a joint schedule for all the tasks that can be obtained without revising any of the underlying plans.

In previous work [2, 11], we have concentrated on designing coordination mechanisms for task-based planning domains. In particular, we have investigated the computational complexity of finding such coordination mechanisms for tasks that are moderately coupled, i.e., tasks where the only dependency relation is a partially-ordered precedence relation. The computational complexity turned out to be rather high, but we identified a polynomial-time approximation algorithm. For some domains, like the logistics domain, this algorithm offered a quite effective and almost minimal coordination mechanism for constructing multi-modal transportation plans [12] by autonomous agents.

In this paper, we extend this approach to coordination mechanisms for tasks that are constrained by qualitative temporal relations. Using a set of basic temporal constraints as distinguished by Allen [1], we first show that, in order to represent such relations in a task-based framework, it suffices to use a synchronisation relation $\preceq$ between tasks besides a precedence relation $\prec$. We show that a set of tasks with both these relations a tightly-coupled task instance. Surprisingly, it turns out that from a computational point of view designing coordination mechanisms for tightly-coupled tasks does not differ essentially from designing coordination mechanisms for moderately-coupled task instances: The decision variants of both problems are $\Sigma^p_2$-complete. Although this problem is most probably intractable, we show that there exists a very simple polynomial-time approximation algorithm that provides a (non-optimal) coordination mechanism for tightly-coupled tasks enabling agents to plan completely autonomous their part of the task while ensuring feasibility of the joint plan thus produced. From this joint plan, a joint schedule can be derived without forcing any of the agents to revise any part of their plan.

The remainder of this paper is structured as follows. In Section 2, we describe the framework for coordination multi-agent task-based planning problems. In Section 3, we define tightly-coupled task instances and partition qualitative temporal constraints into two categories (i.e., requiring moderately and tightly-coupled task relations respectively). In Section 4, we describe the way tasks constrained with tightly-coupled relations can be represented and analyse the problem of solving the coordination problem for such instances. In Section 5, our approach to coordinating temporally constrained tasks is related to other work and some directions are given for future research.

2. A FRAMEWORK FOR MODERATELY-COUPLED TASKS

In this section, a brief but necessary discussion of the existing framework for moderately-coupled tasks is given. We introduce the associated coordination problem and present some of the results obtained in previous work.

We consider a set of agents $A = \{A_1, \ldots, A_n\}$ that have to complete a set of tasks $T = \{t_1, \ldots, t_k\}$. We allow tasks to be interdependent by specifying a partially-ordered set of precedence constraints $\prec$, where $t \prec t'$ indicates that task $t$ has to be completed before task $t'$ might start. We assume that each task $t \in T$ is assigned to at least one agent $A_i$ and that, as a result, every agent $A_i$ has obtained its disjoint subset $\hat{T}_i \subseteq T$ of tasks together with the local subset $\mathcal{S}_i \subseteq \mathcal{S}$ of precedence constraints induced by $\hat{T}_i$. In order to execute its partially-ordered subset of tasks $(\hat{T}_i, \prec_i)$, agent $A_i$ needs to construct a plan $P_i$ for it. Such a plan can be simply conceived as a refinement of the partially-ordered set $(\hat{T}_i, \prec_i)$, i.e., $P_i = (\hat{T}_i, \prec'_i)$ where $\prec'_i$ is a partially-ordered extension of $\prec_i$.\n
38
If we assume the agents to plan independently, we should ensure that whatever feasible plans \( P_i \) are chosen by the individual agents \( A_i \), the simple joining of these plans constitutes a feasible global plan for the original set of tasks \( T \) satisfying all the precedence constraints (i.e., for every possible extension \( \xi \) of \( \xi' \), we should require \( \bigcup_{i=1}^{n} \xi_i \cup \xi \leq \cdot \)). It is easy to see that cycles occurring in such an extension imply a deadlock when trying to execute the resulting joint plan.

Note that, due to the interdependencies between the tasks in \( T \), tasks assigned to different agents might be dependent upon each other and therefore independent planning of the individual agents might easily lead to a deadlock as the following simple example (see Figure 1) shows.

![Figure 1: An uncoordinated task instance.](image)

In Figure 1(a), a task instance is shown where agents \( A_1 \) and \( A_2 \) can plan \( t_3 \prec t_1 \) and \( t_2 \prec t_4 \) (see Figure 1(b)). But when these plans are joined, a cycle \( \langle t_1, t_2, t_4, t_3, t_1 \rangle \) is introduced. Such a cycle indicates an infeasible global plan since it implies \( t_1 \) to precede \( t_2 \), but also vice versa. Since such a combination of individual plans is possible, we call this instance *uncoordinated*.

We can now define the pre-planning coordination problem for tasks with precedence constraints as follows: Given a task instance \( \langle (T_i)_{i=1}^{n}, \prec \rangle \), how to guarantee the feasibility of a global plan for all tasks respecting the precedence constraints when every agent \( A_i \) is allowed to autonomously construct a plan for its set \( T_i \) of subtasks, respecting only the set \( \xi_i \) of local constraints?

Plans are feasible when the tasks are partially ordered, which graphically boils down to not containing any directed cycles. Note that referring to Figure 1 this task instance becomes coordinated if we add, e.g., the constraint \( t_1 \prec t_3 \) to the set of local constraints of agent \( A_1 \). Since then the only plan \( A_1 \) can produce is the plan where \( t_1 \) is executed before \( t_3 \), while agent \( A_2 \) has three options: executing \( t_3 \) and \( t_4 \) concurrently, \( t_3 \) before \( t_4 \), or vice-versa. Clearly, no combination of the individual plans creates a cycle.

It is not difficult to prove that, in general, for every uncoordinated task instance, there is a set \( \Delta \) of local precedence constraints that when added to the existing set of constraints and taking the transitive closure, transforms the uncoordinated instance into a coordinated one [2]. In order to minimise the loss of freedom for the individual agents, we have to identify a *minimum set* of such additional precedence constraints. We call this the coordination problem for *moderately-coupled task instances*.

**Coordinating moderately-coupled task instances**

**INSTANCE:** Moderately-coupled task instance \( \langle (T_i)_{i=1}^{n}, \prec \rangle \) and positive integer \( K \).

**QUESTION:** Does there exist a coordination set \( \Delta \) with \( |\Delta| \leq K \) such that the task instance \( \langle (T_i)_{i=1}^{n}, (\xi_i \cup \Delta) \rangle \) is coordinated?\(^2\)

In previous work [11], this problem has been studied extensively. It turns out that this problem is \( \Sigma_2 \)-complete in general, and \( \Pi_2 \)-complete when the number of agents is bounded by some constant. In addition, it was shown that this coordination problem is \( \mathcal{APX} \)-hard and that a constant-ratio approximation algorithm is not likely to exist [13].

There is at least one planning domain, the logistics domain, where this framework applies. Here, we have a number of precedence constraints for multi-modal transportation tasks to be executed by autonomous transportation agents. Applying the above stated approach, we can show that there exists no polynomial algorithm for either a cooperative or a selfish multi-agent system solving the multi-modal transportation task with approximation ratio \( \varepsilon < 1.2 \) unless some generally accepted conjecture in complexity theory is violated [12].

There is, however, a very simple polynomial-time approximation algorithm that succeeds in finding an approximate but sufficient coordination set for distributed tasks with precedence constraints.

This algorithm is based on the following idea: In constructing a local plan for \( (T_i, \prec_i) \), each agent \( A_i \) can safely start with the subset \( T_i \) of tasks that are not dependent upon other tasks, the so-called prerequisite-free tasks. Each agent, therefore, sends its set of prerequisite-free tasks (according to \( \prec_i \)) to a blackboard managing the inter-agent precedence constraints. The blackboard checks which of these tasks is also globally prerequisite free and sends the resulting set to the agent and removes all constraints pertaining to these tasks. Each agent now stores the set of tasks obtained from the blackboard in the set \( T_i \) and removes this set from the original set \( T \). As a result, other tasks in \( T \) will become prerequisite free in the next round and each agent again selects its subset of prerequisite-free tasks, sends it to the blackboard and stores the result in \( T_i \), etc. After at most \( k = |T| \) iterations, all tasks have been selected exactly once as a prerequisite-free task. For each agent \( A_i \), we now have a set of disjoint subsets \( T_i^k \) for \( k = 1, \ldots, |T| \), where \( T_i^k \) denotes the (possibly empty) set of prerequisite-free tasks selected by agent in iteration \( k \). The resulting coordination set \( \Delta \) is constructed as follows: \( \Delta \) is the union of the sets \( \Delta_i \) for agent \( A_i \). These sets \( \Delta_i \) are obtained as follows:

1. Remove all empty subsets \( T_i, k \) and let \( m \) be the number of remaining subsets.
2. A precedence constraint \( t \prec t' \) is added to \( \Delta_i \) for every pair of tasks \( t \in T_i \) and \( t' \in T_i^{k+1} \), for every \( j = 1, \ldots, m - 1 \).

\(^2\)For any relation \( \rho \), the transitive closure of \( \rho \) is denoted by \( \rho^+ \).
It can easily be shown that although not creating a minimum coordination set the resulting set $\Delta$ is sufficient for coordination. Pseudo code for this distributed algorithm delivering the sets $T_k^i$ is given below.

\textbf{Coordination-by-Partitioning} $(T_i, \sim_i)$

\begin{enumerate}
\item $k \leftarrow 1$
\item \textbf{while} $T_i \neq \emptyset$
\item \hspace{1em} Get the set $F_i$ of prerequisite-free tasks in $T_i$
\item \hspace{1em} Send $F_i$ to the blackboard $B$
\item \hspace{1em} Wait for the response $T_k^i \subseteq F_i$ from $B$
\item \hspace{1em} $T_i \leftarrow T_i \setminus T_k^i$
\item \hspace{1em} $k \leftarrow k + 1$
\item \textbf{return} $(T_1^i, \ldots, T_k^i)$
\end{enumerate}

In previous work [14], we showed that this algorithm is a constant-ratio approximation algorithm for coordinating problem instances in the \textsc{logistics} domain. The algorithm above is a $1.25$-approximation algorithm for this domain, where—as stated above—$1.25$-approximation algorithms are the current lower bound.

3. TASKS AND TEMPORAL CONSTRAINTS

The framework discussed above has been used to investigate pre-planning coordination problems for moderately-coupled task instances, i.e., sets of tasks with precedence constraints.

Many planning domains, however, such as airport planning, manufacturing, and supply-chain management require the ability to use \textit{temporal relations} to constrain the execution of a set of tasks especially with respect to the \textit{time intervals} certain tasks should be executed. Several of such \textit{qualitative temporal constraints} have been identified for constraining time intervals in Allen's time interval algebra [1]. As we will briefly show, all these qualitative temporal constraints can be represented in a task-based framework using precedence constraints and an additional type of constraints, the \textit{synchronisation constraints}.

Neglecting the converse of each of these relations.

3.1 Representing Allen's temporal relations in a task-based framework

Allen identifies seven temporal relations between time intervals: \textit{before}, \textit{overlaps}, \textit{during}, \textit{meets}, \textit{starts}, \textit{finishes} and \textit{equals} [1]. Because these relations are defined on intervals instead of tasks, we assume that every task $t$ can be represented as an interval $[t_s, t_e]$ with $t_s < t_e$ representing the starting point of task $t$ and the end of start $t$.

We recall that the \textit{overlaps} relation constrains two tasks to partially overlap (like running in steppe chase), and that the \textit{during} relation constrains a task to be executed between the starting and ending of another task's execution. Therefore, this new representation for tasks enables us to translate three of the qualitative temporal constraints: \textit{before}, \textit{overlaps}, and \textit{during} as follows: We start by splitting the tasks $t_1$ and $t_2$ into time intervals with end points $t_{1s}, t_{1e}, t_{2s}, t_{2e}$, respectively, and constrained by $t_{1s} < t_{1e}$ and $t_{2s} < t_{2e}$. Now, we can rewrite $t_1 \textbf{ before} t_2$ as $t_{1s} < t_{1e} < t_{2s} < t_{2e}$ (see Figure 2(a)), $t_1 \textbf{ overlaps} t_2$ as $t_{1s} < t_{2s} < t_{1e} < t_{2e}$ (see Figure 2(b)), and change $t_1 \textbf{ during} t_2$ into $t_{1s} < t_{1e} < t_{2s} < t_{2e}$ (see Figure 2(c)). Note that these three constraints have in common that the endpoints of the time intervals are not allowed to coincide (i.e., for every pair of time points a precedence relation is defined). We conclude that the basic task framework with precedence constraints suffices to represent these three temporal relations.

Contrary to the previous three constraints, using the remaining constraints \textit{meets}, \textit{starts}, \textit{finishes}, and \textit{equals} end points of tasks need to coincide. Such synchronisation cannot be represented by precedence constraints between the end points. Therefore, we have to introduce the notion of \textit{synchronised events} colliding two synchronised time points $t_{1s}, t_{2s}$ into one $t_{1/2}$ and then constrain the tasks with precedence constraints. In this way we can rewrite $t_1 \textbf{ meets} t_2$ as $t_{1s} < t_{2s} < t_{1e} < t_{2e}$ (see Figure 2(d)), $t_1 \textbf{ starts} t_2$ as $t_{1s} < t_{1e} < t_{2s} < t_{2e}$ (see Figure 2(e)), $t_1 \textbf{ finishes} t_2$ as $t_{1s} < t_{1e} < t_{2e} < t_{2e}$ (see Figure 2(f)), and $t_1 \textbf{ equals} t_2$ as $t_{1s} < t_{1e} < t_{1s} < t_{2e}$ (see Figure 2(g)).
Remark 1. Notice that in $t_1$ starts $t_2$ it is correct to include $t_1 \prec t_2$ as translated from Allen's formalism [1]. However, we could leave out this additional precedence constraint between the end points without any problem. In this way, we are actually representing, in terms of Allen, the disjunction relation $t_1$ starts $t_2 \lor t_2$ starts $t_1$. A similar remark can be made for the finishes constraint.

Summarising, it turns out to be possible to represent all basic qualitative temporal constraints as used by Allen into a task-based framework if we are prepared to introduce the notion of a synchronised task or event. Tasks that are subject to temporal relations and can be represented by precedence constraints are {moderately}-coupled tasks, while tasks that use temporal relations requiring synchronisation are {tightly}-coupled tasks. This corresponds to the distinction made in Section 1, which was based on whether or not coordination was needed during plan execution. This exactly is covered by the synchronisation constraints, because the tightly coupled tasks not only need to have the same place in the partial order but need to be scheduled and executed synchronously.

We will now extend our coordination framework to include synchronisation constraints in order to be able to represent qualitative temporal constraints.

4. COORDINATING THE PLANNING OF TIGHTLY-COUPLED TASKS

We want to extend the task-based framework we discussed above to deal with tightly-coupled task instances. To represent such instances we need to include synchronisation information.

We define a tightly-coupled task instance then as a triple $((T_j)^{\leq 1}, \prec, \simeq)$, where $\prec \subseteq (T \times T)$ is a precedence relation and $\simeq \subseteq (T \times T)$ represents the synchronisation relation, that is $t \simeq t'$ holds if $t$ and $t'$ are tasks that have to be synchronised. We consider $\simeq$ to be an equivalence relation on $T$ and $\prec$ to be a partial order on $T$. The combination of $\prec$ and $\simeq$ satisfies the following natural properties:

1. $(\prec \circ \simeq) \subseteq \prec$ and $(\simeq \circ \prec) \subseteq \prec$, i.e., $(t \prec t'$ and $t' \simeq t''$ implies $t \prec t'')$ and $(t \simeq t'$ and $t' \prec t''$ implies $t \prec t'')$.

2. $\prec$ and $\simeq$ are orthogonal relations, that is $\prec \cap \simeq = \emptyset$.

Tightly-coupled task instances are said to be coordinated if the combination of individually feasible plans always results in a joint plan that respects both the precedence constraints and the synchronisation constraints.

Therefore, we say that an instance $((T_j)^{\leq 1}, \prec, \simeq)$ is coordinated if for every set $\{T_j\}^{\leq 1}$ of plans that (locally) respect $\prec$ and $\simeq$, there exists a function $s : T \rightarrow \mathbb{Z}^+$ assigning a time point $s(t)$ in $\mathbb{Z}^+$ to each task $t$ in $T$ such that for each $t_1, \ldots, t_n$ and for all $t, t' \in T$, (i) $t \prec t'$ implies $s(t) < s(t')$, (ii) $t \simeq t'$ implies $s(t) = s(t')$, (iii) $t \prec t'$ implies $s(t) < s(t')$, and (iv) $t \simeq t'$ implies $s(t) = s(t')$.

Remark 2. Notice that, in general, violations of the synchronisation constraints in a tightly-coupled task instance can be easily detected if we consider a pair of synchronised tasks as a single task. Given an instance $((T_j)^{\leq 1}, \prec, \simeq)$ of a tightly-coupled task-planning instance, let us define the associated moderately-coupled task-planning instance by the tuple $((T_j)^{\leq 1}, \prec, \simeq)$ where $(T_j)^{\leq 1}$ consists of the equivalence classes modulo $\simeq$ of the set of original tasks and $\prec$ is the associated precedence relation defined on $T$ where the representative $t$ of each equivalence class $[t]_\simeq$ inherits all the precedence relations occurring in $\prec$.

For example, in Figure 4, the moderately-coupled task instance associated with the tightly-coupled instance depicted in Figure 3(b) is given. In this case, the synchronised pair $(t_1, t_5)$ is represented by $t_1$ and the the synchronised pair $(t_5, t_7)$ by $t_5$. It is clear that synchronisation constraints are violated by the occurrence of an inter-agent cycle in the plan of agent $A_1$.

The example we discussed above suggests that in general, a tightly-coupled task instance $((T_j)^{\leq 1}, \prec, \simeq)$ should satisfy two conditions in order to be coordinated:

1. Tasks in shared synchronisation constraints should be ordered. Whenever there exist tasks $t_{i_1}, t_{i_2} \in T_i$ and $t_{j_1}, t_{j_2} \in T_j$ such that $t_{i_1} \simeq t_{i_2}$ and $t_{j_1} \simeq t_{j_2}$ then either $t_{i_1} < t_{i_2}$ and $t_{j_1} < t_{j_2}$ or $(t_{i_1} < t_{i_2} \land t_{j_1} > t_{j_2})$ or $(t_{i_1} > t_{i_2} \land t_{j_1} < t_{j_2})$.

2. If we abstract from the synchronisation relation, the resulting moderately-coupled task instance should be coordinated. This guarantees that the joint plan with respect to the precedence relation is feasible.
These conditions together enable us to reduce the test for being tightly coordinated to the test for being moderately coordinated as expressed in the following proposition.

**Proposition 1.** Consider a tightly coupled task instance \( (\{(T_i)_{i=1}^n, \preceq, \Xi\}) \). This instance is (tightly) coordinated whenever the following conditions are satisfied:

1. whenever there exist tasks \( t_{j,1}, t_{j,2} \in T_i \) and \( t_{j,3}, t_{j,4} \in T_j \) such that \( t_{j,1} \preceq t_{j,3} \) and \( t_{j,2} \preceq t_{j,4} \) then either \( (t_{j,1} \prec t_{j,2} \text{ and } t_{j,3} \preceq t_{j,4}) \) or \( (t_{j,1} \preceq t_{j,2} \text{ and } t_{j,3} \prec t_{j,4}) \) holds;

2. the moderately coupled task instance \( (\{(T_i)_{i=1}^n, \preceq\}) \) is coordinated. Here, \((T_i)_{i=1}^n\) consists of the representatives (in \( T_i \)) of the equivalence classes of \( \Xi \).

**Proof.** See Appendix A for a sketch of the proof. \( \square \)

Using the results we have obtained for establishing the computational complexity of designing coordination mechanisms for moderately coupled task instances, this proposition can be used directly to establish the complexity of designing tightly coupled task instances. Let us consider the following decision variant of the problem of coordinating tightly coupled task instances.

**Coordinating tightly coupled task instances**

**INSTANCE:** Tightly coupled task instance \( (\{(T_i)_{i=1}^n, \preceq, \Xi\}) \) and positive integer \( K \).

**QUESTION:** Does there exist a coordination set \( \Delta \) with \( |\Delta| \leq K \) such that the instance \( (\{(T_i)_{i=1}^n, \preceq \cup \Delta, \Xi\}) \) is coordinated?

To establish the complexity of this problem, we know that it should be \( \Sigma^p_2 \)-hard, because the (contained) coordination problem for moderately coupled task instances is known to be \( \Sigma^p_2 \)-complete [2]. Moreover, if we guess a coordination set \( \Delta \), the complexity of verifying coordination of the extended instance \( (\{(T_i)_{i=1}^n, \preceq \cup \Delta, \Xi\}) \) can be accomplished by verifying the conditions stated in Proposition 1. Obviously, verification of the first condition can be done in polynomial time, while we have shown in [2] that verifying the second condition can be done in nondeterministic polynomial time.

**Proposition 2.** Coordinating tightly coupled tasks is \( \Sigma^p_2 \)-complete.

Surprisingly, viewing the problem computationally, coordinating moderately-coupled and tightly-coupled problems do not differ significantly.

### 4.1 An approximation algorithm for solving tightly coupled instances

The problem of coordinating tightly coupled task instances is APX-hard, because APX-hard problem of coordinating moderately coupled task instances is contained. However, we will show that with some minor modifications, the partitioning algorithm (see Section 2) can be used also to solve a tightly coupled task instance.

First of all, observe that whenever two tasks \( t \in T_i \) and \( t' \in T_j \) are synchronised, they will be selected in exactly the same round \( k \) of the algorithm used by agent \( A_i \) and by agent \( A_j \). This is a trivial consequence of the fact that \( t \cong t' \) implies that \( \{x \mid x \prec t\} = \{y \mid y \prec t'\} \), hence they have the same set of predecessors and, therefore, will be elected as prerequisite free in exactly the same round \( k \). This implies that for local synchronisation constraints \( t \cong t' \), the existing algorithm does not need to be adapted, since no additional precedence constraints between these tasks will be added.
Therefore, the only problem to solve is to prohibit that two pairs of synchronised tasks \( t_i t_j, t_k t_l \) such that \( t_i \preceq t_j \) and \( t_k \preceq t_l \) are selected in the same round \( k \), while \( t_i t_k \in T_i \) and \( t_j t_l \in T_j \) for some \( i \neq j \). For then the first condition of Proposition 1 could be violated by independently constructed plans by agent \( A_i \) and \( A_j \).

One way then to adapt this algorithm to obtain a solution is to adapt the blackboard in such a way that it is aware of both precedence relations and synchronisation constraints. Upon receiving the locally prerequisite-free sets \( F_i \) from the agents \( A_i \), the blackboard selects maximal subsets \( T_k \) from these sets \( F_i \) such that (i) all tasks occurring in \( T_k \) are globally prerequisite-free and (ii) these sets satisfy the property that whenever there are tasks \( t_i \preceq t_j \) and \( t_k \preceq t_l \) such that \( t_i t_k \in F_i \) and \( t_j t_l \in F_j \), only one of these pairs occurs in the resulting sets \( T_k \) sent back to the agents. As a result, all such shared tasks will be totally ordered, since each of them will appear in a set returned in a new round \( k \) and tasks occurring in round \( k \) will precede any task occurring in a round \( k' \) \( > k \) by the way the additional ordering constraints are added.

Let us give a final example to solve the coordination problem for the task instance depicted in Figure 3(a) using this approximation algorithm.

In the first round, agent \( A_1 \) will send \( F_1 = \{ t_1, t_3 \} \), while agent \( A_2 \) will send \( F_2 = \{ t_5, t_7 \} \) to the blackboard. The blackboard checks their prerequisite freeness and then detects that the first condition of Proposition 1 is violated. It selects \( t_1, t_3 \) as the single pair of synchronised constraints and sends back the sets \( T_1 = \{ t_1 \} \) and \( T_3 = \{ t_3 \} \). The agents adapt their set of local tasks and select again a set of prerequisite-free tasks: \( F_1 = \{ t_3, t_2 \} \) and \( F_2 = \{ t_5, t_3 \} \). Both sets are checked by the blackboard and returned: \( T_2 = T_3 = F_2 \). The agents remove these tasks from the set of to be selected tasks and now \( F_1 = \{ t_4 \} \) and \( F_2 = \{ t_8 \} \). Both are prerequisite-free and returned to the agents.

After removal of these tasks both sets of local tasks are empty. As a result of ordering the sets \( T_k \), the following constraints are added to \( T_1 \): \( t_1 \preceq t_3 \) and \( t_2 \preceq t_4 \). Likewise, \( t_5 \preceq t_8 \) and \( t_6 \preceq t_8 \) are added to \( T_2 \). The reader might check that indeed the resulting task instance is coordinated.

5. DISCUSSION AND CONCLUSIONS

We introduced a framework for specifying task instances with qualitative temporal constraints, we analysed the coordination problems for these task instances, we analysed the computational complexity of this problem, and we provided a fast approximation algorithm to find a coordination set.

As we have remarked before, coordination is needed to guarantee that using local planning autonomy does not cause conflicts to the global goal. In coordination, a distinction can be made between \textit{pre}, \textit{interleaved}, and \textit{post}-planning coordination. Both interleaved and post-planning coordination assume communication to be available during and after planning and thus during execution. Since it is not unlikely that communication is lost in a crisis situation, or that agents are unwilling to revise their plans, interleaved and post-planning coordination is not always applicable to all domains. Therefore, we used a pre-planning approach to coordination, while other approaches might be relevant to future extensions of this work. For example, although post-planning coordination techniques sometimes do not suffice, they might be used as an additional tool in case communication is available. In addition, techniques applied in pre-planning coordination constitute a source of inspiration for future research in pre-planning coordination. For instance, the Partial-Order Causality Link (POCL) framework \cite{5}, that is used in post-planning coordination, allows symmetric \textit{concurrency} and \textit{non-concurrency} relations to constrain tasks. These constraints are very useful for describing, for instance, disaster plans. Note that the concurrency relation is more or less comparable to the synchronisation relation in our framework (see Section 4).

With respect to the use of temporal constraints, existing work on plan coordination has been limited. In schedule-coordination approaches often used a framework that allows more qualitative temporal information to be used. For example, in the \textit{Temporal Constrained Satisfaction Problem} (TCSP) \cite{6} framework, it is possible to represent arbitrary intervals of temporal distances between time points. Some work on distributed autonomous scheduling in this domain has been proposed by Hunsberger \cite{8} introducing a \textit{Temporal Decoupling Problem} on STPs, a special subset of TCSP, which can be classified as a pre-scheduling coordination approach. Analogously, our coordination method could be called a \textit{Plan Decoupling Problem}, because it resolves all interdependencies between agents on a plan level instead of a schedule level. In the future, it would be interesting to combine the pre-planning and pre-scheduling coordination approaches using an even more expressive framework including both qualitative and quantitative temporal information.

Acknowledgements

J. René Strooper is supported by the Dutch Ministry of Economic Affairs, grant nr. 801862. The ICIS project is hosted by the DECIS Lab, the open research partnership of the Netherlands Institute for the University of Amsterdam, the University of Amsterdam, and the Netherlands Foundation of Applied Scientific Research (TNO).

6. REFERENCES

\begin{thebibliography}{9}
\end{thebibliography}
APPENDIX

A. PROOF OF PROPOSITION 1 (SKETCH)

Consider a tightly-coupled task instance \(\{(T_i)_{i=1}^n, \prec_i, \succeq_i\}\) satisfying the conditions stated in the proposition. For every agent \(A_i\), let \(P_i = (T_i, \presucc_i, \succeq_i)\) be an arbitrarily chosen local plan satisfying the conditions, that is, \(\prec\) is a partial order relation extending \(\succeq\), \(\succeq_i\) is an equivalence relation extending \(\presucc_i\), \(\succeq_i \cap \succeq_i = \emptyset\), and \(\succeq_i\) is both left- and right-closed under composition with \(\succeq_i\).

We have to show that there exists a function \(s : T \rightarrow \mathbb{Z}^+\) satisfying all local and global precedence and synchronisation constraints. Clearly, there exists such a function \(s_i : T_i \rightarrow \mathbb{Z}^+\) for every plan \(P_i\) satisfying all local precedence constraints and synchronisation constraints. We only have to prove that from these local functions \(s_i\) a function \(s\) can be constructed that also satisfies all inter-agent precedence constraints and all inter-agent synchronisation constraints.

First, we construct a total ordering of all these inter-agent constraints that satisfies the following conditions.

1. Whenever \((x, y)\) is an inter-agent precedence constraint, and \(x\) occurs in \(T_i\), then every inter-agent constraint \((u, v)\) (precedence or synchronisation constraint) such that \(v \prec_i x\) should occur before \((x, y)\) in the ordering.

2. Whenever \((x, y)\) is an inter-agent synchronisation constraint between \(T_i\) and \(T_j\), then every inter-agent constraint \((u, v)\) (precedence or synchronisation constraint) such that \(v \prec_j x\) or \(v \prec_j y\) should occur before \((x, y)\) in the ordering.

If such an ordering exists, it is very easy to construct a schedule \(s\) satisfying all constraints as follows: We use the set \((s_1, s_2, \ldots, s_n)\) of local schedules constructed and process the inter-agent constraints one by one in the order as specified above. If all inter-agent constraints have been processed, we construct a joint schedule \(s(t)\) by defining \(s(t) = s_j(t)\) if \(t \in T_j\).

The correctness of this procedure then follows from the fact that in this process of adapting to a new inter-agent constraint, we never invalidate earlier satisfied constraints. To see this we specify the adaptation procedure as follows: Let \((s_1, s_2, \ldots, s_n)\) be the current set of local schedules and let \((x, y) \in (T_i \times T_j)\) be the first inter-agent constraint in the ordering not yet processed.

If \(x \prec y\), then adapt \(s_j\) as follows: if \(s_j(y) < s_i(x)\) then for \(z \prec y\) and all \(z \in T_j\) such that \(y \prec z\), let \(s_j(z) := s_j(z) + (s_i(x) - s_j(y)) + 1\). If \(s_j(y) > s_i(x)\), no adaptation is necessary and the constraint is satisfied. This adaptation ensures that the set of local schedules also satisfies this precedence constraint \((x, y)\).

If \(x \equiv y\), then set \(s_i(x) := \max\{s_i(x), s_j(y)\}\) and we adapt all successors of the updated function accordingly, i.e., if \(s_i(x) < s_j(y)\) then for all \(z\) such that \(x \prec z\), let \(s_i(z) := s_i(z) + (s_j(y) - s_i(x))\). This ensures that the resulting set of schedules also satisfies the synchronisation constraint.

The ordering between the constraints now ensures that when the \(s\) value of a task \(t\) is updated, all the values of all tasks "below" \(t\) have obtained their definitive value and do not need to be adapted. Moreover, although \(t\) might be involved in more than one update operation, its value \(s(t)\) will only increase and, therefore, never invalidates values assigned to tasks below \(t\). Therefore, at the end, the schedule \(s\) composed as \(s(t) = s(t)\) if \(t \in T_j\) satisfies all constraints.

The only detail left is to prove that such an ordering as defined above exists. But this is easily seen with \(\bigcup_{i=1}^n \succeq_i\) being acyclic and the conditions enforced on \(\succeq\).